

ER Mapper Professional

Batch Scripting Training Handbook

www.ermapper.com

ER Mapper and ER Storage software and documentation is proprietary to

Earth Resource Mapping Ltd.

Copyright © 1988, 1989, 1990, 1991, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000 Earth Resource Mapping Ltd

All rights reserved. No part of this work covered by copyright hereon may be reproduced in any form or by any means - graphic, electronic, or mechanical - including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

While every precaution has been taken in the preparation of this manual, we assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

PostScript is a registered trademark of Adobe Systems, Inc.

Unix is a registered trademark and OPEN LOOK is a trademark of AT & T.

X Window System is a trademark of Massachusetts Institute of Technology.

Windows 95, Windows 98, Windows NT and Windows 2000 are trademarks of Microsoft Corp.

Sun, Sun-4, SunOS, SPARCstation, Solaris, Open Windows are trademarks of Sun Microsystems, Inc.

ARC/INFO and ArcView are trademarks of Environmental Systems Research Institute, Inc.

Autodesk World, AutoCAD and AutoCAD MAP are trademarks of Autodesk, Inc.

MapInfo is a trademark of Mapinfo Corporation.

GenaMap is a trademark of GENASYS, Inc.

ORACLE is a trademark of Oracle Corporation.

Motif is a trademark of Open Software Foundation, Inc.

All other brand and product names are trademarks or registered trademarks of their respective owners.

Table of Contents

1	Introduction to ER Mapper Scripting Language	1
	Input to batch process.	3
	ER Mapper Objects.	11
	Output.	45
	Library of batch scripts	48
2	Creating an image Tiff file	51
	About image tiff files	51
	Hands-on exercises	52
	Exercises:	55
3	Creating a toolbar file	57
	About toolbar files	57
	Hands-on exercise	59
	1:Creating a toolbar file using a CALLBACK function	59
	Available callback options	61
	2:Add an existing batch script file to the toolbar file	63
	Exercises:	65
4	Variables and input	67
	About variables in batch scripts	67
	Hands-on exercises	67
	1:Creating a batch script with variables	68
	2:Input	71
	Exercises:	75
5	Image manipulation, Density slicing and RGB composite	77
	About image manipulation	77
	Hands-on exercises	78
	1:Densitysliced image	78
	2:RGB composite	84
	Exercises:	90

6 Wizard Scripting	91
7 Wizard Scripts, Densityslicing and RGB composite	105
About wizards	105
Hands-on exercises	106
1: Wizard : Densityslicing	106
Exercises:	123
8 Hardcopy printing	125
About hardcopy printing	125
Hands-on exercises	128
Printing in Batch Script	129
Printing in Wizard Script	132
Exercises:	135
9 Formula	137
Formula - Quick Reference	138
Full Specifications	142
Operators	145
Standard Functions	146
Special Functions	147
Special Constructs	149
Dataset and Region Statistics	150
Dataset and Region Statistics with Band Lists	154
Numeric Constants	156
Symbolic Constants	156
Special Constants	156
10 Ratio and Principal Component Analysis	159
Hands-on exercises	159
1: Setting up the batch script button	160
1:Band ratio	161
1: Setting up the batch script button for PCA	164
2:Principal Component Analysis	166
Exercises:	172

11 Wizard Ratio, RGB-ratio and PCA 173

Hands-on exercises 173

1: Setting up the wizard button 174

1: Wizard - Band ratio, RGB-ratio & PCA 178

Exercise: 195

12 Multi-tasks Wizard 197

Multi-tasks in a wizard 197

Hands-on exercises 197

1: Setting up the wizard button 198

Exercises: 212

13 Wizard: Colordrape images in multisurfaces 213

About wizards creating images in multi-surfaces 213

Hands-on exercises 214

1: Setting up the wizard button 215

Exercises: 232

14 Scripting reference 235

Operators 235

Mathematical functions 236

Variables 236

Page size options 240

Keywords 241

Flow control 243

Including files 244

Error reporting 244

Script Commands - Alphabetical listing 245

Contents

Introduction to ER Mapper Scripting Language

ER Mapper scripting language is developed by the programmers of ER Mapper in Perth and is included as part of the ER Mapper software. The ER Mapper scripting language is comprehensive. With some training and the understanding of some basic logic of programming you can easily write batch scripts for specific image processing techniques.

Scripts are written using a text editor and are saved as **text** files. You can call the **Microsoft Notepad** text editor from the **Utilities** menu on the main menu of ER Mapper. To call the Notepad text editor menu select **Batch Script** from the drop_down list of **Utilities**. And from the drop_down list of **Batch Script** select **Create a Batch Script**. **Choose a name for the new script** dialog box appears. Type in a name for your script and click the **OK** button. The **Microsoft Notepad** text editor with a skeleton script appears. After writing your script save it in the 'ERMAPPER/batch' directory. You can also use your favourite text editor to write

your script and save it in the ‘ERMAPPER/batch’ directory. Scripts are run in a batch engine separate from ER Mapper. To display an algorithm written in a script you need to copy the algorithm to ER Mapper and display it in an image window.

To include a batch script in a toolbar button, wizard button or menu, you must edit the appropriate toolbar or menu file. Each of the ER Mapper toolbars and menus is defined by a configuration file. These files are stored in the ‘\$ERMAPPER/config’ directory. The structures of the toolbar and menu configuration files are virtually the same except that in the toolbar configuration file you must specify the icon to be included on the toolbar or wizard buttons. Any configuration toolbar ‘*.bar’ files in the ‘ERMAPPER/config’ directory are automatically listed in the ER Mapper toolbar menu in alphabetical order.

This ER Mapper Batch Scripting Training Handbook describes and teaches you how to write batch and wizard scripts and include them in toolbars. Users can process images very easily and quickly through toolbar buttons of batch and wizard scripts.

Toolbars are made up of three parameters and an option separator:

Buttons	An image in tiff file format is required to be placed in the icon of the toolbar or wizard button. The tiff file (*.tif) must be in the ‘ERMAPPER/icons’ directory and must be a 24-bit 16x16 pixel tiff file. The tiff file must be enclosed in two sets of quotation marks but without the .tif extension. For example, "New" refers to the ‘New.tif’ file in the ‘ERMAPPER/icons’ directory.
Tool Tips text	This text is copied to the small text window that appears just below the cursor when a user points to the toolbar or wizard button with the mouse cursor. The text must be enclosed in two sets of quotation marks. For example, "New Image Window"

Program calls

A command to be executed when the option is selected. This can be one of the following forms:

CALLBACK function name - for calling ER Mapper internal functions. No quotation mark is necessary for the function name. For example, **CALLBACK zoom** (calling zoom internal ER Mapper function)

BATCH scriptname - for specifying a batch script to be run. The batch script must be enclosed in two sets of quotation marks. For example, **BATCH "Copy_Window"** (calling 'Copy_Window.erb' batch script file in the 'ERMAPPER/batch' directory).

Option separators

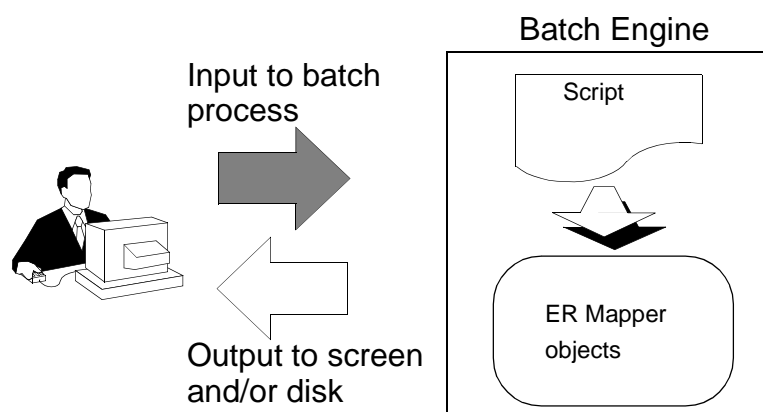
If you want to include many toolbar buttons in a toolbar you can separate the buttons using dashes in two sets of quotation marks as shown below. Since no toolbar button or wizard is used there is no need for Tool Tips text and Program call but just empty second and third sets of quotation marks.

"-----" "" ""

Batch scripts usually perform the following overall functions:

- Input data or arguments from a user
- Perform actions on specified objects
- Output the results of the actions to screen and/or to disk

Input to batch process.



Once a batch script has been invoked by a command from the user, it generally requires the input of data or arguments before it can perform any processing. In some unusual situations this input information is "hard coded" into the script so

that, apart from editing the script, the user has no control over the batch process. This is generally sufficient for “one-off” scripts which are developed to perform a very specific set of tasks. It is often preferable to have inbuilt default arguments which are used by the batch process in the absence of any from the user. This makes the batch script more flexible and, thus, more efficient.

You can also pass arguments to scripts from menu (.erm), and toolbar (.bar) files and dynamic link choosers. See Chapter 19, “Menu and toolbar files (.erm) and (.bar)”, and Chapter 21, “Dynamic Links menu dynamiclinks.erm” of the Customizing manual.

Methods for the user to input data are as follows:

- Include arguments in the command line string.
- Provide on-screen dialog boxes with controls and fields
- Provide wizards which interactively prompt users to input data.

Command line arguments

This method is used when the user is running the batch file from a command line using the `ermapper -b` command. The command string syntax is as follows:

```
ermapper -b batch_script [argument1 argument2]
```

For example:

```
ermapper -b copy_file image1.ers image2.ers
```

The hypothetical batch script, `copy_file.erb` copies a image file `image1.ers`, specified by `argument1`, to file `image2.ers`, specified by `argument2`.

To run your script from a command line you must have the following line at the end of the script:

$$\text{exit } n$$

Where n is the exit code returned from ER Mapper.

$n = 0$ exits the script and leaves ER Mapper open.

$n = >0$ exits the script and closes ER Mapper.

Dialog boxes

There are a number of commands which create dialog boxes for users to enter data.

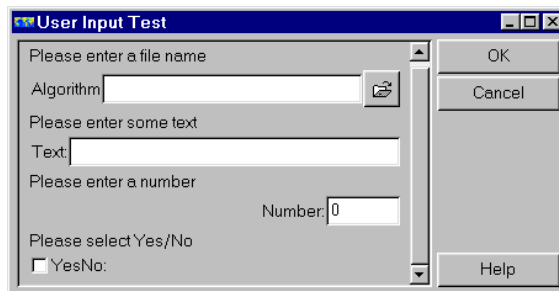
The following script brings up a dialog box for input into ER Mapper. The default title is “batch engine input”. The ask begin/end block allows multiple inputs to be obtained from the user. The say statements create a single line of text in the dialog

box. The ask statements create an alphanumeric entry box. The example below opens a dialog box with the title ‘User Input Test’ and asks for an algorithm file name, some text, a number and a yes-or-no answer.

```
ask begin
    title "User Input Test"
    say "Please enter a file name"
    ask file "Algorithm:" ".alg" $file_name_input
    say "Please enter some text"
    ask text "Text:" $some_text_input
    say "Please enter a number"
    ask number "Number:" $some_number_input
    say "Please select Yes/No"
    ask yesno "YesNo:" $yesno_input
ask end
```

- The input dialog is opened when the “ask end” statement is read.
- There are a number of different types of field.
- The input variables can be arrays.
- To get multiple lines of text use multiple ask statements.

The above example will create the dialog box shown below:



Dialog box input fields

You can create the following types of input fields in dialog boxes Refer to “Script Commands - Alphabetical listing” for descriptions of the command.

Type	Description	Script command
Band chooser	Entry box with a button to open a chooser dialog listing the band descriptions for the specified file. You can also specify multiple or single choice.	ask bandchooser

Type	Description	Script command
Band menu	Drop down selection list of descriptions for all the bands in a specified file.	ask bandmenu
Color chooser	Alphanumeric entry box with a button to open a color chooser dialog	ask colorchooser
Datum chooser	Alphanumeric entry box with a chooser button to open the datum chooser dialog	ask datum
Directory chooser	Alphanumeric entry box with a chooser button for entering a directory name.	ask directory
Exclusive generic list	List of defined entries with exclusive radio buttons.	ask listmenu_exclusive
File	Alphanumeric entry box with a file chooser button for entering a file name.	ask file
Generic list	Selection list containing defined entries	ask listmenu
Grid layer	Drop down selection list of descriptions for all the gridding layers in a specified project file.	ask gridlayermenu
Hardcopy driver	Alphanumeric entry box with chooser button to open either the Windows printer driver or the ER Mapper hardcopy driver selection dialog box.	ask hardcopy
Link	Alphanumeric entry box with a link chooser button for entering a dynamic link name	ask link
Lookup table	Drop down selection list of the Color Lookup tables for selection.	ask lutmenu

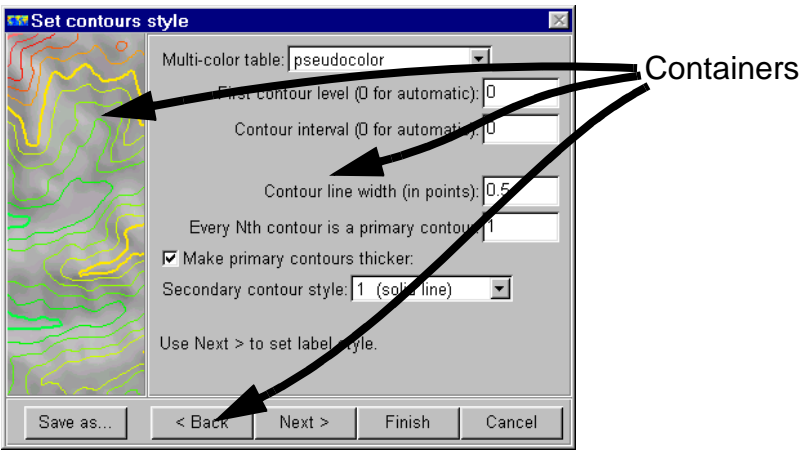
Type	Description	Script command
Navigation buttons	Button to select a defined action	ask action
Projection chooser	Alphanumeric entry box with a chooser button to open the projection chooser dialog.	ask projection
Text and number	Alphanumeric entry box	ask text number
Yes/No	Check box for selecting an option.	ask yes/no

Wizards

Wizards consist of a sequence of dialogs or pages through which a user is led. Each wizard page contains a number of containers. These wizard page containers have the same input fields as normal dialog boxes. In the scripting language, wizard pages and containers are defined by WizardPage begin..... WizardPage end, and container begin.... container end statements.

```
WizardPage begin "Wizard Title
.....
    container begin
        .....
    container end
    container begin
        .....
    container end
.....
WizardPage end
```

An example wizard dialog page is shown below:



The following commands are used to create wizards. Refer to“Script Commands - Alphabetical listing” on page 365 of the *Customizing ER Mapper* manual for descriptions of the commands

Function	Description	Script command
Wizard Page block	Starts a new wizard page	WizardPage begin end
Close wizard	Closes the wizard	Wizard close

Function	Description	Script command
Container block	Creates a container within a wizard page.	container begin end
Container button justification	Justifies button position	Container right left justify
Container position	Specifies how this container is to be placed in relation to another container	Container above below left right
Container item positions	Specifies the direction of placement of the items in a container	Container Items
Container label positions	Specifies where the labels for an item in the container will appear	Container Labels
Container size	Specifies the container width and height as a percentage of the remaining space.	Container width height
Display image	Displays image in container	show image

Using preferences to remember settings

ER Mapper maintains a list of preferences that can be retrieved at any time. Using this facility, you can design wizards to “remember” values entered by users so that they do not have to re-enter them when they run the wizard again.

For example: a wizard could ask the user to enter a background color which defaults to “white”. This value could be stored in a variable `$background_color`. The value in `$background_color` could then be stored as a preference. When the wizard is run again, it could retrieve the `$background_color` value from the preference and thus avoid having the user enter it again to change it from the default value of “white”.

To set a preference

In the above example you use the following command to set the preference:

```
set preference "Wizard:Image:BackGroundColor" $background_color
```

This command assigns the value in variable `$background_color` to a preference named “Wizard:Image:BackGroundColor”.

You can use any preference name, but it should be meaningful and non-ambiguous. ER Mapper wizards use the “Class:Name:Variable” format made up as follows:

Class: (e.g. Wizard)

Name: (e.g. Image)

Variable (e.g. BackGroundColor)

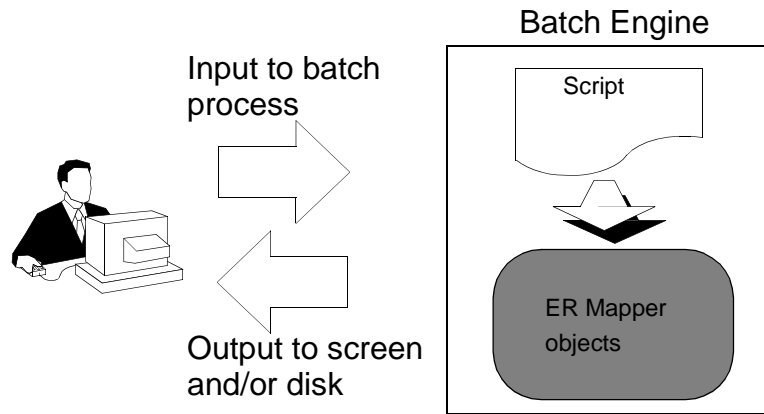
To retrieve a preference

In the above example you use the following command to retrieve the preference:

```
$background_color = get preference "Wizard:Image:BackgroundColor"  
"white"
```

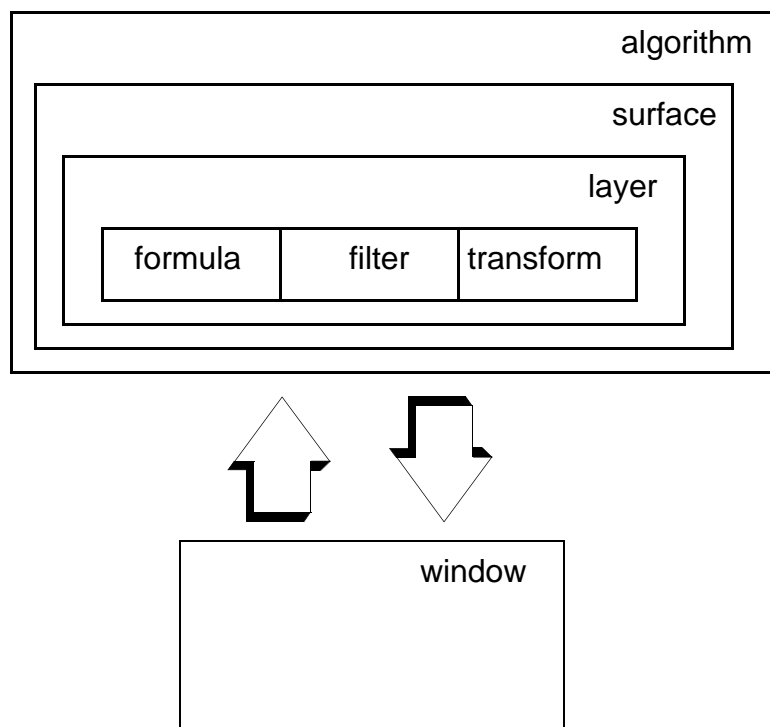
This command assigns the value in preference “Wizard:Image:BackgroundColor” to variable `$background_color`. It defaults to “white” if the preference has not been set. Refer to “Script Commands - Alphabetical listing” for descriptions of the command.

ER Mapper Objects.



ER Mapper comprises a number of objects which, in turn, contain attributes. Batch scripts instruct the batch engine to perform actions on these objects and their attributes. These actions can include creating new instances of an object or setting specific attributes pertaining to those objects.

The diagram below illustrates the objects and their relationships to one another:



An **algorithm** object will contain **surface** objects which, in turn, contain **layer** objects. The **layer** objects contain **formula**, **filter** and **transform** objects.

The **Window** object is used for displaying algorithms copied to it.

All the objects can exist on their own within the batch engine; e.g. you can have a defined layer object that is not contained within a surface or algorithm. It is also possible to have an empty window. However, you can only copy an algorithm (with its contained objects) to a window to display it.

You can assign an object to a variable (e.g. an algorithm can be assigned to \$alg), and then use this variable to define that object in your script. You can also set the batch engine to point to a specific object; i.e. make it current. Any batch commands that do not name a specific object will be performed on the current object.

For example,

```
select $alg[1] #makes $alg[1]the current algorithm
set algorithm mode to rgb
```

has the same effect as

```
set $alg[1] mode to rgb
```

Image Manipulation

When you write scripts to manipulate ER Mapper images and algorithms, the algorithms you construct and edit with the scripting language are within the batch engine—quite separate to ER Mapper itself.

Thus, if you create an algorithm using a script, to view it you need to copy it to ER Mapper.

Similarly, say you have an algorithm already being viewed and edited in ER Mapper. A script to change the Color mode from Pseudocolor to HSI must copy the algorithm from ER Mapper into the batch engine, make the appropriate color mode change, and then copy the resulting algorithm back to ER Mapper.

In the batch engine there are a number of reserved words which point to objects.

current window	indicates the current window. When a batch script starts, the current window is set to point to the currently active window in the GUI.
current algorithm	indicates the current algorithm within the batch engine
current surface	indicates the current surface within the algorithm
current layer	indicates the current layer within the surface
current formula	indicates the current formula within the layer
current transform	indicates the current transform within the layer
current filter	indicates the current filter within the layer
current input	indicates the current layer input within the layer

window, algorithm, surface, layer, transform, filter, and input usually refer to the current object. Some examples:

```
$win1 = current window
```

means set \$win1 to the current window.

```
copy window
```

means copy the current window, and update the current window pointer to refer to the new window.

```
$win = copy window
```

means copy the current window, update the current window to refer to the new window, and set \$win to refer to the new window.

```
select $win
```

means make the window defined by the variable \$win the current window.

To give you a feel for how ER Mapper and the batch engine interact with each other here are some examples to work through. These are all from the `ERMAPPER\batch` directory. Please look through the other script files for more extensive examples.

This first example copies the algorithm to the batch engine, the included file sets up the layers for the colordrape algorithm, then the lut and algorithm description are set. The changed algorithm must be copied back to the window before the user sees the result.

```
#from Create_CD.erb to create a colordrape algorithm
copy algorithm from window

#include code to create the colordrape layers
include "lib/Create_CD.erb"

set algorithm lut to "pseudocolor"
set algorithm description to "Colordrape"

copy algorithm to window
exit
```

This more extensive example is part of lib\Clip_99_All_Active_Layers.erb. It is included in the 'Go_Limits_99.erb' script. The algorithm has already been copied to the batch engine.

```
# Run the Algorithm at 100x100 resolution
#
go algorithm 100 100

# Cycle through all layers setting limits to actual data limits
#
first active raster layer
if ($ERROR != 0) then goto no_algorithm
next_active_layer:

last transform
set transform limits to actual
next active raster layer
if ($ERROR == 0) then goto next_active_layer

# Run the Algorithm at 100x100 resolution.
#
go algorithm 100 100

# Cycle through all layers setting the transform clip to 99.0%
#
first active raster layer
if ($ERROR != 0) then goto no_algorithm

next_active_layer:
last transform
set transform clip to 99.0
next active raster layer
if ($ERROR == 0) then goto next_active_layer

no_algorithm:
```

Actions

The batch engine controls the objects by performing actions on them. The following table lists these actions and shows their applicable objects. Refer to "Script Commands - Alphabetical listing" for descriptions of the command.

Command	Object types							Comment
	window	algorithm	surface	layer	transform	formula	filter	
add			*	*	*	*	*	Add object to a containing object.
copy	*	*	*	*	*	*	*	Duplicate object but do not insert into another object.
copy to copy from	*							Copy algorithm to and from window for display
current	*	*	*	*	*	*	*	Point to current object
delete	*	*	*	*	*	*	*	Delete object
duplicate			*	*	*	*	*	Make a duplicate of the object and insert it into the containing object
first, last, next, previous	*	*	*	*	*	*	*	Point to object and make it current
fit page		*						Fit algorithm page to hardcopy device.

Command	Object types							Comment
	window	algo- rithm	surface	layer	trans- form	form- ula	filter	
get		*	*	*	*	*	*	Get the value of the specified attribute. See Attributes below for a list of the attributes
go	*	*						Run the object
go background	*							Run object in background
load		*		*		*	*	Load algorithm, formula or filter into the batch engine.
move			*	*				Move to new position.
new	*	*	*	*	*	*	*	Create new object
save		*				*	*	Save object to file.
select	*	*	*	*	*	*	*	Make specified object current
set	*	*	*	*	*	*	*	Set the value of the specified attribute. See Attributes below for a list of the relevant attributes.
turn on turn off			*	*				Turn object on or off

Attributes

The objects have attributes associated with them. You can use the **set** and **get** commands to either set the attributes to required values or interrogate the object to return the values of specific attributes. The following table lists the attributes and shows which objects they are associated with. It also indicates whether you can perform sets and/or gets on them. Refer to "Script Commands - Alphabetical listing" for descriptions of the command.

Attribute	Object types						Comment
	algorithm	surface	layer	trans- form	formula	filter	
azimuth			set/ get				Sun shade azimuth
background	set						Specifies the background color as an rgb value or a color defined by a color variable.
cell sizex cell sizey			get				Cell size
cell type			get				Cell type
clip				set			Percentage clip
color			set/ get				Link layer color specified by rgb values or defined in a colorval variable.
contents	set/get						Page contents extents
contents extents	set						Set page contents extents to algorithm extents.
coordsys	set/get						Sets the coordinate system type.
dataset			set/ get				Layer image file name
datum	set/get						Sets the datum
description	set/get	set/get	set/ get			set/ get	Description text

Attribute	Object types						Comment
	algorithm	surface	layer	trans- form	formula	filter	
edit program			set/ get				Edit program name (dlink)
editable			set/ get				Editable flag true or false
elevation			set/ get				Sun shade elevation on or off
equalize				set			Set to histogram or gaussian equalize
formula			get/ set/ load/ save		load/ save		Sets the formula as a text value, or loads a formula (.frm) file.
init program			set/ get				Init program name
input count			get				Number of inputs
input filter count			get				Number of filter inputs
input to band					set		Formula input to band
input transform count			get				Number of input transforms
input output				set/get			Input/output limits
layer count	get	get					Number of layers
limits				set			Transform limits
link extension			set/ get				Link file extension
link type			set				Sets link type to mono- or truecolor
lut	set/get	set/get					Sets the color table for the specified surface or first surface in an algorithm.

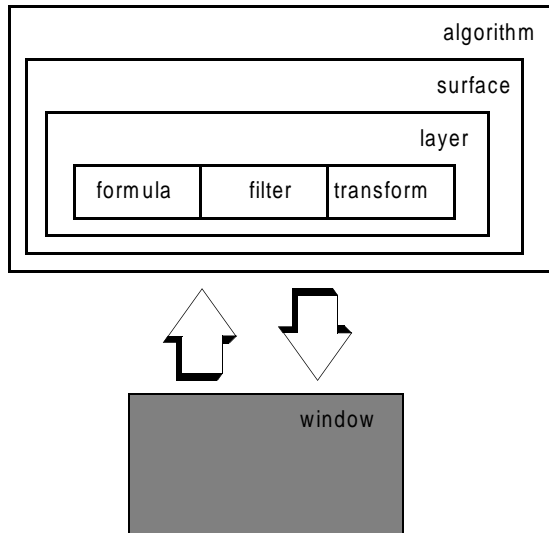
Attribute	Object types						Comment
	algorithm	surface	layer	trans- form	formula	filter	
matrix						set/ get	Element in filter matrix
mode	set/get	set/get					Sets the color mode for the specified surface or first surface in an algorithm.
mosaic type	set/get						Sets the mosaic type to overlay or feather
name		set					Name text
output filter count			get				Number of output filters
output transform count			get				Number of output transforms
page autovary_value	set						Calculate autovary value
page border	set/get						Page borders
page center	set						Center image on page; yes/no
page constraints	set/get						Page constraints
page extents	set						Page extents
page scale	set/get						Page scale
page size	set/get						Standard page size, e.g “US Letter”
page topleft page bottomright	set/get						Page extents coordinates
page width page height	set/get						Inside dimesions of page.
page view mode	set/get						View image only or with page layout.

Attribute	Object types						Comment
	algorithm	surface	layer	trans- form	formula	filter	
params						set/ get	Parameters
postsampled						set/ get	Postsampled process flag
projection	set/get						Image projection
rotation	set/get						Image rotation
rows cols						set/ get	Number of rows and columns.
scale						set/ get	Filter scale
shading			set/ get				Sets sunshading to on or off.
supersample type	set						Supersample type
threshold						set/ get	Filter threshold value
topleft bottomright	set/get						Sets the extents.
transparency		set					Surface transparency
type			set/ get	set/get		set/ get	Object type
units	set/get						Measurement units
userfile						set/ get	Source file name
userfunc						set/ get	Function name
view mode	set/get						View mode (2D/3D)
zoffset		set					Surface Z offset
zscale		set					Surface Z scale

Command summaries

The following sections summarize the commands applicable to the different objects. Refer to "Script Commands - Alphabetical listing" for descriptions of the command.

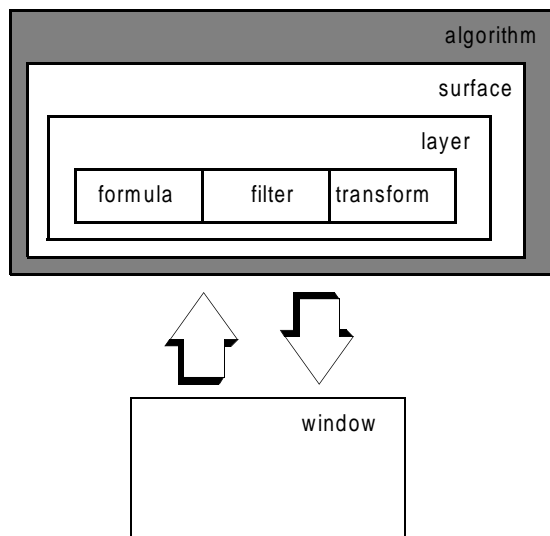
Windows



Description	Command
Copies the window reference from win1 to win2	\$win2 = \$win1
Copies the algorithm from the current or specified window to the batch engine and set the current algorithm pointer to point to it.	copy algorithm from window \$win
Copies the specified algorithm or that indicated by the current algorithm pointer to the current window in ER Mapper.	copy algorithm to window
Makes a duplicate of the current or specified window and its algorithm	copy window \$win
Sets pointer to the current window.	current window
Deletes the current or specified window and its algorithm.	delete window \$win

Description	Command
Updates the current window pointer to point to the oldest ,newest, next oldest or next newest window open.	first last next previous window
Runs the algorithm in the current or specified window as a background task	go background window \$win
Runs the algorithm in the current or specified window.	go window \$win
Opens a new image window, with a default algorithm	new window [x y w h]
Open the designated dialog box on screen	open window
Zoom in or out using specified option	previouszoom zoom in out zoom to
Updates the current window pointer to point to the given window.	select \$win
Sets the mouse pointer to the specified mode: zoom, zoombox, roam(hand) or pointer.	set pointer mode to zoom zoombox roam pointer
Various geolink functionality	set window geolink mode

Algorithms

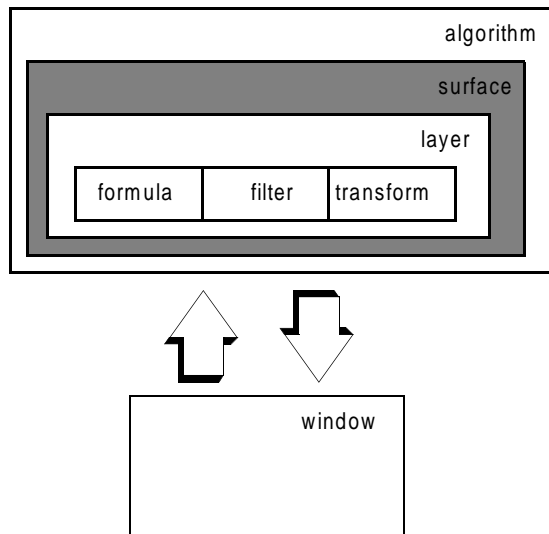


Description	Command
Sets \$alg2 to refer to \$alg1	\$alg2 = \$alg1
Copies the current or specified algorithm.	copy algorithm
Sets the pointer to the current algorithm	current algorithm
Deletes all references to an algorithm.	delete algorithm
Changes the current algorithm pointer to point to the first, last, next or previous algorithm.	first last next previous algorithm
Gets the current or specified algorithm's description.	get algorithm description
Gets the algorithm's coordinate system type.	get algorithm coordsys
Gets the algorithm's datum or projection type.	get algorithm datum projection
Gets the number of layers in the current or specified algorithm.	get algorithm layer count
Gets the algorithm's pseudocolor LUT name.	get algorithm lut
Gets the algorithm's mode.	get algorithm mode

Description	Command
Gets the algorithm's mosaic type.	get algorithm mosaic type
Gets the algorithm's extents.	get algorithm topleft bottomright eastings longitude meters_x
Gets the algorithm's units or rotation. Rotation is in decimal degrees, units is a units string.	get algorithm units rotation
Runs the current or specified algorithm.	go algorithm [width height][match]
Loads the given algorithm.	load algorithm \$name
Creates a new (empty) algorithm.	new algorithm
Saves the current or specified algorithm with the given name.	save algorithm \$name
Saves the current or specified algorithm as a virtual dataset	save algorithm as virtual dataset \$name
Saves the current or specified algorithm as a dataset	save algorithm as dataset
Changes the current algorithm pointer to point to the specified algorithm.	select \$alg
Changes the algorithm's background color to the given RGB values or to a value specified by a variable.	set algorithm background to \$red \$green \$blue set [algorithm] background to colorval \$colorvariable
Sets the algorithm's coordinate system type to the given type.	set algorithm coordsys to \$csys raw en ll
Sets the algorithm's datum or projection to the given type. This will cause layers of an incompatible type to be turned off within the algorithm.	set algorithm datum projection
Changes the current or specified algorithm description to the given text.	set algorithm description to \$text
Changes the pseudocolor LUT for the first surface in the algorithm to the given lut file.	set algorithm lut [to] \$lutname

Description	Command
Changes the processing mode for the first surface in the algorithm to the given mode.	set algorithm mode to \$mode pseudo rgb hsi
Sets the mosaic type (where different datasets overlay) to the given type	set algorithm mosaic type
Sets the current or specified algorithm supersample type.	set algorithm supersample type
Sets the current or specified algorithm topleft and bottomright extent fields in easting/northing, latitude/longitude or raw coordinate systems.	set algorithm topleft bottomright
Sets the units or rotation to the given value.	set algorithm rotation units to \$units

Surfaces

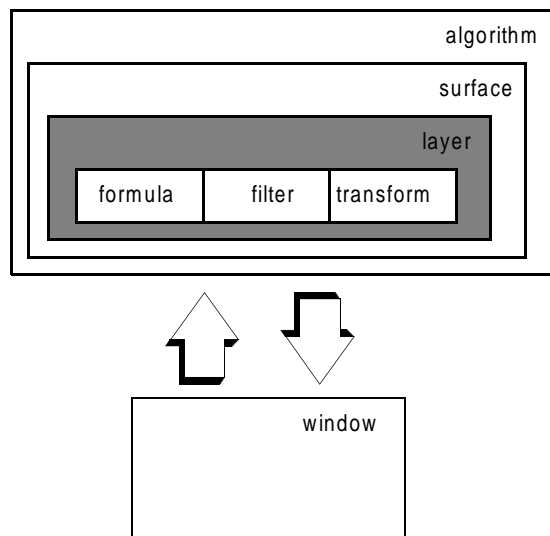


If an algorithm has more than one surface, they have to be individually defined and configured by these commands. This is not necessary if the algorithm only has one surface because the algorithm commands default to the top surface.

Description	Command
Adds the specified surface to the current algorithm.	add \$srf
Copies a the current or specified surface, but does not add the new surface to any algorithm	copy surface
Sets the pointer to the current surface.	current surface
Deletes the current or specified surface from the current algorithm.	delete surface
Duplicates the current or specified surface within the current algorithm.	duplicate surface
Selects a surface within the current algorithm and makes it current.	first last previous next surface
Gets the current or specified surface description.	get surface description
Gets the number of layers in the current or specified surface.	get surface layer count

Description	Command
Moves the current surface within the current algorithm.	move surface up down top bottom
Creates a new surface but does not add it to any algorithm.	new surface
Selects the given surface and makes it the current surface.	select \$srf
Changes the current or specified surface description to the given text.	set surface description to \$text
Sets the Color Table for the current surface.	set surface lut
Sets the color mode for the current surface.	set surface mode
Sets the current surface name to the given name.	set surface name
Sets the Z Scale, Z Offset or transparency of the current surface to \$value.	set surface zscale zoffset transparency
Checks whether current surface is active, and returns TRUE (1) for active and FALSE (0) for inactive.	surface active
Enables/Disables the current surface.	turn surface on off

Layers



All layer commands are related to the current surface. For example, an **add layer** command will add the layer to the current surface. If there is no designated current surface then it will default to the top surface within the current algorithm.

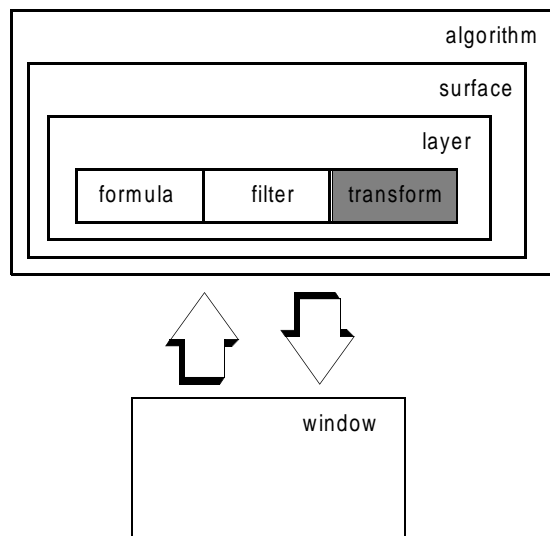
Description	Command
Assigns specified layer (e.g. \$lay1) to another variable (e.g. \$lay2)	\$lay2 = \$lay1
Adds the current or specified layer to the current surface, at the end of the layer list.	add layer
Adds a new layer of the given type after the current layer in the current surface.	add layer
Copies the current or specified layer but does not insert it into a surface.	copy layer
Sets the pointer to the current layer.	current layer
Deletes the current or specified layer. When all layers in a surface are deleted, the surface will also be deleted unless it is the only surface on the algorithm.	delete layer
Duplicates the current layer in the current surface.	duplicate layer

Description	Command
Sets the current layer pointer to point to the specified layer in the current surface.	first last next previous layer
Gets the current or specified layer color. Valid only on link layers.	get [layer] color
Gets the layers sun shading azimuth/elevation.	get layer azimuth elevation
Gets the number of bands in the current or specified layer.	get layer band count
Gets the nominated band description of the current or specified layer.	get layer band description
Reads the current or specified layer cell size.	get layer cell size x sizey
Gets the current or specified layer's image cell type.	get layer cell type
Gets the EN or LL coordinates from the given cellX and cellY values.	get layer x_coordinate y_coordinate from \$cellX \$cellY
Gets the current or specified layer's image name.	get layer dataset
Gets the current or specified layer's description.	get layer description
Gets the current or specified layer's editable flag. Valid only on link layers.	get layer editable
Gets the current or specified layer's edit/init program name.	get layer edit init program
Gets a copy of the current or specified layer's formula.	get layer formula
Gets the number of inputs in the current or specified layer.	get layer input count
Gets the number of input filters in the current or specified layer input.	get layer input filter count
Gets the number of input transforms in the current or specified layer input.	get layer input transform count

Description	Command
Gets the link file extension string for the current or specified layer.	get layer link extension
Gets the number of output filters in the current or specified layer.	get layer output filter count
Gets the number of output transforms in the current or specified layer.	get layer output transform count
Gets the current or specified layer's shading value.	get layer shading
Gets the current or specified layer's type.	get layer type
Checks whether the current layer is active, and returns TRUE (1) for active and FALSE (0) for inactive.	layer active
Loads the formula file into the current or specified layer's formula.	load layer formula
Moves the current or specified layer within the current surface to the given position within its surface.	move layer up down top bottom
Creates a new layer of the given type.	new layer
Moves the pointer to the next layer of the type specified.	next [active][raster vector] layer (type)
Saves the current or specified layer's formula to the formula file.	save layer formula
Sets the current layer pointer to point to the specified layer.	select \$lay
Sets the sun shade azimuth/elevation to the given value in degrees.	set layer azimuth elevation
Sets the current or specified layer color to the given RGB or color values. Valid only on link layers.	set layer color
Sets the image file name for the current or specified layer to \$dsname.	set layer dataset
Changes the layer description to the text given.	set layer description

Description	Command
Sets the editable flag for current or specified layer. Valid only for link layers.	set layer editable [to] true false
Sets the current or specified layer's edit/init program to the given name. Valid only on link layers.	set layer edit init program
Sets the current or specified layer formula.	set layer formula
Sets the designated layer input to the specified band number	set layer input \$n1 to band \$n2
Sets the link file extension of the current or specified layer to the given string.	set layer link extension
Sets the link type of the current or specified layer to monocolour or truecolour. Valid only on link layers.	set layer link type to monocolour truecolour
Turns sunshading on/off for the current or specified layer.	set layer shading on off
Changes the current or specified layer type to the given type.	set layer type
Turns the current or specified layer on or off.	turn layer on off

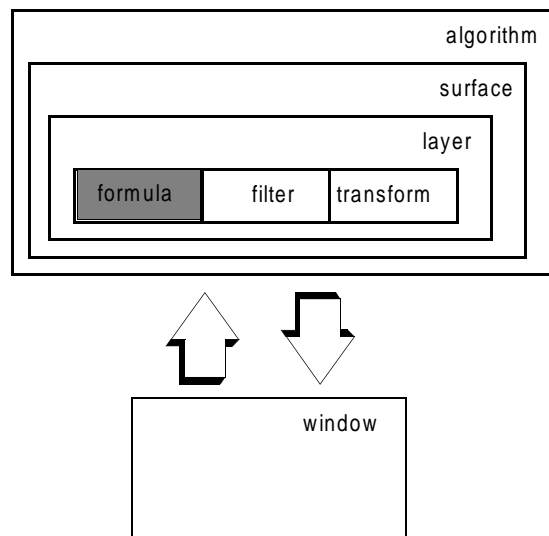
Transforms



Description	Command
Assign specified transform (e.g. \$tra1) to another variable (e.g. \$tra2).	\$tra2 = \$tra1
Adds the specified transform after the current transform in the current layer.	add \$tra
Adds the specified transform after the current transform in the given stream input of the current layer.	add \$tra to layer input
Adds a transform of the given type after the current transform.	add transform
Adds a point to the current or specified transform at x, y.	add transform point
Makes a duplicate of the current or specified transform but does not insert it into the current algorithm.	copy transform
Sets the pointer to the current transform.	current transform
Deletes the transform, and all references to it.	delete transform
Copies the current transform in the current layer, and inserts the copy after the current transform.	duplicate transform

Description	Command
Sets the current transform to the first, last, next or previous transform in the current layer. Optionally select an input transform, and/or a transform type.	first last next previous transform
Gets the current or specified transform's limits.	get transform input output min max
Gets the current or specified transform type.	get transform type
Matches the existing output transforms of all layers to the current or specified layer in the same surface.	match transform [to \$layer]
Creates a new transform which then becomes the current transform.	new transform
Sets the current transform to the specified transform.	select \$tra
Applies a clip of \$pct percent to the current or specified transform.	set transform clip
Sets the current or specified transform input/output limits.	set transform input output min max
Sets the current or specified transform limits to \$percent or actual.	set transform limits to actual \$percent
Sets the current or specified transform's output limits to the input limits.	set transform output limits to input limits
Applies a gaussian equalize operation to the current or specified transform.	set transform to gaussian equalize
Applies a histogram equalize operation to the current or specified transform.	set transform to histogram equalize
Sets the current or specified transform type.	set transform type

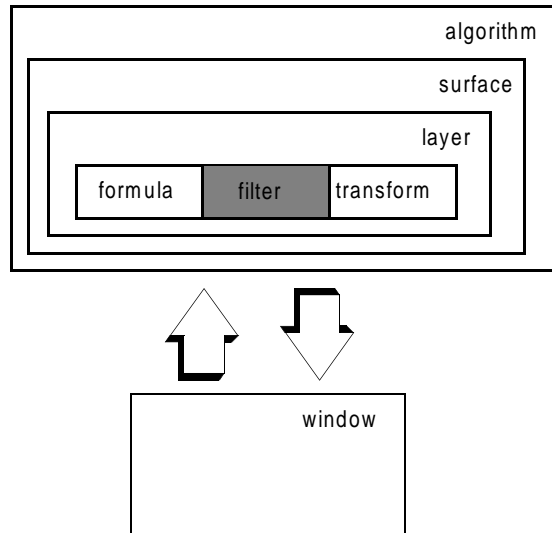
Formula



Description	Command
Adds the current or specified formula to the current layer.	add formula
Makes a copy of the current or specified formula, but does not add it to a layer.	copy formula
Sets the pointer to the current formula.	current formula
Deletes the current formula in the current layer.	delete formula
Duplicates the current formula and adds it to the current layer.	duplicate formula
Sets the current formula pointer to point to the specified formula in the current layer.	first last next previous formula
Loads the current or specified formula file.	load formula from \$fname
Creates a new formula.	new formula
Saves the current or specified formula to the formula file.	save formula to \$fname
Sets the current formula to the specified formula.	select \$for

Description	Command
Sets the current or specified formula (\$formula is a string).	set formula to \$formula
Sets the current formula input to the image band given.	set input \$n1 to band \$n2

Filters

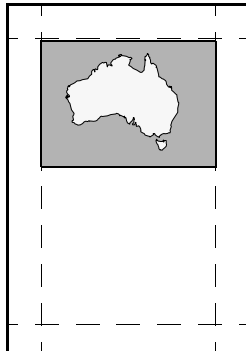


Description	Command
Assigns specified filter (e.g. \$fil1) to new variable (e.g. \$fil2)	\$fil2 = \$fil1
Adds the current or specified filter to the current layer after the current filter.	add filter
Makes a copy of the current or specified filter but does not insert the new filter into a layer.	copy filter
Sets the pointer to the current filter.	current filter
Deletes the filter and all references to it.	delete filter
Copies the current filter, and inserts the copy after the current filter.	duplicate filter
Sets the current filter to the first, last, next or previous filter in the current layer.	first last next previous filter
Gets the current or specified filter description.	get filter description
Gets an element from the current or specified filter matrix.	get filter matrix

Description	Command
Gets the current or specified user filter's parameters.	get filter params
Gets the current or specified filter's post sampled process flag.	get filter postsampled
Gets the number of rows/columns in the current or specified filter.	get filter rows cols
Gets the scale or threshold for the current or specified filter. Valid only on convolution and threshold filters.	get filter scale threshold
Gets the current or specified filter's type.	get filter type
Gets the current or specified user filter source file name.	get filter userfile
Gets the current or specified user filter function name. Valid only on usercode filters.	get filter userfunc
Loads the current or specified filter from the given file.	load filter
Creates a new filter, but does not add it to any layer.	new filter
Saves the current or specified filter to the given file.	save filter
Sets the current filter to the specified filter.	select \$fil
Changes the current or specified filter description to the given text.	set filter description to \$text
Sets an element of the current or specified filter matrix. Valid only on convolution and threshold filters.	set filter matrix
Set the current or specified user filter parameter string. Valid only on usercode filters.	set filter params to \$paramstring
Sets whether the current or specified filter can process resampled data, or source data.	set filter postsampled

Description	Command
Sets the number of rows/columns for the current or specified filter to \$number.	set filter rows cols
Sets the scale or threshold for the current or specified filter. Valid only on convolution and threshold filters.	set filter scale threshold
Sets the filter type of the current or specified filter.	set filter type
Sets the current or specified user filter source file, for a C usercode filter. Valid only on usercode filters.	set filter userfile [to] \$filename
Sets the current or specified user filter function name. Valid only on usercode filters.	set filter userfunc to \$funcname

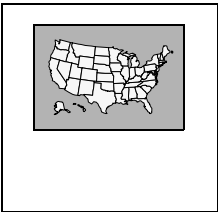
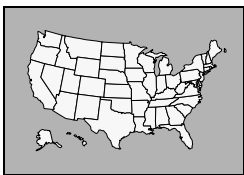
Page Setup



Description	Command
Sets the page width and height of the current or specified algorithm to that of the currently specified hardcopy device.	fit page to hardcopy
Gets the page contents extents coordinates.	get contents topleft bottomright
Gets the page borders of the current or specified algorithm.	get page top bottom left right border
Get the page constraints of the current or specified algorithm.	get page constraints
Gets the page scale of the current or specified algorithm.	get page scale
Gets the page size of the current or specified algorithm.	get page size
Gets the page extents coordinates.	get page topleft bottomright
Gets the specified page parameter of the the current or specified algorithm.	get page width height
Calculates and sets the bottom right contents extents coordinate, based on the topleft coordinate, page size, borders, and scale.	set contents bottomright from topleft
Sets the page contents extents coordinates to that specified by \$value.	set contents topleft bottomright

Description	Command
Calculates one of the page setup variables based on the Constraints setting.	set page autovary_value
Centers the contents of the page horizontally or vertically.	set page center horizontal vertical
Sets the page constraints of the current or specified algorithm.	set page constraints to \$constr zoom page border scale
Calculates and sets the page extents of the algorithm based on the contents extents, the borders, and the scale.	set page extents from contents
Sets the page scale to a specific number or the maximum scale possible.	set page scale to \$scale max
Sets the page size of the current or specified algorithm.	set page size to \$size
Sets the page extents coordinates to that specified in the variable \$value.	set page topleft bottomright
Sets the page borders.	set page top bottom left right border
Sets the specified page size parameter of the current or specified algorithm.	set page width height
Set the contents extents to the algorithm extents (where algorithm is the current algorithm).	set contents extents to [algorithm] extents

Page view mode



Description	Command
Gets the page view mode of the current or specified algorithm.	get page view mode
Sets the page view mode of the current or specified algorithm to normal or layout.	set page view mode to \$pvmode normal layout

View mode



Description	Command
Gets the view mode of the current or specified algorithm.	get view mode
Sets the view mode of the current or specified algorithm to 2D, 3D perspective or 3D flythru.	set view mode to \$vmode 2d perspective flythru

Preferences

Preferences enable batch scripts to retain parameters when they are shut down and re-run. There are a number of preferences already defined in ER Mapper, but you can define and set new ones.

The following commands write to the user's preference file.

Refer to "Scripting Commands - Alphabetical listing" for descriptions of these commands.

Function	Description	Script command
Set preference entry	Adds or updates the named preference entry	set preference
Get preference entry	Returns the value in the named preference entry. Can return a default value if the entry does not exist.	get preference
Set color preference entry	Adds or updates the named color preference entry	set preference
Get color preference entry	Returns the value in the named color preference entry. Can return a default value if the entry does not exist.	get preference

File and Path separators

Function	Description	Script command
Build an absolute file specification	Builds the absolute file specification from a given relative file specification and its parent directory	build absolute filespec <parent_dir> <rel_file>
Build a file path name	Builds a complete path for a file from up to 4 given elements.	build file path <element_1> <element_2><element 4>
Build a relative file specification	Builds file specification relative to its parent directory from a given absolute file specification.	build relative filespec <parent_dir> <abs_file>
Convert file separators	Converts the file separators in a given file specification from English to native and vice versa.	convert <filespec> to english native sep separator
Get English file separator	Returns the standard English file separator used by the host workstation or PC. This would be “\” on a Win32 (PC) platform or “/” on a Unix platform.	get english file sep separator
Get native path file separator	Returns the path separator used natively by the host workstation or PC.	get native path file sep separator

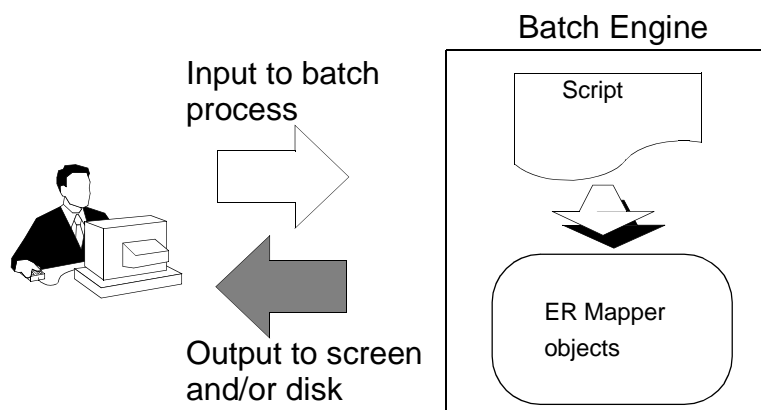
Other commands

There are a number of commands that control the batch processing environment. These are listed below:

Function	Description	Script command
get file size	Gets the given file's size in bytes.	get \$filename size
get free space on disk	Gets the amount of free space, in bytes, on the given disk.	get \$diskname free space
get environment variable	Gets the given environment variable.	getenv \$envname
set environment variable	Sets the given environment variable.	setenv \$envname
delete directory or file	Deletes a directory or file. If the file name has an ".ers" extension, the raster file is also deleted.	delete \$dir \$file
list contents of directory	Lists the contents of the specified directory into an array.	listdir
edit text file	Edits the given file name in a text editor. The file is created if it doesn't exist.	edit \$filename
exit batch process	Exits the batch process and returns you to the ER Mapper main window. See also "Command line arguments" on page 4	exit
file exists	Verifies the existence of the specified file and returns true or false.	if \$filename exists
split string into array	Splits the given string into an array of substrings at the given token	split \$string at \$token

Function	Description	Script command
format number of digits after decimal point	Formats a number to have a specified number of digits after the decimal point .	format \$input_value \$precision
get ER Mapper version number	Returns the ER Mapper version number as a string or as a number.	get version_string version_number
execute system command	Executes a system command	system \$command
system command status dialog	Similar to the system syntax, except that a progress dialog is created.	system \$command status ["title"]
File name, extension and path keywords	Returns the file name, path and extension of a given file specification	filename \$string dirname \$string fileext \$string
Color keywords	You can use the color_red , color_green and color_blue keywords to get the rgb (0-255) components of a color variable.	\$bg_color color_red \$bg_color color_green \$bg_color color_blue

Output.



The results of a batch process can be directed to an image window, to a Batch Engine Dialog box, or to be saved as a file.

Output to image window

To view an algorithm on the screen, you have to copy it to a window object. The following commands load an algorithm into the current algorithm object, and then copy it to the current window.

```
load algorithm test_algorithm.alg
copy algorithm to window
```

Output to file

You can copy instances of the following object types to a file using the ‘save’ command:

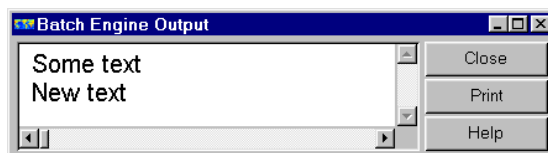
- algorithm
- formula
- filter

Examples of the save commands are given below:

```
save algorithm my_alg.alg
save algorithm as dataset my_image.ers
save algorithm as virtual dataset my_image_vds.ers
save layer formula my_form.frm
save formula my_form.frm
save filter my_filter.ker
```

Output to Batch Engine Output dialog (print commands)

The Batch Engine Output dialog opens automatically when you use ‘print’ commands. New print commands append text to that what exists in already open dialogs.



There are two print commands:

Note:	print	no CR/LF
	println	CR/LF

By default, numbers are printed to 6 decimal places. Use the ~ to specify a different number of decimals.

statement	result
<code>print 1</code>	1.000000
<code>\$var = 2</code> <code>print \$var</code>	2.000000
<code>\$text = "Hello World"</code> <code>print \$text</code>	Hello World
<code>\$text = "Hello World"</code> <code>print \$text</code> <code>println \$text</code>	Hello WorldHello World
<code>print ~3 1</code>	1.000

The `print` and `println` commands write to an output dialog by default. Alternatively, you can print out to a file using:

```
set output to $filename
```

The `print` or `println` commands in a batch script after this `set` command will create the file if it doesn't already exist and write the output to the end of the file.

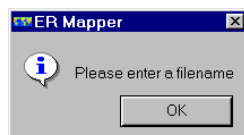
To reset the output to write to a dialog use the following command:

```
set output to output window.
```

Warning dialog

You can use the 'say warning' command to open an ER Mapper warning dialog box with a specified message. The following is an example of its use:>

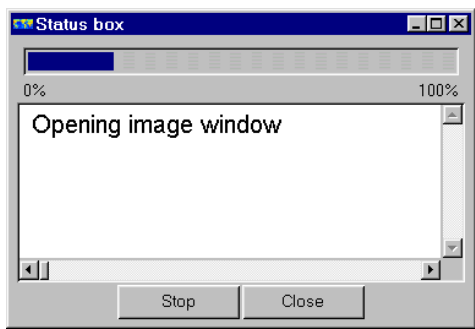
```
checkdataset:
if ($filename!= "") then goto DatasetOK
    say warning "Please enter a filename"
    goto wizard_page_1
DatasetOK:
```



Status dialog

You can create a status dialog box which indicates the progress, as a percentage and/or as text, of any process the user has invoked. This is usually used in wizards

```
...
ask action "Status" goto OpenStatus
...
OpenStatus:
open status
$percent = 20
say status $percent "Opening image window\n"
....
goto wizard_page_1
```



The following commands create and write to a status or warning dialog box.

Note: There is only one status dialog. If it is already open, new messages will be added to it.

Refer to “Script Commands - Alphabetical listing” on page 365 of the *Customizing ER Mapper* manual for descriptions of the commands.

Script command	Description
open status	Opens status dialog
say status	Adds message to status dialog
delete status	Deletes status dialog
say warning	Opens warning dialog with a message.

Library of batch scripts

Common batch script functions are stored in %ERMAPPER%\batch\lib. These can be included in your scripts. For example, if you include lib\BE_Startup.erb, you get \$machine_type, \$ERMAPPER, \$ERMBIN and \$ERMSCRIPTS and a number of other variables defined for you.

Browse through the directory to see what is available. Some of the scripts include others, so study them to see how to include the code in your script.

Creating an image Tiff file

This chapter explains how to create image tiff files using ER Mapper Hardcopy Control Files option. Image tiff files are used as pictures on toolbar and wizard buttons.

About image tiff files

Image tiff files are raster images saved as tiff file format. Image tiff files can be processed using ER Mapper. The image tiff file must be placed in the 'ERMAPPER/icons' directory and must be a 24-bit 16x16 pixel tiff file to be used as pictures on toolbar and wizard buttons. When incorporating it in the configuration toolbar file the tiff file must be enclosed in two sets of quotation marks but without the .tif extension. For example, "New" refers to the 'New.tif' file in the 'ERMAPPER/icons' directory.

Tip: You can use an exiting Tiff file from the 'ERMAPPER/icon' directory. Image tiff files of 24-bit 64x64 pixels or larger sizes are sometimes used with wizard pages. These larger size image tiff files can also be created using the Hardcopy Control Files option in ER Mapper.

Hands-on exercises

These exercises teach you how to create tiff files from processed images using the Hardcopy Control Files option in ER Mapper.

What you will learn...

After completing this exercise, you will know how to create tiff files from processed images using the Hardcopy Control Files option in ER Mapper

- Load a RGB composite
- Select the Hardcopy Control Files option and set up the necessary parameters
- Save the tiff file to a given file name and it will be saved in the /TEMP directory
- Move or copy the tiff file to the 'ERMAPPER/icons' directory


Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

Creating a TIFF_16x16_24bit file

Create a RGB composite TM7(Red), TM4(Green) and TM1(Blue) using "Landsat_TM_year_1985.ers" dataset. (

Note: The Landsat_TM_year_1985.ers dataset is in the 'ERMAPPER\dataset\Core_Datasets' directory.


Tip: Click the **Open Algorithm into Image Window**  button on the main menu. The file chooser **Open** dialog box appears. In the file chooser **Open** dialog box set the directory to 'ERMAPPERalgorithm\Data_Examples\Landsat_TM'. Double click on the 'RGB_741.alg' from the directory. The RGB_741 color composite is displayed.

- 1 Click the **Print**  button on the main menu. The **Print** dialog box appears.

Note: The temporary algorithm for the RGB_741.alg has already been loaded for you in the **Algorithm** text field.

- 2 Select the **Hardcopy Control Files** option at the bottom part of the **Print** dialog box.

Note: For UNIX platform only the **Hardcopy Control Files** option is available.

- 3 On the **Print** dialog box click the load file button  of the **Output Name:** and the **Open Hardcopy Control** dialog box appears.
- 4 On the **Open Hardcopy Control** dialog box set the directory to 'ERMAPPER\hardcopy\Graphics'.
- 5 Double click and load the 'TIFF_16x16_24bit.hc' file into the **Output Name:** text field.
- 6 Click the **Setup** button on the **Print** dialog box.
- 7 The **Setup** dialog box appears.
- 8 In the **Filter Program** text field, type in the name of the Tiff file you want the RGB_741 image to be saved. Example: \$ERMTMP/JS_RGB_741_16x16.tif.
- 9 After you have typed in the Tiff file name **OK** the **Setup** dialog box.


Tip: You can select the options in the **Setup** dialog box such as adjusting the Gamma Correction, Force 1 Dataset Cell = 1 Image Pixel etc.


- 10 On the **Print** dialog box select **Fit page to output device**.
- 11 Click the **Print** button on the **Print** dialog box.
- 12 The RGB_741 image is saved as JS_RGB_741_16x16.tif file in the 'WINDOWS\TEMP' directory.
- 13 After the "Hardcopy finished successfully" message is displayed, copy or move the 'JS_RGB_741_16x16.tif' file to the 'ERMAPPER/icon' directory.

Note: You have created the 'JS_RGB_741_16x16.tif' image tiff file and placed it in the 'ERMAPPER/icon' directory. You will use the tiff file as the picture in the icon for one of the toolbar buttons.


Creating a TIFF_64x64_24bit file

Follow the above mentioned procedures and create a RGB composite TM4(Red), TM3(Green) and TM2(Blue) using “Landsat_TM_year_1985.ers” dataset which is in the ‘ERMAPPER\dataset\Core_Datasets’ directory.

Tip: Click the **Open Algorithm into Image Window**  button on the main menu. The file chooser **Open** dialog box appears. In the file chooser **Open** dialog box set the directory to ‘ERMAPPER\algorithm\Data_Examples\Landsat_TM’. Double click on the RGB_432.alg from the directory. The RGB_432 color composite is displayed.

- 1 Click the **Print**  button on the main menu. The **Print** dialog box appears.

Note: The temporary algorithm for the RGB_432.alg is loaded for you in the **Algorithm** text field.

- 2 Select the **Hardcopy Control Files** option at the bottom part of the **Print** dialog box.
- 3 On the **Print** dialog box click the load file button  of the **Output Name:** and the **Open Hardcopy Control** dialog box appears.
- 4 On the **Open Hardcopy Control** dialog box set the directory to ‘ERMAPPER\hardcopy\Graphics’.
- 5 Double click and load the ‘TIFF_64x64_24bit.hc’ file into the **Output Name:** text field.
- 6 Click the **Setup** button on the **Print** dialog box.
- 7 The **Setup** dialog box appears.
- 8 In the **Filter Program** text field, type in the name of the Tiff file you want the RGB_432 image to be saved. Example: ‘\$ERMTMP/JS_RGB_432_64x64.tif’. After you have typed in the Tiff file name **OK** the **Setup** dialog box.

Note: \$ERMTMP is the system variable of temporary directory.

Tip: You can select the options in the **Setup** dialog box such as adjusting the Gamma Correction, Force 1 Dataset Cell = 1 Image Pixel etc.

- 9 On the **Print** dialog box select **Fit page to output device**
- 10 Click the **Print** button on the **Print** dialog box.
- 11 The RGB_432 image is printed to the 'JS_RGB_432_64x64.tif' tiff file and the tiff file is saved in the WINDOWS\TEMP directory.
- 12 After the "Hardcopy finished successfully" message is displayed, copy or move the 'JS_RGB_432_64x64.tif' file to the 'ERMAPPER/icon/standard_icons' directory.

Note: Image tiff files used in wizard pages are placed in the 'ERMAPPER/icons/standard_icons' directory.

Note: You have created an image tiff file and placed it in the 'ERMAPPER/icon/standard_icons' directory. You will use it as the picture in one of the wizard pages.

Exercises:

- 1 Create a densitysliced image tiff file of 16x16 pixels size using the Hardcopy Control File option of ER Mapper. Place the image tiff file you have created in the 'ERMAPPER/icons' directory.
- 2 Create a RGB composite image tiff file of 64x64 pixels size using the Hardcopy Control File option of ER Mapper. Place the image tiff file you have created in the 'ERMAPPER/icons/standard/icons' directory.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:
 - For Windows, select **Close** from the window control-menu.
 - For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).
- 2 Click **Close** on the **Algorithm** window to close it.

Chapter 2 Creating an image Tiff file ● Exercises:

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to perform the following tasks in ER Mapper:

- Load a RGB composite from an existing algorithm
- Create tiff files using hardcopy control files
- Save the processed images as tiff files in given names
- Place the 16x16 pixel tiff file in the 'ERMAPPER/icons' directory to be used as a picture on a toolbar or wizard button and the 64x64 pixel tiff file in the 'ERMAPPER/icons/standard/icons' directory to be used in wizard pages.

Creating a toolbar file

This chapter explains how to create a toolbar ‘*.bar’ file which is necessary for your toolbar or wizard button/s to be displayed on the toolbar of the ER Mapper main menu.

About toolbar files

A toolbar ‘*.bar’ file is a configuration file and is placed in the ‘ERMAPPER/config’ directory. The toolbar file will automatically list it in the ER Mapper toolbar menu in alphabetical order.

An example of a toolbar file is shown below:

```
#####

# Copyright 1990/91 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# FILE:      config/standard.bar
# CREATED:Thu Feb  3 13:39:03 WST 1994
# AUTHOR: Andrew Hunt
```


Chapter 3 Creating a toolbar file ● About toolbar files

```
# PURPOSE:ER Mapper standard toolbar config file.
#####
"New"          "New Image Window" CALLBACK new_alg_cb
"Open"         "Open Algorithm into Image Window" CALLBACK load_alg_cb
"Copy_Window"  "Copy Window and Algorithm"BATCH "Copy_Window"
"Save"         "Save Algorithm"          CALLBACK save_alg_cb
"Save_As"      "Save Algorithm As"        CALLBACK save_alg_as_cb
"Save_As_DS"   "Save Algorithm As Dataset" CALLBACK save_alg_as_ds_cb
"Save_As_VDS"  "Save Algorithm As Virtual Dataset" CALLBACK save_alg_as_vds_cb
"-----" " " " "
"Printer"      "Print"                   CALLBACK open_hardcopy_cb
"-----" " " " "
"Go"           "Run Algorithm (GO)" CALLBACK go_cb
"Go_Limits"    "Run Algorithm (GO) with 99% clip on limits" BATCH "Go_Limits_99"
"Stop"         "Halt Processing (STOP)" CALLBACK stop_cb
```

Note: In the first line, for the first toolbar button, **Field 1** "New" is the name of the Tiff file which refers to 'New.tif' file in the 'ERMAPPEER/icon' directory. **Field 2** "New Image Window" is the Tool Tips or the explanation of the function of the toolbar button that appear just below the cursor when you place the cursor on the toolbar button. **Field 3** CALLBACK is the ER Mapper Function name that refers to the function name "new_alg_cb". The third toolbar button **Field 3** BATCH is the ER Mapper Function name that refers to the batch script file "Copy_Window" (Copy_Window.erb) in the ERMAPPER/batch directory. "-----" " " " " are blank spaces to separate toolbar buttons.

Tip: You can use the template "Skeleton.bar" file from the 'ERMAPPER\batch\template' directory to create your toolbar file.

Tip: You can run a batch script without creating a toolbar file. From the ER Mapper main menu window you can run a batch script directly through **View/ Batch Engine Script Control** menu. Click the **View** menu and select the **Batch Engine Script Control**. The **Batch Engine Script Control** dialog box appears. On the **Batch Engine Script Control** dialog box click the **Run Script** button. **Run a Batch Script** dialog box appears. From the 'ERMAPPER/batch' directory select the batch script "*.erb" file you want to run. This option is handy especially when you are test running your batch script. .

What you will learn...

After completing these exercises, you will know how to create a toolbar file

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

Hands-on exercise

1:Creating a toolbar file using a CALLBACK function

This exercise shows you how to create a simple toolbar file using a CALLBACK function and runs the function from the toolbar button.

Tip: You can use the template "Skeleton.bar" file from the ERMAPPER\batch\template directory. The file can be edited using your favourite Text (ASCII) editor or the **Notepad** text editor accessed through ER Mapper under the **Utilities/User Menu/Edit a File** option.

- 1 Load the "Skeleton.bar" file from the 'ERMAPPER\batch\template' directory into a Text Editor.

```
#####
```

```
# Copyright 1990/91 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
```

Chapter 3 Creating a toolbar file ● 1:Creating a toolbar file using a CALLBACK function

```
# code contained herein.
#
# FILE:      config/XXXX.bar
# CREATED:XXXX
# AUTHOR: XXXX
# PURPOSE:XXXX
#####
"Icon_name"      "Example CALLBACK toolbar item"CALLBACK callback_fn
"Icon_name"      "Example BATCH toolbar item"BATCH "Script_Name"
"-----" "Example separator""""Icon_name""Example BATCH toolbar item"
BATCH "Script_Name"
"-----" "Example separator""""
#####
```

2 Edit the file as below: (Text added are in bold letters)

```
#####
# Copyright 1990/91 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# FILE:      config/test1.bar
#CREATED: 4th August, 1997
# AUTHOR: John Smith
# PURPOSE:Testing CALLBACK function annotate_cb
#####
"JS_RGB_741_16x16"      "Example CALLBACK annotate_cb" CALLBACK annotate_cb
#####
```

3 save the toolbar file as “test1.bar” in the ‘ERMAPPER/config’ directory

Note: You have placed the “JS_RGB_741_16x16.tif” file to be the picture of the icon of the annotate_cb function toolbar button in your “test1.bar” toolbar file

- 4 Exit the ER Mapper and call it again so that it will recognize your configuration toolbar file in the ‘ERMAPPER/config’ directory
- 5 On the ER Mapper main menu click on the toolbar menu. Your test1.bar appears as “test1” in the Toolbar menu in alphabetical order.

Tip: You can only call one function at a time. You cannot call two functions with one CALLBACK command. To call two or more ER Mapper internal functions you must call them separately. For example, ("JS_RGB_741_16x16" "Example CALLBACK annotate_cb" CALLBACK annotate_cb) and ("JS_RGB_432_16x16" "Example CALLBACK transform_open_cb" CALLBACK transform_open_cb).

- 6 Click the **test1** on the toolbar menu
- 7 A button for the annotate_cb function is displayed on the ER Mapper main menu.
- 8 Place your cursor on the annotate_cb function button. The " Example CALLBACK annotate_cb" text appears in the small text window just below the cursor.
- 9 Click on the annotate_cb function button. An image display window and the **New Map Composition** window appear.

Available callback options

The following callback options are available:

- geoposition_cb
- annotate_cb
- zoom_buttons_cb
- profile_cb
- position_cb
- dig_config_cb
- dig_session_cb
- gcp_cb
- extract_traverse_cb
- scatter_cb
- edit_alg_cb
- new_alg_cb
- load_alg_cb
- load_alg_from_vds_cb
- dataset_load_cb

- transform_open_cb
- ilter_open_cb
- formula_open_cb
- sunangle_open_cb
- save_alg_cb
- save_alg_as_cb
- save_alg_as_ds_cb
- save_alg_as_vds_cb
- delete_alg_cb
- proc_info_cb
- preferences_cb
- batch_gui_cb
- batch_job_cb
- exit_cb
- open_hardcopy_cb
- go_cb
- stop_cb
- open_window_cb
- notyet_cb
- pagesize_cb
- set_mode_zoom_cb
- set_mode_not_zoom_cb

The following are parametes accepted by some callbacks.

- 1
- 2
- 3
- 4
- 5
- ZOOM_BUTTON_PREVIOUS
- ZOOM_BUTTON_ZOOM_TO_WINDOW
- ZOOM_BUTTON_ALLDS

- ZOOM_BUTTON_CURRENTDS
- ZOOM_BUTTON_ALLVECDS
- ZOOM_BUTTON_ALLRASTDs
- ZOOM_BUTTON_CONTENTS
- ZOOM_BUTTON_PAGE
- ZOOM_BUTTON_ZOOMIN
- ZOOM_BUTTON_ZOOMOUT
- ZOOM_BUTTON_PAN_LEFT
- ZOOM_BUTTON_PAN_RIGHT
- ZOOM_BUTTON_PAN_UP
- ZOOM_BUTTON_PAN_DOWN
- ZOOM_BUTTON_GEOLINK_NONE
- ZOOM_BUTTON_GEOLINK_WINDOW
- ZOOM_BUTTON_GEOLINK_SCREEN
- ZOOM_BUTTON_GEOLINK_OVERVIEW_ZOOM
- ZOOM_BUTTON_GEOLINK_OVERVIEW_ROAM
- ZOOM_BUTTON_ZOOMIN_100
- ZOOM_BUTTON_ZOOMIN_200
- ZOOM_BUTTON_ZOOMOUT_50
- ZOOM_BUTTON_ZOOMOUT_100
- ZOOM_BUTTON_ZOOMOUT_200

2:Add an existing batch script file to the toolbar file

This exercise shows you how to add an existing batch script file to a toolbar file.

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

- 1 Load the “test1.bar” toolbar file which you have created into a text editor.
- 2 Edit the “test1.bar” file as follows: (Text added are in bold letters)

#####

Copyright 1990/91 Earth Resource Mapping Pty Ltd.

Chapter 3 Creating a toolbar file ● 2: Add an existing batch script file to the toolbar file

```
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# FILE:      config/test1.bar
# CREATED: 4th August, 1997
# AUTHOR: John Smith
# PURPOSE: Add a batch script to a toolbar file
#####

"JS_RGB_741_16x16""Example CALLBACK annotate_cb" CALLBACK
annotate_cb

"JS_RGB_432_16x16""Example BATCH RGB composite" BATCH "
Create_RGB"

#####
```

Note: You have placed the “JS_RGB_432_16x16.tif” tiff file as the picture of the icon for the Create_RGB batch script file toolbar button. The Create_RGB batch script toolbar button will appear together with the annotate_cb toolbar button on the “Test1” toolbar .

- 3 Exit the ER Mapper and call it again so that it will recognize your configuration toolbar file which is in the ‘ERMAPPER/config’ directory
- 4 On the ER Mapper main menu click on the **Toolbar** menu. Your test1.bar appears as “test1” in the **Toolbar** menu in alphabetical order.
- 5 Click the **test1** on the toolbar menu
- 6 Two toolbar buttons appears on the ER Mapper main menu. The first button is for the annotate_cb function and the second button is for the Create_RGB batch script.
- 7 Place your cursor on the Create_RGB button. The” Example RGB Composite” text appears in the small text window just below the cursor.
- 8 Click on the Create_RGB batch script button (the second button). An image window and the **Select a Dataset** dialog box will appear.
- 9 On the **Select a Dataset** dialog box, from the ‘ERMAPPER/dataset/Core_datasets’ directory double click on the ‘Landsat_TM_1985.ers’ dataset.
- 10 **OK** the **Select a Dataset** dialog box.
- 11 The Landsat TM321 RGB composite is displayed in the image window.

Exercises:

- 1 Create a toolbar file and include an internal function. Use one of the image tiff files (16x16 pixels size) you have created previously and call an internal function using CALLBACK command. Run the function from the toolbar button you have created.
- 2 Add a batch script file to the toolbar file you have created in exercise one. Use one of the image tiff files (16x16 pixels size) you have created previously and call an exiting batch script using BATCH command. Run the script from the toolbar button you have created.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:
 - For Windows, select **Close** from the window control-menu.
 - For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).
- 2 Click **Close** on the **Algorithm** window to close it.

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to perform the following tasks in ER Mapper:

- Create a toolbar file and place it in the 'ERMAPPER/config' directory.
- Create an icon for a toolbar button using an image tiff file.
- Call an ER Mapper internal function using CALLBACK command.
- Create a toolbar button for a batch script and run the batch script from the toolbar button.

4

Variables and input

This chapter shows you how to define number and string variables and manipulate them in batch scripts. It also explains how to select various types of files.

About variables in batch scripts

Variables have a \$ leading character, followed by a letter and then alphanumeric characters (including underscores). Variable names can be of any length. The only limitation is the memory allocation. Once a variable has been defined, its type becomes fixed. For examples, \$var_number = 1 becomes a number variable whereas \$variable_text = “Hello” becomes a string variable. Numbers assigned to variables are taken as real numbers with six decimal places as default.

Hands-on exercises

These exercises teach you how to define variables in batch scripts and selecting various types of files

What you will learn...

After completing this exercise, you will know how to define variables and manipulate them. You will also learn how to select raster data files and various types of vector files.

- Define number and string variables
- Select raster files
- Select various types of vector files

- Print the variables and file names in the Batch Engine Output dialog box

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

1:Creating a batch script with variables

Note: To run a batch script from a toolbar or wizard button, the batch script file has to be included in the toolbar file. Hence, add the name of the batch script you are going to create in the “test1.bar” toolbar file.

- 1 Edit the “test1.bar” file and add a new line “**JS_RGB_531_16x16**”
“**Example BATCH var_math**” BATCH “var_math”

```
#####  
  
# Copyright 1990/91 Earth Resource Mapping Pty Ltd.  
# This document contains unpublished source code of  
# Earth Resource Mapping Pty Ltd. This notice does  
# not indicate any intention to publish the source  
# code contained herein.  
#  
# FILE:    config/test1.bar  
# CREATED: 4th August, 1997  
# AUTHOR:  John Smith  
# PURPOSE: Testing a CALLBACK function and a batch script  
#####  
  
"JS_RGB_741_16x16"      "Example CALLBACK annotate_cb" CALLBACK annotate_cb  
"JS_RGB_531_16x16" "Example BATCH var_math" BATCH "var_math"  
#####
```


Note: You have added a simple “var_math” batch script file which will refer to the “var_math.erb” batch script file in the ‘ERMAPPER/batch’ directory. You will create the “var_math.erb” batch script file below.

Create an image Tiff file to be used as the picture in the icon of the toolbar button

Process an image that you would like to use it as the picture in the icon of your batch script toolbar button.

Tip: You can use an exiting Tiff file from the 'ERMAPPER\icon' directory

- 1 Use "Landsat_TM_year_1985.ers" dataset from the 'ERMAPPER\dataset\Core_Datasets directory' and create a (TM5(Red), TM3(Green) and TM1(Blue)) RGB composite image.

Tip: Click the **Open Algorithm into Image Window**  button on the main menu. The file chooser **Open** dialog box appears. In the file chooser **Open** dialog box set the directory to 'ERMAPPER\algorithm\Data_Examples\Landsat_TM'. Double click on the 'RGB_531.alg' from the directory. The RGB_531 color composite is displayed.

- 2 Follow the procedure in chapter 3 and create a "RGB_531_16x16.tif" tiff file.

Create a batch script with variables.

Tip: You can use the template "Skeleton.erb" file from the ERMAPPER\batch\template directory. The file can be edited using your favourite Text (ASCII) editor or the **Notepad** text editor accessed through ER Mapper under the **Utilities/User Menu/Edit** a File option.

- 1 Load the "Skeleton.erb" file from the 'ERMAPPER\batch\template' directory into a Text Editor.

```
#####

# Copyright 1995 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:  $ERMAPPER/batch/XXXX.erb
# Date:    XXXX
```

```
# Author:   XXXX
#
# Summary:  XXXX
#
# Details:  XXXX
#
#
include "lib/BE_Startup.erb"
XXXX
#####
```

2 Edit the file as follows:

```
#####

# Copyright 1995 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   $ERMAPPER/batch/var_math.erb
# Date:     4th August, 1997
# Author:   John Smith
#
# Summary:  Testing variables and mathametic functions in a batch script file
#
# Details:  XXXX
#
#
include "lib/BE_Startup.erb"

$var1 = 1
$var2 = $var1
$var3 = $var1 + $var2
println $var1
println $var2
println $var3

$hello = "Hello"
$world = "World"
$hello_world = $hello + $world
println $hello
println $world
println $hello_world
#####
```

Note: **Println** command is to print a variable in a new line. **Print** command is to print a variable without carrying on to a new line.

- 3 Save the batch script file as “var_math.erb” in the ‘ERMAPPER/batch’ directory
- 4 Exit the ER Mapper and call it again so that it will recognize your test1.bar toolbar file in the ‘ERMAPPER/config’ directory
- 5 On the ER Mapper main menu click on the **Toolbar** menu. Your “test1.bar” appears as “test1” in the **Toolbar** menu in alphabetical order.

Tip: Your batch script file “var_math.erb” in the ‘ERMAPPER/batch’ directory should have the same name as the batch script (“JS_RGB_531_16x16” “Example BATCH var_math” BATCH “var_math”) in the “test1.bar” toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 6 Click the “test1” on the **Toolbar** menu. Two buttons will be displayed on the Toolbar. The first one is for the CALLBACK annotate_cb function and the second button is for the var_math batch script.
- 7 Click on the batch script file (second) button.
- 8 The Batch Engine Output dialog box appears and the result are printed in the Batch Engine Output dialog box.

Note: Batch scripts are run in the Batch Script Engine separate from ER Mapper. To display an algorithm written in a script you need to copy the algorithm to ER Mapper and display it in an image window.

2:Input

This exercise shows you how to input a raster dataset, an algorithm and various types of vector files such as ER Mapper vector file and a ‘.dxf’ file. You will also learn how to use **ask yes/no** command.

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

- 1 Load your var_math.erb batch script file into a text editor and add the script in bold letters.

```
#####

println $hello
println $world
println $hello_world
println $string_number

ask begin title "User Input Test"

    say "Please enter a dataset file name"
    ask file "Dataset:" ".ers" $dataset_input
    say "Please enter an algorithm"
    ask file "Algorithm:" ".alg" $algorithm_input
    say "Please enter some text"
    ask text "Text:" $some_text_input
    say "Please enter a DXF vector file"
    ask link "DXFLink:" ".dxf" $vector_DXF_input
    say "Please enter a number"
    ask number "Number:" $some_number_input
    say "Please select Yes/No"
    ask yesno "YesNo:" $yesno_input

ask end

println "start printing"
println $dataset_input
println $algorithm_input
println $some_text_input
println $vector_DXF_input
println ~3 $some_number_input
println $yesno_input
println "end printing"

$count = 3
$filename[$count] = "vegetation"
println $filename[3]
exit

#####
```

Note: You can also dynamically link ARCINFO file with ER Mapper. For example: **ask link "ARCINFOLink:" "\$CHOOSER=arc_chooser \$DEFAULT" \$vector_ARCINFO_input** and to link with an ER Mapper vector file you will use the following **ask link** command: **ask link "ERVLink:" ".erv" \$vector_ERV_input**, for DXF vector files : **ask link "DXFLink:" ".dxf" \$vector_DXF_input** and for tabular format vector files : **ask link "TBLLink:" ".tbl" \$vector_TBL_input**

- 2 Save the batch script file and run it from the **View** menu on the ER Mapper main menu window. Select the **Batch Engine Script Control..** option from the **View** menu drop down list.
- 3 The **Batch Engine Script Control** dialog box appears.
- 4 On the **Batch Engine Script Control** dialog box click on the **Run Script..** button.
- 5 The **Run a batch script** dialog box appears.
- 6 On the **Run a batch script** dialog box double click on the batch script file and run the batch script.

Note: This is the preferred option in running a batch script while testing a batch script. When you are satisfied with your batch script you should run the batch script from a toolbar or wizard button.

- 7 Variable, file names, strings, numbers and values for **Yes/No** variables are printed in the **Batch Engine Output** dialog box.

Note: File chooser dialog box appears with default extension for respective file type. For example, when you input a raster dataset, the file chooser dialog box will show files with '.ers' extension.

Note: The number you have typed in is printed out with 3 decimals. That is because you have specified it with ~3 to print it out in 3 decimals.

Note: The value for **yes** of the **Yes/No** variable is 1.000000 and the value for **no** of the **Yes/No** variable is 0.000000.

An example: Variables and input

```
#####

# Copyright 1995 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_var_math_input.erb
# Date:     4th August, 1997
# Author:   Abdullah Mah
#
# Summary:  Understanding numerical, string variables and importing data
#
# Details:
#           (1) variables (Numerical and string variables),
#           (2) mathematical functions (* / + -)
#           and (3) ask begin/end block for input data
#
include "lib/BE_Startup.erb"

$var1 = 1
$var2 = $var1
$var3 = $var1 + $var2
println $var1
println $var2
println $var3
$hello = "Hello"
$world = "World"
$hello_world = $hello + $world
$string_number = $hello_world + $var3
println $hello
println $world
println $hello_world
println $string_number
ask begintitle "User Input Test"
    say "Please enter a dataset file name"
    ask file "Dataset:" ".ers" $dataset_input
    say "Please enter an algorithm"
    ask file "Algorithm:" ".alg" $algorithm_input
    say "Please enter some text"
    ask text "Text:" $some_text_input
    ask number "Number:" $some_number_input
```

```

        say "Please select Yes/No"
        ask yesno "YesNo:" $yesno_input
    ask end
    println "start printing"
    println $dataset_input
    println $algorithm_input
    println $some_text_input
    println ~3 $some_number_input
    println $yesno_input
    println "end printing"
    $count = 3
    $filename[$count] = "vegetation"
    println $filename[3]
    exit
#####

```

Note: Syntaxes of ask number, ask text, ask file & ask yes/no commands are as follow: ask number "prompt text" \$number_input; ask text "prompt text" \$text_input; ask file "prompt text" "default directory" "default file" ".ers" \$file name; ask yes/no "prompt text" \$yesno_input.

Note: Syntaxes of all the **ask number**, **ask text**, **ask file** and **ask yes/no** commands have "prompt text" and variables to store the input. The syntax of ask file command has options for "default directory" "default file" and ".ers" file extension.

Exercises:

(1) Write a batch script to solve the following equation. Print the result in two decimal real numbers in the Batch Engine Output dialog box.

$$A = B1 / (B1 + B2 + B3) * B4 \quad \text{where } B1=20, B2=35, B3=45, B4=60$$

(2) Write a batch script to print the following comment in the Batch Engine Output dialog box.

“BIL or Band-Interleaved by Line is the ER Mapper raster data format.”

(3) Write a batch script to select a raster dataset (.ers), an algorithm (.alg), a DXF (.dxf) vector file and print the file names in the Batch Engine Output dialog box.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:
 - For Windows, select **Close** from the window control-menu.
 - For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to write batch scripts to perform the following tasks :

- Define and manipulate numerical and string variable.
- Select various type of files: raster, algorithm, vector
- Confirm the index numbers of **Yes** and **No** for **ask yes/no** command
- Run batch scripts from toolbar buttons
- Run batch scripts from the **View** menu on the main menu window

Image manipulation, Density slicing and RGB composite

This chapter shows you how to write a batch script to select a data file, process it and display it as a single band densitysliced image and a RGB composite.

About image manipulation

Manipulating images using batch scripting language is done through the Batch Engine- quite separate from ER Mapper itself. Hence, if an algorithm is developed using the scripting language, to display it you must copy it to ER Mapper and do a **Go** in the window. If on the other hand after viewing it, you want to change the algorithm, you must copy the algorithm from ER Mapper into the Batch Engine. You do the modification to the algorithm in the batch engine and copy it back to ER Mapper and do a **Go** in the window to display it.

Hands-on exercises

These exercises teach you how to select a dataset, select bands from the dataset and assign them to layers, enhance transforms of the bands and display the processed images.

What you will learn...

After completing these exercises, you will know how to import a dataset, select bands using the **ask bandmenu** command, set the bands to layers, enhance the transforms of the bands and display the algorithm.

- Select a dataset
- Set the algorithm mode to pseudo
- Choose bands from the dataset using **ask bandmenu** command
- Enhance the transforms of the bands in the Batch Engine
- Copy the algorithm to ER Mapper and display the processed image

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

1: Densitysliced image

Note: To run a batch script from a toolbar or wizard button, the batch script file has to be included in the toolbar file. Hence, add the name of the batch script you are going to create in the “test1.bar” toolbar file.

- 1 Edit the “test1.bar” file and add a new line (“**JS_Density_16x16**” “**Example BATCH densitysliced**” **BATCH “densitysliced”**)

```
#####
```

```
# Copyright 1990/91 Earth Resource Mapping Pty Ltd.
```

```
# This document contains unpublished source code of
```

```
# Earth Resource Mapping Pty Ltd. This notice does
```

```
# not indicate any intention to publish the source
```

```
# code contained herein.
```

```
#
```

```
# FILE: config/test1.bar
```

```
# CREATED: 4th August, 1997
# AUTHOR: John Smith
# PURPOSE: Testing batch scripts

#####

"JS_RGB_531_16x16" "Example BATCH var_math" BATCH "var_math"

"JS_Density_16x16" "Example BATCH densitysliced" BATCH
"JS_densitysliced"

#####
```

Note: You have excluded the CALLBACK function and added "JS_densitysliced" batch script file which will refer to the "JS_densitysliced.erb" batch script file in the 'ERMAPPER/batch' directory. You will create the "JS_densitysliced.erb" batch script file below.

Create an image Tiff file to be used as the picture in the icon of the toolbar button

Process a densitysliced image that you would like to use it as the picture in the icon of your batch script toolbar button.

Tip: You can use an exiting Tiff file from the ERMAPPER\icon directory

- 1 Create a densitysliced image using a TM band of the "Landsat_TM_year_1985.ers" dataset.

Note: The \Landsat_TM_year_1985.ers dataset is in the 'ERMAPPER\dataset\Core_Datasets' directory.

- 2 Follow the procedure in chapter 3 and create a "JS_Density_16x16.tif" tiff file.
- 3 Place the tiff file in the 'ERMAPPER/icons' directory.

Write a batch script to process and display a densitysliced image.

Tip: You can use the template “Skeleton.erb” file from the ‘ERMAPPER\batch\template’ directory. The file can be edited using your Text editor or the editor included in ER Mapper under the **Utilities/User Menu/Edit** a File option.

- 1 Load the “Skeleton.erb” file from the ERMAPPER\batch\template directory into a Text Editor.

```
#####  
  
# Copyright 1995 Earth Resource Mapping Pty Ltd.  
  
# This document contains unpublished source code of  
  
# Earth Resource Mapping Pty Ltd. This notice does  
  
# not indicate any intention to publish the source  
  
# code contained herein.  
  
#  
  
# Script: $ERMAPPER/batch/XXXX.erb  
  
# Date:   XXXX  
  
# Author: XXXX  
  
#  
  
# Summary: XXXX  
  
#  
  
# Details: XXXX  
  
#  
  
#  
  
include "lib/BE_Startup.erb"  
  
XXXX  
  
#####
```

- 2 Edit the file as follows:

```
#####  
  
#  
  
# Copyright 1995 Earth Resource Mapping Pty Ltd.
```

```

# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_densitysliced.erb
# Date:     4th August 1997
# Author:   Abdullah Mah
#
# Summary:  Select a dataset, select a band and display it as a densitysliced image
#
# Details:
#   1) Use ask file command to select a dataset
#   2) Use ask bandmenu command to select a band
#   3) Set the layer, color mode
#   4) Enhance the transform
#   5) Copy algorithm to window and display the densitysliced image
#
include "lib/BE_Startup.erb"
#set the default band
$band =1
# Import a dataset and select a band
ask begintitle "User Input Dataset"
    say "Please select a file and a band"
    ask file "Dataset:" ".ers" $dataset_input
    ask bandmenu "Which band do you want to use? " $dataset_input $band
ask end
# Call a new window and make it the current window
new window
# Call a new algorithm, set up its description and mode
new algorithm
set algorithm description "Density Sliced"
set algorithm mode pseudo
# setup the layer's description, band, lut and display it
first layer
    set layer description "Pseudocolor"
    set layer dataset to $dataset_input
    set layer input 1 to band $band
    set algorithm lut to "pseudocolor"
    copy algorithm to window
go window
# copy the algorithm from the window to the batch engine
# to set the transform limits to actual

```



```

copy algorithm from window
    first layer
    first transform
    set transform limits to actual
    copy algorithm to window
go window
# copy the algorithm from the window to the batch engine
# to set the transform clip to 95% and apply smoothing filter
copy algorithm from window
    first layer
    first transform
    set transform clip to 95 # gaussian equalize
    set algorithm supersample Type to bilinear
    copy algorithm to window
go window
exit
#####

```

Tip: (i) Select a dataset and band/s in an **ask begin/end block** (ii) use **ask file** command to select a file. The syntax used is (**ask file “prompt text” “default directory” “.ers” \$filename**). (iii) use **ask bandmenu** command to select a band from the file. The syntax used is (**ask bandmenu “prompt text” \$filename \$band_number**). (iv) There has to be current window, algorithm, layer to create an algorithm and enhance the image. Hence set them up with **new window, new algorithm, first surface, first active raster layer** commands. (v) Set algorithm mode and lut as, **set algorithm mode pseudo** and **set algorithm lut to “greyscale”**. (Note: **set algorithm mode pseudo** and **set algorithm mode to pseudo** will function the same. **To** is optional). (vi) To display the algorithm, copy the algorithm to window and display it.

- 3 Save the batch script file as ‘JS_densitysliced.erb’ in the ‘ERMAPPER/batch’ directory
- 4 Exit the ER Mapper and call it again so that it will recognize your ‘test1.bar’ toolbar file in the ‘ERMAPPER/config’ directory
- 5 On the ER Mapper main menu click on the toolbar menu. Your ‘test1.bar’ will appear as ‘test1’ in the Toolbar menu in alphabetical order.

Tip: Your batch script file '**JS_densitysliced.erb**' in the 'ERMAPPER/batch' directory should have the same name as the batch script ('JS_Density_16x16' "Example BATCH densitysliced" BATCH "**JS_densitysliced**") in the 'test1.bar' toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file 'test1.bar', for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 6 Click the 'test1' on the Toolbar menu. Two buttons will be displayed on the Toolbar. The first one is for the var_math batch script and the second button is for the densitysliced batch script.
- 7 Click on the densitysliced batch script (second) button, the User Input Dataset dialog box will appear.

Note: Batch scripts are run in the Batch Script Engine separate from ER Mapper. To display processed image copy algorithm to window and display it.

- 8 Click the load dataset button on the User Input Dataset window and select a file from the file chooser.
- 9 Select a band from the drop down list from the band selection button on the User Input Dataset window.
- 10 Click the OK button on the User Input Dataset window.
- 11 The densitysliced image of the selected band is displayed.

Note: To check the algorithm, click the **View Algorithm for Image Window** button on the main menu. The algorithm window will appear. Check the algorithm. You will see that smoothing is applied and the transform of the selected band has been enhanced with clip to 99% transform.

2:RGB composite

This exercise teaches you how to select a dataset, select bands from the dataset and assign them to layers, enhance transforms of the bands and display the processed RGB composite.

What you will learn...

After completing this exercise, you will know how to import a dataset, set the algorithm mode to rgb, select bands from bandmenu, assign the bands to red, green and blue layers, enhance the transforms of the bands and display the RGB algorithm.

- Select a dataset
- Set the algorithm mode to rgb
- Set layers to red, green and blue layers
- Choose bands from the dataset using **ask bandmenu** command
- Enhance the transforms of the bands in the Batch Engine
- Copy the algorithm to ER Mapper and display the processed RGB composite

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

Note: To run a batch script from a toolbar or wizard button, the batch script file has to be included in the toolbar file. Hence, add the name of the batch script you are going to create in the “test1.bar” toolbar file.

- 1 Edit the “test1.bar” file and add a new line “**JS_RGB_16x16**” “**Example BATCH RGB**” **BATCH “JS_RGB”**”

```
#####
```

```
# Copyright 1990/91 Earth Resource Mapping Pty Ltd.
```

```
# This document contains unpublished source code of
```

```
# Earth Resource Mapping Pty Ltd. This notice does
```

```
# not indicate any intention to publish the source
```

```
# code contained herein.
```

```
#
```

```
# FILE: config/test1.bar
```

```
# CREATED: 4th August, 1997
```

```
# AUTHOR: John Smith

# PURPOSE:Testing CALLBACK function annotate_cb

#####

“JS_RGB_531_16x16” “Example BATCH var_math” BATCH “JS_var_math”

“JS_Density_16x16” “Example BATCH densitysliced” BATCH
“JS_densitysliced”

“JS_RGB_16x16” “Example BATCH RGB” BATCH “JS_RGB”

#####
```

Note: You have added “JS_RGB” batch script file which will refer to the ‘JS_RGB.erb’ batch script file in the ‘ERMAPPER/batch’ directory. You will create the ‘JS_RGB.erb’ batch script file below.

Create an image Tiff file to be used as the picture in the icon of the toolbar button

Process an image that you would like to use it as the picture in the icon of your batch script toolbar button.

Tip: You can use an exiting Tiff file from the ERMAPPER\icon directory

- 1 Create a RGB composite image using three TM bands from the “Landsat_TM_year_1985.ers” dataset.

Note: The \Landsat_TM_year_1985.ers dataset is in the ‘ERMAPPER\dataset\Core_Datasets’ directory.

- 2 Follow the procedure in chapter 3 and create a ‘JS_RGB_16x16.tif’ tiff file. Place the tiff file in the ‘ERMAPPER/icons’ directory.

Write a batch script to process and display a RGB composite.

Tip: You can use the template “Skeleton.erb” file from the ERMAPPER\batch\template directory. The file can be edited using your Text (ASCII) editor or the editor included in ER Mapper under the **Utilities/User Menu/Edit** a File option.

- 1 Load the “Skeleton.erb” file from the ERMAPPER\batch\template directory into a Text Editor.

```
#####  
  
# Copyright 1995 Earth Resource Mapping Pty Ltd.  
  
# This document contains unpublished source code of  
  
# Earth Resource Mapping Pty Ltd. This notice does  
  
# not indicate any intention to publish the source  
  
# code contained herein.  
  
#  
  
# Script: $ERMAPPER/batch/XXXX.erb  
  
# Date: XXXX  
  
# Author: XXXX  
  
#  
  
# Summary: XXXX  
  
#  
  
# Details: XXXX  
  
#  
  
#  
  
include "lib/BE_Startup.erb"  
  
XXXX  
  
#####
```

- 2 Edit the file as follows:

```
#####  
  
#  
  
# Copyright 1995 Earth Resource Mapping Pty Ltd.
```

```

# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_RGB.erb
# Date:     4th August 1997
# Author:   Abdullah Mah
#
# Summary:  Importing a dataset, select three bands, process them as a RGB
composite
#
# Details:
#      1) Use ask file command to select a dataset
#      2) Set the algorithm mode to rgb
#      3) Set three layers as red, green and blue
#      4) Use ask bandmenu command to select three bands
#      5) load the three bands into red, green blue layers
#      6) Enhance the transforms of the bands in the three layers
#      7) Copy the algorithm to window and display the RGB composite
#
include "lib/BE_Startup.erb"
ask begintitle "User Input Dataset and bands"
    say "Please select a dataset"
    ask file "Dataset:" ".ers" $dataset_input
    ask bandmenu "Select a band  for Blue Layer" $dataset_input $band1
    ask bandmenu "Select a band for Green Layer" $dataset_input $band2
    ask bandmenu "Select a band  for Red  Layer" $dataset_input $band3
ask end
# Call a new window and make it as current window
new window
# Set the algorithm description, mode
# Set the layer description, dataset and bands in layers
    new algorithm
    set algorithm description "RGB_Composite"
    set algorithm mode rgb
    add blue layer
    set layer description "blue"
    set layer dataset to $dataset_input
    set layer input 1 to band $band1
    add green layer
    set layer description "green"
    set layer dataset to $dataset_input
    set layer input 1 to band $band2

```

```

        add red layer
        set layer description "red"
        set layer dataset to $dataset_input
        set layer input 1 to band $band3
# Copy algorithm to window and display the image
        copy algorithm to window
        go window
# Copy algorithm from window to batch engine to enhance transform
        copy algorithm from window
# Cycle through all layers setting the transform clip to 99.0%
        first blue layer
if ($ERROR !=0) then goto no_algorithm
        next_active_layer:
        first transform
        set transform limits to 95.00
        set algorithm supersample Type to bilinear
        next active raster layer
        if ($ERROR ==0) then goto next_active_layer
no_algorithm:
# Copy algorithm to window and display the image
include "lib/Delete_All_Inactive_Layers.erb"
#include "lib/Clip_99_All_Active_Layers.erb"
        copy algorithm to window
        go window
exit
#####

```

Tip: (i) You can easily add specific layer types by simply **add red/green/blue layer/s**. (2) Use specific machine variable **\$ERROR** as a control and loop through the three red/green/blue layers to enhance the transforms. This reduces the length of script. (3) Take advantage of the existing scripts by just including the **"lib/Delete_All_Inactive_Layers.erb"** script to delete any inactive layer/s.

- 3 Save the batch script file as "JS_RGB.erb" in the 'ERMAPPER/batch' directory
- 4 Exit the ER Mapper and call it again so that it will recognize your test1.bar toolbar file in the 'ERMAPPER/config' directory
- 5 On the ER Mapper main menu click on the toolbar menu. Your "test1.bar" will appear as " test1" in the Toolbar menu in alphabetical order.

Tip: Your batch script file “**JS_RGB.erb**” in the ERMAPPER/batch directory should have the same name as the batch script (“**JS_RGB_16x16**” “**Example BATCH RGB**” BATCH “**JS_RGB**”) in the “test1.bar” toolbar file.

Tip: You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 6 Click the “test1” on the Toolbar menu. Three buttons will be displayed on the Toolbar. The first one is for the JS_var_math batch script, the second button is for the JS_densitysliced batch script and the third button is for the JS_RGB batch script.
- 7 Click on the JS_RGB batch script (third) button, the User Input Dataset and bands dialog box will appear.

Note: Batch scripts are run in the Batch Script Engine separate from ER Mapper. To display an algorithm written in a script you need to copy the algorithm to ER Mapper and display it in an image window.

- 8 Click the load dataset button on the **User Input Dataset and bands** window and select a file from the file chooser.
- 9 Select three bands from the drop down list from the band selection buttons on the User Input Dataset window.
- 10 Click the OK button on the **User Input Dataset and bands** window.
- 11 The RGB composite of the three selected bands is displayed.

Note: To check the algorithm, click the **View Algorithm for Image Window** button on the main menu. The algorithm window will appear. Check the algorithm. You will see that smoothing is applied to all the bands and the transforms of the selected bands have been enhanced with clip to 99% transform.

Exercises:

- (1) Write a batch script to select a raster dataset, select a band from the dataset, process and display it as a densitysliced image. Use greyscale lookuptable and gaussian equalize transform to enhance the image.
- (2) Write a batch script to select a raster dataset, select three bands from the dataset, process and display it as a RGB composite image. Enhance the transforms of the red/green/blue layers individually with clip to 97%. Modify the script and use \$ERROR variable as a control to loop through the three layers and enhance the transforms with gaussian equalize transform. Delete the inactive layer/s before displaying the processed image.

Close all image windows and dialog boxes

- Close all image windows using the window system controls:
- For Windows, select **Close** from the window control-menu.
- For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to write batch scripts to perform the following tasks in ER Mapper:

- Write a script to process a densitysliced image
- Write a script to process a RGB composite image
- Use **\$ERROR** variable as a control and loop through the red/green/blue layers to enhance the transforms.
- Include '**lib/Delete_All_Inactive_Layers.erb**' batch script to delete inactive layer/s

6

Wizard Scripting

The ER Mapper batch script language has been enhanced in release 6.0 to allow you to create interactive guided wizard scripts. Wizards are made up of one or (usually) more pages that step a user through a task. They guide the user to making choices, providing any necessary information and suggesting appropriate choices when possible. They are especially useful for complex tasks or tasks requiring a number of steps. This chapter describes how to create a wizard using a batch script. General batch script commands mentioned in previous chapters are applicable in writing wizard scripts.

Example wizard script

Batch commands to create a wizard interface can be inserted into any batch script. The structure of the script will vary depending on the nature of the task but will generally be made up of the following sections.

- a setup section in which variables and default values are set up and any necessary information is copied from the current algorithm or elsewhere
- a wizard section in which the layout and contents of the wizard pages are specified, as well as where to store any user input and any procedures to be run as a consequence of the user's choices
- an outcome section in which the procedures to be carried out as a consequence of the user's choices in the wizard are specified.

This chapter is concerned with the wizard section.

The simple wizard script 'Image_View.erb' is shown below.

```
#####

# Script:   Image_View.erb
# Date:     Friday Dec6 WST 1996
# Author:   Mike Dunne
# Summary:  Wizard to view an image
# Details:  Wizard wrapper around Create_RGB.erb
# Creates an RGB algorithm from the current window algorithm.
# It uses the following strategy:
#1)    Look for any valid RGB layers.If any are present, turn them
#       on, run & exit.
#2)    Check the current layer - if it has a dataset it uses it to
#       create the algorithm (simply turning the layer on if it's a
#       pseudocolor layer).
#3)    Look for an active pseudocolor layer.  If one is found, use
#       its dataset to create the RGB algorithm.
#4)    Look for any raster layer with a valid dataset.  Use it to
#       create the new algorithm.
#5)    At this point there is no valid dataset, so ask for one &
#       use it to create the RGB algorithm.

include "lib/BE_Startup.erb"
if current window then goto window_ok
new window

window_ok:
copy algorithm from window

# Set the layer to it
$LIB_lay1 = current layer
$LIB_alg1 = current algorithm
$LIB_tra1 = last transform
$image_filename = get layer dataset
#####
```

As shown in the example above, wizard pages are defined using the following format.

```
#####

include "lib/BE_Startup.erb"                                #wizard set up

WizardPage1:                                                #label for page 1
WizardPage begin "WizardName"                                #title for first page
    title "Page1Name"                                        #if a new wizard
    container begin "Image"
```

```

container information                #usually image info
...                                #on first page
container end
container begin "DataEntry"
    container information
    ...
container end
container begin "PageControls"  #control buttons
    ask action "< Back"
    ask action "Next >" goto WizardPage2
    ask action "Cancel" close
container end
WizardPage end

WizardPage2:                        #label for page 2
WizardPage begin                    #no title
    title "Page2Name"
    container begin "container1name"
        container information
        ...
    container end
    container begin "container2name"
        container information
        ...
    container end
    container begin "container3name"
        container information
        ...
container end
WizardPage end
#####

```

The layout of wizard pages

Wizard pages are made up of a number of different areas. The exact design of wizard pages varies but generally the first page consists of:

- a graphic on the left
- a short paragraph that welcomes the user and explains what it does
- controls for entering or editing wizard input if there is enough space
- navigation buttons at the bottom

and subsequent pages consist of:

- an information section at the left
- a user input area to the right
- navigation buttons at the bottom
- graphics can be used if they help illustrate the process and are easily recognised as being non-interactive
- Scroll bars appear if the container sizes would otherwise cause the dialog to be larger than 440 pixels wide by 320 pixels high.
- The wizard does not check values that are entered by the users^{3/4}it is up to your batch script to do this.

WizardPage Block

Each wizard page is set up by WizardPage Block, defined between a `WizardPage begin' and `WizardPage end' statement. You need as many WizardPage blocks as there are pages in the wizard. Each of the areas `contains' different things and is therefore known as a `container'. The size and contents of each container must be specified. An example wizard block is:

```
WizardPage begin "WizardName" # title for first page
    title "Page1Name" # if a new wizard
    container begin "Image"
        container information # usually image info
        ...# on first page
    container end
    container begin "DataEntry"
        container information
        ...
    container end
    container begin "PageControls # control buttons
        ask action "<Back"
        ask action "Next >" goto label1
        ask action "Cancel" close goto label2
    container end
wizard close
WizardPage end
```

The entries in a Wizard Block are as follows:

Wizard begin "WizardName" or WizardPage begin "WizardName"

This starts a new wizard page. 'WizardName' is the name of the wizard within the script. If it is a different name from the previous Wizard Block a new wizard window is drawn; if it is the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. WizardName is optional for second and subsequent pages: if omitted it defaults to the name from the previous Wizard Block.

For example,

Wizard begin "ClassificationWizard"

defines the beginning of the Wizard Block for the first page in a wizard while,

Wizard begin

defines the beginning of the Wizard Block for subsequent pages in the wizard.

Title "title_text" Specifies the title which will appear at the top of the wizard page on screen. This should include the name of the Wizard (which indicates what it is used for), followed by a hyphen and the step number. For example,

title "ClassificationWizard - Step 1 of 4"

Container blocks See "Container block" below. If no container block is defined a default container block the size of the whole page will be included.

Wizard end or Wizardpage end Defines the end of the wizard page.

Wizard close [Name] If you don't specify a Name, all wizards are closed. If you specify a Name, then the named wizard, and all sibling wizards created after it, are closed.

The "Wizard Close" command is needed because you often can't automatically close a wizard when the user presses the Finish button. You usually need to check the necessary parameters have been entered. In this case you would check everything is fine, and if it is, close the wizard using the "Wizard Close" command, and if it is not, go back to the appropriate page of the wizard. You can use Exit to exit a script and close down the wizard automatically but this is not recommended.

Container block

A container block defines the size, contents and layout of a single 'container' or 'pane' in a wizard page. Any number of containers can be defined though realistically there will be only two or three per page. For example,

The entries in a Container Block are described below.

Container Begin "Name" Defines a new container. Name is the name of the container within the script. Each of the containers on a single page (within a single Wizard block) must have a different name. For example,

Container begin "Image"

[Container] [Items] Horizontal | Vertical | Newline (Optional) Specifies the direction of placement of the items in a container defined between this command and the next Container Items command. If omitted, the items will be placed vertically. For example,

Container Items Horizontal

or

Horizontal

[Container] Labels_above | Labels_left (Optional). Specifies where the labels for an item will appear. The default is 'above'. For example,

Container labels above

[Container] Above | Below | Left | Right "Name" (Optional). Specifies the placement of the container. The default is for container to be below the previous container.

[Container] width_pct | height_pct Percentage_number (Optional). Specifies the container width and height as a percentage of the remaining space. For example,

Container width_pct 30

width_pct 30

[Container] right|left justify (Optional) This is used to justify a button or a row of buttons. It will not affect other item types such as lists and files as these are placed so as to make best use of the existing space. For example,

container right justify

ask action "< Back"

ask action "Next >"

Ask and show commands Any standard Ask commands can be included in containers. See "Navigation buttons container" below.

Container End Ends the definition of the container.

You can use the 'container' commands as separate statements within a container block, you can put a number of them together on a single line or add them to the end of the "Container Begin" command. However, the number of words on a single line is limited to 12. The following example has 9 words.

items horizontal width_pct 30 height_pct_30 labels_left right justify

Navigation buttons container

The last container on any page should contain the standard navigation buttons.

```
container begin "con2"
container height_pct 12
container below "con1"
ask action "< Back"
ask action "Next >" goto wizard_page2
ask action "Finish" goto check_parameters_and_close
ask action "Cancel" close goto wizard_close
container end
```

The buttons should be programmed to work in the following way:

<Back Returns to the previous page. It should be greyed out on the first page. When the user presses the Back button they should be presented with any choices they have made up to that point rather than the original defaults. Therefore defaults should be set up before the first Wizard Block.

Next> Moves to the next page in the sequence, maintaining whatever settings the user provides in previous pages. This is greyed out on the last page.

Finish Applies the settings defined in the wizard (either default or input by the user) to ER Mapper and completes the task. The finish button must appear on the last page and should be included at any point that the wizard can complete the task (even the front page if there are reasonable defaults).

Cancel Discards any settings specified in the wizard, terminates the process, and closes the wizard window. The **Cancel** button is always the right most button.

A single page wizard has only a **Finish** and a **Cancel** button.

For a consistent look multi-page wizards should have **Next** and **Back** buttons on all pages. However, the **Next** button should be greyed out on the last page and the **Back** button should be greyed out on the first page.

Ask and Show commands

All the "Ask", "Say" and "Title" commands, such as

Ask Yesno "Show Contours:" \$contours_yn

work within a Wizard block. However, the 'Ok', 'Cancel' and 'Help' buttons which are automatically added to Ask Blocks do not appear in Wizard blocks.

In addition, a number of new Ask commands have been designed especially for specifying input into wizards. They include, for example, 'Ask Action' which draws the navigation buttons on the bottom of the wizard page.

Show Image "Filename"

Shows the image TIF file in the container, following the same rules that other Ask/Show items do. The image is shown at its true size, not fitted to the container size. The image files default directory is 'ERMAPPER/icons'.

General guidelines for wizard pages

- Wizards should have Windows 95 look and feel.
- Pages should flow linearly through a series of steps. It is possible to jump from one wizard to another but this should be avoided as it will confuse the user. The user should not have to use any functions outside of the wizard to complete a task.
- Wizard pages should be easy to understand without having to read them very carefully. It is better to use more simple pages than fewer complex pages.
- If it isn't obvious, the last page should give the user information about how to proceed when the wizard is finished.
- Wizards don't automatically progress to the next page. the user may have neglected to provide all necessary information, in which case a 'Wizard Error' will be reported. The wizard does not proceed until the appropriate field is filled in.
- Use a conversational writing style, with words like you' and 'your', contractions and short, common words.

·Ask users what they would like to do rather than telling them what to do. Thus, use 'which option do you want ' or 'would you like ' instead of 'choose a layout'.

·Avoid using technical terminology that may be confusing to a novice user.

·Use as few words as possible.

·Keep the writing clear, concise and simple, but don't be condescending.

·It is quite possible (as with ask forms) to decide under program control if containers or ask/say commands are to be added to a wizard form. This makes it quite powerful.

·If a label name of a 'labels_left' container item is too long, part of it will not be displayed. If this occurs, either change to 'labels_above' or shorten the label.

Example wizard

Below is an example of a wizard. This and other examples can be found in the 'ERMAPPER/batch' directory.

```
#####

#
# Script:    3D_Wizard.erb
# Date:      Mon Jan 13 10:26:02 WST 1997
# Author:    Mark Sheridan
#
# Summary:   Wizard to create a standard single surface 3D algorithm
#
# Details:   Ask for a pseudo or RGB alg, choose the datasets,
#             create an algorithm, copy it to the window and
#             set the view mode to 3D. Simple!
#
include "lib/BE_Startup.erb"
$yesno_Pseudocolor = "1"
$yesno_RGB = "0"
$listmenu["0"] = "Psuedo Color"
$listmenu["1"] = "RGB"
$list_choice = $listmenu["0"]
$RasterDataset = ""
$HeightDataset = ""
$lut = "greyscale"
wizard_page_1:
Wizard begin "3D Algorithm Wizard"
    title "3D Algorithm Wizard"
    container begin "Mode"
        container items labels_left
        container right "Image"
        say "This Wizard will help you to create a"
        say "single surface, 3D algorithm."
        say ""
        say "You need a raster dataset and a height"
        say "dataset over the same area."
        say ""
        say "Select the type of 3D algorithm you want"
        say "to produce:"
        ask listmenu_exclusive "" $listmenu $list_choice
    container end
container begin "PageControls"
    container height_pct 12
    container below "Mode"
    container items horizontal right justify
    ask action "< Back"
    ask action "Next >" goto CheckPseudo_or_RGB
```

```

        ask action "Cancel" close goto WizardCancel
    container end
    container begin "Image"
        container width_pixels 131
        container height_pixels 280
        container left "Mode"
        container above "PageControls"
        show image "standard_icons/Wizards/3D_wiz"
    container end
Wizard end
CheckPseudo_or_RGB:
    if ($list_choice == "RGB") then goto wizard_page_2_rgb
    goto wizard_page_2_pseudo
wizard_page_2_rgb:
Wizard begin "3D Algorithm Wizard"
    title "3D Algorithm Wizard"
    container begin "InputDatasets"
        container items labels_above
        container right "Image2"
        say "Select the datasets to use in this"
        say "algorithm and the band to use for the"
        say "height layer:"
        say ""
        say "The raster dataset must have at least 3"
        say "bands to produce a 321-RGB algorithm"
        ask file "Raster dataset: " ".ers" $RasterDataset
        ask file "Height dataset: " ".ers" $HeightDataset
        say ""
        ask bandmenu "Band for height data:" $HeightDataset
    $bandmenu
    container end
    container begin "Image2"
        container width_pixels 131
        container height_pixels 280
        container above "PageControls2"
        show image "standard_icons/Wizards/3D_wiz"
    container end
    container begin "PageControls2"
        container height_pct 12
        container below "InputDatasets"
        container items horizontal right justify
        ask action "< Back" goto wizard_page_1
        ask action "Finish" goto WizardFinish
    container end

```

```

        ask action "Cancel" close goto WizardCancel
    container end
Wizard end
wizard_page_2_pseudo:
Wizard begin "3D Algorithm Wizard"
    title "3D Algorithm Wizard"
    container begin "InputDatasets"
        container items labels_above
        container right "Image2"
        say "Select the datasets to use in this"
        say "algorithm and the band to use for the"
        say "height layer:"
        say ""
        ask file "Raster dataset: " ".ers" $RasterDataset
        ask lutmenu "Color table:" $lut
        say ""
        ask file "Height dataset: " ".ers" $HeightDataset
        ask bandmenu "Band for height data:" $HeightDataset $bandmenu
    container end
    container begin "Image2"
        container width_pixels 131
        container height_pixels 280
        container above "PageControls2"
        show image "standard_icons/Wizards/3D_wiz"
    container end
    container begin "PageControls2"
        container height_pct 12
        container below "InputDatasets"
        container items horizontal right justify
        ask action "< Back" goto wizard_page_1
        ask action "Finish" goto WizardFinish
        ask action "Cancel" close goto WizardCancel
    container end
Wizard end
Warn1:
    say warning "You must choose a Raster dataset"
    if ($list_choice == "RGB") then goto wizard_page_2_rgb
    goto wizard_page_2_pseudo
Warn2:
    say warning "You must choose a Height dataset"
    if ($list_choice == "RGB") then goto wizard_page_2_rgb
    goto wizard_page_2_pseudo
WizardFinish:

```

Chapter 6 Wizard Scripting ●

```
if ($RasterDataset == "") then goto Warn1
if ($HeightDataset == "") then goto Warn2
wizard close
if ($list_choice == "RGB") then goto DoRGB
goto DoPseudo
*****
DoPseudo:
    $alg = new algorithm
    set $alg description "Pseudo Color 3D"
    set $alg mode pseudo
    set $alg lut to $lut
    $layer1 = first layer
    set $layer1 description "Raster"
    set $layer1 dataset to $RasterDataset
    $lasttran= last transform
    set $lasttran limits to actual
    add height layer
    $layer2 = current layer
    set $layer2 description "Height"
    set $layer2 dataset to $HeightDataset
set $alg view mode to perspective
    new window
    copy algorithm to window
    go window
    goto FinishUp
*****
DoRGB:
    $alg = new algorithm
    set $alg description "RGB 3D"
    set $alg mode rgb
    $layer1 = first layer
    set $layer1 type red
    set $layer1 description "Red"
    set $layer1 dataset to $RasterDataset
    set $layer1 input 1 to band 1
    $lasttran= last transform
    set $lasttran limits to actual
add green layer
    $layer2 = current layer
    set $layer2 description "Green"
    set $layer2 dataset to $RasterDataset
    set $layer2 input 1 to band 2
    $lasttran= last transform
```

```

    set $lasttran limits to actual
    add blue layer
    $layer3 = current layer
    set $layer3 description "Green"
    set $layer3 dataset to $RasterDataset
    set $layer3 input 1 to band 3
    $lasttran= last transform
    set $lasttran limits to actual
    add height layer
    $layer4 = current layer
    set $layer4 description "Height"
    set $layer4 dataset to $HeightDataset
set $alg view mode to perspective
    new window
        copy algorithm to window
    go window
    goto FinishUp
*****
WizardCancel:
FinishUp:
exit
#####

```


Wizard Scripts, Densityslicing and RGB composite

This chapter shows you how to create a wizard script. It describes how to create wizard pages, select a dataset, select single and multiple bands, build Densitysliced and RGB composite algorithms.

About wizards

Wizards are made up of one or more pages that step a user through a task. They guide the user to making choices, providing any necessary information and suggesting appropriate choices when possible. They are especially useful for complex tasks or tasks requiring a number of steps. General batch script commands mentioned in the last chapter are applicable in writing wizard scripts.

As in batch script, wizard scripts execute the commands through batch engine - quite separate from ER Mapper itself. Hence, to display an algorithm developed in a wizard script it has to be copied to ER Mapper and do a **Go** in the window.

Hands-on exercises

These exercises teach you how to create wizards with multiple pages, select a dataset, select a single and multiple bands from the dataset and assign them to layers, enhance transforms of the bands and display the processed images.

What you will learn...

After completing these exercises, you will know how to create wizard pages, select a dataset using ask file command, select single and multiple bands using ask bandmenu command, assign the bands to layers, enhance the transforms of the bands and display the algorithm.

- Create multiple page wizards.
- Set wizard pages using **wizard begin/end** blocks.
- Use **container begin/end** blocks in wizard pages to select dataset/s, bands and set options to select certain image processing tasks such as smoothing etc.
- Select a dataset using **ask file** command in a **container begin/end** block.
- Use **ask bandmenu** command to choose bands for pseudocolor and RGB layers in a **container begin/end** block.
- Use **ask action** command to navigate wizard page and unconditional (**goto---**) and conditional (**if ---- then ---- else goto---**) controls to channel the processing tasks.
- Set the color mode to **pseudo** and **rgb**.
- Copy the algorithms (Densitysliced & RGB composite) to ER Mapper and display the processed image/s.

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

1: Wizard : Densityslicing

Note: To run a wizard script from a wizard button, the wizard script file has to be included in the toolbar file. Hence, add the name of the wizard script you are going to create in the “test1.bar” toolbar file.

Edit the “test1.bar” file and add a new line “**JS_wizard_densitysliced_16x16**”
“**Example BATCH wizard_densitysliced**” BATCH “**JS_wizard_densitysliced**”

#####

Copyright 1990/91 Earth Resource Mapping Pty Ltd.

```
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# FILE:  config/test1.bar
# CREATED: 4th August, 1997
# AUTHOR: John Smith
# PURPOSE:Testing batch and wizard scripts
#####
"JS_RGB_531_16x16" "Example BATCH var_math" BATCH "JS_var_math"
"JS_Density_16x16" "Example BATCH densitysliced" BATCH
"JS_densitysliced"
"JS_RGB_16x16" "Example BATCH RGB" BATCH "JS_RGB"
"JS_wizard_densitysliced_16x16" "Example wizard_densitysliced"
BATCH "JS_wizard_densitysliced"
#####
```

Note: You have added "JS_wizard_densitysliced" wizard script file which will refer to the "JS_wizard_densitysliced.erb" wizard script file in the 'ERMAPPER/batch' directory. You will create the "JS_wizard_densitysliced.erb" wizard script file below.

Create an image Tiff file to be used as the picture in the icon of the wizard button

Process a Tiff image to use it as the picture in the icon of the wizard button.

Tip: You can use an exiting Tiff file from the ERMAPPER\icon directory

- 1 Create a densitysliced image using a magnetic image from the "Newcastle_Magnetics.ers" dataset. (The Newcastle_Magnetics.ers dataset is in the ERMAPPER\dataset\Core_Datasets directory)

- 2 Follow the procedure in chapter 3 and create a 24-bit 16x 16 pixels “JS_wizard_densitysliced_16x16.tif” tiff file. Place the tiff file in the ERMAPPER/icons directory.
- 3 Also create a 24-bit 64x 64 pixels “JS_wizard_densitysliced_64x64.tif” tiff file. Place the tiff file in the ‘ERMAPPER/icons/standard_icons’ directory.

Note: This 64x64 pixels tiff file will be used on the wizard page. To use a tiff file of a size that will fit a specific area on the wizard you can create a tiff file using other software and import it into ER Mapper.

Write a wizard script to process and display a densitysliced image.

Logic:

- Select a single band from a dataset.
- Set the color mode to pseudocolor.
- Assign a color from the color lookup table.
- Enhance the densitysliced image using transformation.
- Display the image.

Processing

- Use a **wizard begin/end** block to set up a wizard page.
- In the first **container begin/end** block of the wizard page 1, use **ask file** command to select a dataset.
- In the first **container begin/end** block of the wizard page 1, use **ask bandmenu** command to select a band from the dataset.
- In the second **container begin/end** block of the wizard page 1, use **ask action** command to navigate processing tasks.
- Set the color mode to pseudocolor.
- Select a color from the color lookup table.
- Apply transformation to enhance the image.
- Copy the algorithm to window and display the densitysliced image.

Writing the wizard script

Type the following wizard script in a Text Editor.

```
#####  
  
#  
# Copyright 1995 Earth Resource Mapping Pty Ltd.
```

```

# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_wizard_Densitysliced.erb
# Date:     4th August 1997
# Author:   Abdullah Mah
#
# Summary:  Creating a wizard. Importing a dataset, select one bands,
#           process it as densitysliced image
#
# Details:
#           1) Use Wizard begin/end block and create a wizard page
#           2) Inside the wizard begin/end block use container begin/end
#           block
#           3) Use ask file command to select a dataset
#           4) Use ask bandmenu command to select a band
#           5) Set the color mode to pseudo
#           6) Use ask lutmenu command to select a color from CLUT
#           7) Use ask yesno command for smoothing option
#           8) Use ask action commands to navigate the process
#           9) Enhance the transforms of the band
#           10) Copy the algorithm to window and display the
#           Densitysliced image
#
include "lib/BE_Startup.erb"      #wizard set up
Wizard begin "Densitysliced Wizard"
    title "Densitysliced Wizard - Image Properties"
    container begin "DataEntry"
        labels_left
        say "This page allows you to specify properties for the"
        say "Densitysliced image to be created."
        say ""
        ask file "Select a dataset" ".ers" $Dataset
        say ""
        ask bandmenu "Select a band for Densitysliced image " $Dataset
$band1
        say ""
        say ""
        newline
        labels_above
        ask lutmenu "Color Lookup Table" $lut
        say ""

```

Chapter 7 Wizard Scripts, Densityslicing and RGB composite ● 1: Wizard : Densityslicing

```
        say ""
        newline
        ask yesno "Smoothing " $yesSmoothing
        say ""
    container end
    container begin "PageControls"
        height_pct 12
        below "DataEntry"
        horizontal right justify
        ask action "Finish" goto Start_Process
        ask action "Cancel" close goto WizardCancel
    container end
Wizard end
Start_Process:
new window
    say status "Creating Densitysliced image...\n"
    new algorithm
    set algorithm description "Densitysliced"
    set algorithm mode pseudo

    first pseudo layer
    set layer description "Pseudo"
    set layer dataset to $Dataset
    set layer input 1 to band $band1
    last transform
    set surface lut to $lut
    copy algorithm to window
    go window

    copy algorithm from window
    first pseudo layer
    set layer description "Pseudo"
    set layer dataset to $Dataset
    set layer input 1 to band $band1
    last transform
    set transform limits to actual
    copy algorithm to window
    go window

    copy algorithm from window
    first pseudo layer
    last transform
    set transform clip to 95
```

```

        if ($yesSmoothing == 1) then goto smoothing else goto no_smoothing
smoothing:
        set algorithm supersample type to bilinear
no_smoothing:
        include "lib/Delete_All_Inactive_Layers.erb"
        copy algorithm to window
        go window
        say status "Done.\n"
        delete status
WizardCancel:
Wizard close
exit
#####

```

- 1 Save the wizard script file as “JS_wizard_densitysliced.erb” in the ‘ERMAPPER/batch’ directory
- 2 Exit the ER Mapper and call it again so that it will recognize your ‘test1.bar’ toolbar file in the ‘ERMAPPER/config’ directory
- 3 On the ER Mapper main menu click on the toolbar menu. Your “test1.bar” will appear as “test1” in the Toolbar menu in alphabetical order.

Tip: Your wizard script file “**JS_wizard_densitysliced.erb**” in the ‘ERMAPPER/batch’ directory should have the same name as the batch script (“JS_wizard_densitysliced_16x16” “Example BATCH wizard_densitysliced” BATCH “**JS_wizard_densitysliced**”) in the “test1.bar” toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and run it again.

- 4 Click the “test1” on the Toolbar menu. Four buttons will be displayed on the Toolbar. The first one is for the JS_var_math batch script, the second button is for the JS_densitysliced batch script, the third button is for the JS_RGB batch script and the fourth button is for the JS_wizard_densitysliced wizard script.
- 5 Click on the JS_wizard_densitysliced wizard script (fourth) button.
- 6 The Densitysliced Wizard - Image Properties dialog box appears.

Note: Batch and wizard scripts are run in the Batch Script Engine separate from ER Mapper. To display the processed image copy the algorithm to window and display the algorithm.

- 7 On the Image Properties dialog box select "Newcastle_Magnetics.ers" dataset from the 'ERMAPPER/dataset/Core_Datasets' directory.
- 8 Select a band
- 9 Select a color from the color lookup table
- 10 Click on the small text field to the left of **Smoothing** if you wish to apply smoothing on the densitysliced image
- 11 Click the **Finish** button
- 12 The densitysliced image is displayed in the image window.

2: Wizard : RGB composite

Logic:

- Select a dataset with at least three bands.
- Set the color mode to RGB color mode
- Select three bands from the dataset and put them in the RGB layers
- Enhance the bands in RGB layers using transformation
- Display the RGB composite

Processing

- Use a wizard begin/end block to set up a wizard page
- In the first container begin/end block of the wizard page 1, use ask file command to select a dataset.
- In the first container begin/end block of the wizard page 1, use ask bandmenu command to select three bands from the dataset.
- In the second container begin/end block of the wizard page 1, use ask action command to navigate processing tasks.
- Set the color mode to RGB color mode
- Apply transformation to red, green and blue layers to enhance the RGB composite
- Copy the algorithm to window and display the RGB composite image

**Load the above written wizard script in a Text Editor and edit it as below:
(New wizard scripts are in bold letters)**

```
#####  
  
#  
# Copyright 1995 Earth Resource Mapping Pty Ltd.  
# This document contains unpublished source code of
```

```

# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_wizard_RGB.erb
# Date:    4th August 1997
# Author:   Abdullah Mah
#
# Summary:  Importing a dataset, select three bands, process them as RGB composite
#
# Details:
#   1) Use Wizard begin/end block and create a wizard page
#   2) Inside the wizard begin/end block use container begin/end block
#   3) Use ask file command to select a dataset
#   4) Use ask bandmenu command to select three bands
#   5) Set the algorithm mode to rgb
#   6) Add red, green, blue layers and load the bands
#   7) Use ask yesno command for smoothing option
#   8) Use ask action commands to navigate the process
#   9) Use $ERROR error code to setup loop and cycle through
#       the three layers and enhance the transforms of the three bands
#  10) Delete inactive layers
#  11) Copy the algorithm to window and display the RGB composite
#
include "lib/BE_Startup.erb"
Wizard begin "RGB Wizard"
    title "RGB Wizard - Image Properties"
    container begin "DataEntry"
        labels_left
        say "This page allows you to specify properties for the"
        say "RGB compoiste to be created."
        say ""
        ask file "Select a dataset" ".ers" $Dataset
        ask bandmenu "Select a band for Red? " $Dataset $band1
        ask bandmenu "Select a band for Green? " $Dataset $band2
        ask bandmenu "Select a band for Blue? " $Dataset $band3
        say ""
        newline
        ask yesno "Smoothing " $yesSmoothing
        say ""
    container end
    container begin "PageControls"
        height_pct 12

```



```

        below "DataEntry"
        horizontal right justify
        ask action "Finish" goto Start_Process
        ask action "Cancel" close goto WizardCancel
    container end
Wizard end
Start_Process:
new window
    say status "Creating RGB composite image...\n"
    new algorithm
    set algorithm description "RGB"
    set algorithm mode rgb

    add red layer
    set layer description "Red"
    set layer dataset to $Dataset
    set layer input 1 to band $band1

    add green layer
    set layer description "green"
    set layer dataset to $Dataset
    set layer input 1 to band $band2

    add blue layer
    set layer description "blue"
    set layer dataset to $Dataset
    set layer input 1 to band $band3
    copy algorithm to window
    go window

    copy algorithm from window
    first active raster layer
    if ($ERROR != 0) then goto no_algorithm
next_active_layer:
    last transform
    set transform clip to 95
    next active raster layer
    if ($ERROR == 0) then goto next_active_layer
no_algorithm:
    if ($yesSmoothing == 1) then goto smoothing else goto no_smoothing
smoothing:
    set algorithm supersample type to bilinear
no_smoothing:

```

```
include "lib/Delete_All_Inactive_Layers.erb"
  copy algorithm to window
  go window
  say status "Done.\n"
  delete status
WizardCancel:
Wizard close
exit
#####
```

- 1 Save the wizard script file as “JS_wizard_RGBcomposite.erb” in the ‘ERMAPPER/batch’ directory.
- 2 Edit the “test1.bar” file from the ‘ERMAPPER/config’ directory and add a new line “**JS_wizard_RGBcomposite_16x16**” “**Example wizard_RGBcomposite**” BATCH “JS_wizard_RGBcomposite”
- 3 Exit the ER Mapper and call it again so that it will recognize your ‘test1.bar’ toolbar file in the ‘ERMAPPER/config’ directory
- 4 On the ER Mapper main menu click on the Toolbar menu. Your “test1.bar” appears as “ test1” in the Toolbar menu in alphabetical order.
- 5 Click on the JS_wizard_RGBcomposite button. The RGB Wizard - Image Properties dialog box appears.

Note: **RGB Wizard - Image Properties** is the title of the wizard and you can change it as you wish in the wizard script.

- 6 On the **RGB Wizard - Image Properties** dialog box select “Newcastle_Radiometrics.ers” dataset from the ERMAPPER/dataset/Core_Datasets directory.
- 7 Select K for Red, Th for Green and U for Blue layers
- 8 Select the smoothing option.
- 9 Click Finish and display the RGB composite image

3: Wizard : Densitysliced & RGB composite

In writing a wizard with options to create densitysliced and RGB composite images you need to channel the processing path appropriately for each image processing option. In fact there are three options ; (i) to process only the

densitysliced image (ii) to process only the RGB composite image and (iii) to process both the densitysliced and RGB composite images consecutively one after another.

Logic:

- Writing a wizard with options to create densitysliced and RGB composite images requires multiple wizard pages
- Wizard page number one to select dataset and the image processing option, either densityslicing or RGB composite or both.
- Wizard page number two to select image properties either only for densityslicing or only for RGB composite or for both techniques
- After selecting the image properties you need to channel the image processing task according to the option/s selected: Either to process only the densitysliced image or only the RGB composite image or both.

Processing

- Wizard page 1: A wizard page is within a wizard begin/end block. There can be many container begin/end blocks within a wizard page. In the first container begin/end block use ask file command to select a dataset (A dataset with only one band will allow you to process densitysliced image; A dataset with three bands or more will allow you to process both types of images); use yesno statement to ask the user to select the processing option/s.
- In the second container block of wizard page 1, use ask action command to navigate the process.
- Wizard page 2: In the first container begin/end block of wizard page 2, set options to choose a single band for densityslicing, three bands for RGB composite or to choose respective band/s for both techniques on the same page if both options were chosen.
- In the second container block of wizard page 2, use ask action command to navigate the process.
- Use conditional (**if ---- then ---- else goto---**) and unconditional (**goto---**) controls to navigate the processing tasks.
- Set the proper color mode, layers and apply transformation to enhance the image/s
- Copy the slgorithm/s to window and display the processed image/s.

**Load the above written wizard script in a Text Editor and edit it as below:
(New wizard scripts are in bold letteres)**

```
#####
#
# Copyright 1997 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
```

```

# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_wizard_densitysliced_RGB.erb
# Date:     31 August 1997
# Author:   Abdullah Mah
#
# Summary:  Creating a wizard with multi-tasks. Importing a dataset,
#           select to create densitysliced and/or RGB composite images.
#           select appropriate band/s pseudo and RGB colordraped algorithms
#
# Details:
#   1) Use Wizard begin/end block and create wizard pages
#   2) Inside the wizard begin/end blocks use container begin/end blocks
#   3) Use ask file command to select a dataset
#   4) Setup option/s to create densitysliced and/or RGB composite image/s
#   5) Use ask bandmenu command to select a band for densitysliced image
#   6) Use ask bandmenu command to select three bands for RGB image
#   7) Set the algorithm mode to pseudo/rgb
#   8) Use ask yesno command for smoothing option
#   9) Use ask action commands to navigate the process
#  10) Check validity of the dataset.
#  11) Enhance the transforms of the band/s
#  12) Delete inactive layers
#  13) Copy the algorithm to window and display the densitysliced/RGB image/s
#
include "lib/BE_Startup.erb"
# set the default bands
$band = 1
$band1= 1
$band2= 2
$band3= 3
#      *      *      *      *      *      *      *      *      *
Page1:
Wizard begin "Densitysliced & RGB composite"
  title "Densitysliced & RGB composite - Data Entry"
  container begin "DataEntry"
    container items labels_left
    say ""
    say ""
    say "This allows you to create Densitysliced & RGB composite images"
    say ""
    newline

```

```

        container items labels_above
        ask file "Select a dataset ?" ".ers" $Dataset1
        newline
        say ""
        say "Which image do you want to create ?"
        ask yesno "Densitysliced" $Densitysliced
        ask yesno "RGB composite" $RGBcomposite
        say ""
        say ""
        newline
        ask yesno "Smoothing " $yesSmoothing
        say ""
    container end
    container begin "Images"
        width_pixels 64
        left "DataEntry"
        above "PageControls"
        show image "standard_icons/JS_wizard_densitysliced_64x64"
    container end
    container begin "PageControls"
    container height_pct 10
    container below "DataEntry"
    container items horizontal right justify
        ask action "Next >" goto Page1Next
        ask action "Finish" goto Page1Finish
        ask action "Cancel" close goto WizardCancel
    container end
Wizard end

Page1Next:
    $Goto = "Page2"
    goto ValidateData

Page1Finish:
    $Goto = "Finish"
    goto ValidateData
#      *      *      *      *      *      *      *      *      *      *
Page2:
Wizard begin "Densitysliced & RGB composite"
    title "Densitysliced and/or RGB composite - Image Properties"
    container begin "DataEntry"
        items labels_left
        if ($Densitysliced ==1) then goto yesdensitysliced else goto
nodensitysliced
        yesdensitysliced:
            say ""

```

```

        say "          Select a band for Densitysliced image"
        say ""
        ask bandmenu "Select a band for Densitysliced layer? " $Dataset1
$band

        goto continue1
nodensitysliced:
continue1:
        if ($RGBcomposite ==1) then goto yesRGBcomposite else goto
noRGBcomposite
        yesRGBcomposite:
            say ""
            say"          Select three bands for RGB composite"
            say ""
            ask bandmenu "Select a band for Blue layer? " $Dataset1 $band1
            ask bandmenu "Select a band for Green layer? " $Dataset1 $band2
            ask bandmenu "Select a band for Red layer? " $Dataset1 $band3
            goto continue2
        noRGBcomposite:
            if ($Densitysliced == 1 or $RGBcomposite == 1) then goto continue2
            say warning "Please select an option Densitysliced and/or RGB
composite"

            goto Page1
        continue2:
        container end
        container begin "PageControls"
        container height_pct 10
        container below "DataEntry"
        container items horizontal right justify
            ask action "< Back" goto Page2Back
            ask action "Finish" goto Page2Finish
            ask action "Cancel" close goto WizardCancel
        container end
Wizard end
Page2Back:
    $Goto = "Page1"
    goto ValidateData
Page2Finish:
    $Goto = "Finish"
    goto ValidateData
#      *      *      *      *      *      *      *      *      *      *
ValidateData:
    if ($Dataset1 != "") then goto DatasetOK
    say warning "Please specify a dataset for Densitysliced and/or RGB
composite"
    goto Page1

```

```

DatasetOK:
    if ($Goto == "Page1") then goto Page1
    if ($Goto == "Page2") then goto Page2
    if ($Goto == "Finish") then goto CheckDensitysliced
#      *      *      *      *      *      *      *      *      *      *
CheckDensitysliced:
if ($Densitysliced==1) then goto beginDensitysliced else goto CheckRGBcomposite

beginDensitysliced:
    new window
    first surface
    new algorithm
    set algorithm description "Densitysliced"
    first pseudo layer
    set layer description "Pseudo"
    set layer dataset to $Dataset1
    set layer input 1 to band $band
    copy algorithm to window
    go window
    copy algorithm from window
        first surface
        first pseudo layer
        first transform
        set transform limits to actual
        first intensity layer
        next transform
        set transform limits to actual

        copy algorithm to window
    go window
    copy algorithm from window
        include "lib/Clip_99_All_Active_Layers.erb"
        set algorithm supersample Type to bilinear
include "lib/Delete_All_Inactive_Layers.erb"
if ($yesSmoothing == 1) then goto smoothing else goto no_smoothing
smoothing:
    set algorithm supersample type to bilinear
no_smoothing:
    copy algorithm to window
    go window
#      *      *      *      *      *      *      *      *      *      *
CheckRGBcomposite:
if ($RGBcomposite == 1) then goto beginRGBcomposite else goto continue4

```

```

beginRGBcomposite:
    new window
    first surface
    new algorithm
    set algorithm description "RGB composite"
    set algorithm mode rgb
    add blue layer
    set layer description "Blue"
    set layer dataset to $Dataset1
    set layer input 1 to band $band1
    $bcount = get layer band count
println $bcount
    if ($bcount < 3) then goto reset_default_B23_layers else goto multiplebands
reset_default_B23_layers:
$band2=1
$band3=1
multiplebands:
    add green layer
    set layer description "Green"
    set layer dataset to $Dataset1
    set layer input 1 to band $band2
    add red layer
    set layer description "Red"
    set layer dataset to $Dataset1
    set layer input 1 to band $band3
    copy algorithm to window
    go window
    copy algorithm from window
        first surface
# Cycle through all layers setting the transform to limits to actual
    first blue layer
if ($ERROR !=0) then goto no_algorithm
    next_active_layer:
    first transform
    set transform limits to actual
    next active raster layer
    if ($ERROR ==0) then goto next_active_layer
no_algorithm:
# Copy algorithm to window and display the image

    copy algorithm to window
    go window

```



```

        copy algorithm from window
            first surface
# Cycle through all layers setting the transform to limits to 95%
    first blue layer
if ($ERROR !=0) then goto no_algorithm
    next_active_layer:
    first transform
    set transform clip to 95
    next active raster layer
    if ($ERROR ==0) then goto next_active_layer
no_algorithm:
    set algorithm supersample Type to bilinear
include "lib/Delete_All_Inactive_Layers.erb"
if ($yesSmoothing == 1) then goto smoothing1 else goto no_smoothing1
smoothing1:
    set algorithm supersample type to bilinear
no_smoothing1:
    copy algorithm to window
    go window
#      *      *      *      *      *      *      *      *      *      *
continue4:
    wizard close
    say status "Done.\n"
    delete status
WizardCancel:
exit
#####

```

- 1 Save the wizard script file as “JS_wizard_densitysliced_rgb.erb” in the ‘ERMAPPER/batch’ directory
- 2 Edit the “test1.bar” file from the ‘ERMAPPER/config’ directory and add a new line “**JS_wizard_densitysliced_rgb_16x16**” “**Example wizard_densitysliced_rgb**” **BATCH** “**JS_wizard_densitysliced_rgb**”
- 3 Exit the ER Mapper and call it again so that it will recognize your ‘test1.bar’ toolbar file in the ‘ERMAPPER/config’ directory
- 4 On the ER Mapper main menu click on the Toolbar menu. Your “test1.bar” appears as “ test1” in the Toolbar menu in alphabetical order.
- 5 Click the ‘test1’ from the dropdown list of the Toolbar menu.
- 6 From the displayed numerous buttons, click on the JS_wizard_densitysliced_rgb button.
- 7 The Densitysliced & RGB - Data Entry dialog box appears. This is the first wizard page.

Note: Densitysliced & RGB - Data Entry is the title of the wizard and you can change it as you wish in the wizard script.

- 8 On the Densitysliced & RGB - Data Entry dialog box select Densitysliced option as the image you want to create. Then select “Newcastle_Magnetics.ers” dataset from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 9 Select the smoothing option
- 10 Click the Next button on the Densitysliced & RGB - Data Entry dialog box
- 11 Densitysliced and/or RGB composite - Image Properties dialog box appears. This is the second wizard page.
- 12 The first band of the dataset is automatically selected for you. Since the Newcastle_Magnetics.ers dataset has only one band, the magnetics band is selected for you.

Note: For a dataset with more than one bands, select the band of your choice

- 13 Click the Finish button to display the densitysliced image and quit the program.
- 14 Repeat the same procedure; select a dataset with three or more bands, select three bands for the RGB layers and display the RGB composite image.
- 15 Repeat the same procedure; select a dataset with three or more bands, select both densitysliced and RGB composite options, select appropriate band/s and display both the densitysliced and the RGB composite images consecutively.

Exercises:

(1) Write a wizard script to process a densitysliced image. In the wizard script select a raster dataset, select a band from the dataset, process and display it as a densitysliced image. Use pseudocolor from the lookuptable and gaussian equalize transform to enhance the image.

(2) Write a wizard script to process a RGB composite. In the wizard script select a raster dataset, select three bands from the dataset, process and display it as a RGB composite image. Enhance the transforms of the red/green/blue layers individually with clip to 98%. Use \$ERROR variable as a control to loop through the three layers. Delete the inactive layer/s before displaying the processed image.

(3) Write a wizard script with options to process a Densitysliced image and a RGB composite. In the script use separate wizard pages to process the two image processing techniques. Use **ask action** command and conditional **If .. then .. else goto** and unconditional **goto** controls to navigate processing tasks. Process two 24 bit 64x64 pixel size Tiff images and use them in the first wizard page.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:
 - For Windows, select **Close** from the window control-menu.
 - For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to perform the following tasks in ER Mapper:

- Create a Densitysliced wizard.
- Create a RGB composite wizard.
- Create a wizard with options to process a Densitysliced image and a RGB composite.

Hardcopy printing

This chapter describes how to write batch and wizard scripts to print hardcopies of the processed algorithms using Hardcopy Control Files option.

About hardcopy printing

Images are sent to hardcopy devices by a program called `ermhe'. Output to a hardcopy device involves 3 programs: ermhe, a filter program, and an output program. The filter and output programs used are specified in the hardcopy control files which are stored in the directory 'ERMAPPER/hardcopy'.

These three stages are shown below.

The hardcopy engine

The hardcopy engine gathers information about a hardcopy device by looking for a file with suffix `.hc' in the 'ERMAPPER/hardcopy' directory. The information in this file includes the page size, print density, and the filter and output programs to use when sending an image to the device.

The output from ermhe is piped into the filter program.

The Filter Program

The filter program takes the data in hardcopy engine format and converts it into the format required by the hardcopy device. The output from the filter program is then piped into the output program.

The Output Program

The output program takes the hardcopy device specific format and pipes it to the

hardcopy device. Usually the output program is some form of the "lpr" (lpr is for UNIX) command (the print spooler). lpr creates a printer job in a spooling area for subsequent printing as facilities become available. If the print spooler was lpr, the hardcopy definition file

OutputProgram command would be as follows:

OutputProgram = "lpr -Ppaintjet" OR OutputProgram = "lpr -Pdjet650C" (It will print out from DesignJet 650C plotter) ("lpr" for UNIX only)

The -P option sends the output to the printer called paintjet or plotter dj650C. For more information on the lpr options see "man lpr".

If your default printer is named something different, then change the output program line -P printer switch.

You can change the OutputProgram to save the plot to a file using "he_writetofiles".

An example of this might be:

OutputProgram = "\"he_writetofile \$ERMTMP/DesignJet.erm\""

Tip: "he_writetofile" can be used for both PC and UNIX. "lpr" is for UNIX only.

The he_writetouniqfiles script outputs each strip of the image to a different file on disk. This may be useful if you wish to print the images in a batch mode. The files are named time.ermhe, time being in the format "HH:MM:SS".

ERMHE:

You must specify the algorithm to process. The usage and switches of **ermhe** are as follows:

ermhe: Usage: ermhe -a Algorithm -h Hardcopy [options]

Algorithm is the algorithm file to process, Hardcopy is the device to output to and [options] are:

- l : fit output to device page size
- b r,g,b : color of null cells
- c [centimeters] : output size in centimeters
- d x,y : device resolution in dpi
- f : force single page

-g : debug
 -h [hardcopy file] : hardcopy file to use
 -i [inches] : output size in inches
 -m [meters] : output size in meters
 -r : force 1 dataset cell = 1 image pixel
 -s [scale] : output scale, e.g. -s 10000 means 1:10,000
 -t x,y : maximum number of dots per strip
 -x : Use GUI dialog window
 -C : Use dynamic compilation
 -D x,y : force dots down & across for the image (use with -f)
 -F [filter program] : program to filter hardcopy
 -G r,g,b : Gamma colour correction
 -L : Orientation is Landscape
 -O [output program] : program to output hardcopy
 -S : GUI setup of hardcopy parameters
 -Z flag,Zscale : Stereo pair generation
 -P : Native (PC) printing (PC Only)
 -p [file name] : Native (PC) printing with config file (internal use only)
 -H [file name] : Input algorithm file to delete on close(internal use only)
 -I [file name] : Input dataset header file to delete on close(internal use only)
 -J [file name] : Input data file to delete on close(internal use only)
 -K [file name] : Input arcinfo coverage to delete on close(internal use only)

Only one of the -s, -i, -c and -m switches can be used.

The -f flag only has an effect when the -r switch is used.

Setting up your own output program

- Click the **Print** button on the main menu of ER Mapper. The **Print** dialog box appears.
- On the **Print** dialog box select **Hardcopy Control Files** option.

- Select the appropriate hardcopy control file from the import hardcopy control files button of the **Output Name:**

Note: In this manual 'ERMAPPER/hardcopy/HP/HP_DesignJet_300dpi_A0.hc' is selected.

- Click the **Setup..** button on the **Print** dialog box. The **Setup** dialog box appears.
- On the Setup window type in the appropriate plotter name in the text field of the Output program.

Note: Example: Output program: **he_wrietofile** \\server\dj650C (plotter name)

- Click the **Save As..** button on the **Setup** dialog box.
- Save your hardcopy control file as '**MyHCfile**' in the Hardcopy/HC directory.

Note: Your '**MyHCfile.hc**' hardcopy control file is saved in the Hardcopy/HC directory.

- Use your hardcopy control file with -h switch of the ermhe script.

Note: Example: **-h D:\ERMAPPER\hardcopy\HP\MyHCfile.hc**

Hands-on exercises

These exercises teach you how to write batch and wizard scripts to print algorithms.

What you will learn...

After completing these exercises, you will know how to write batch and wizard scripts to print algorithms.

- Select a dataset and process a densitysliced image.
- Set system command.
- Use your hardcopy control file and print the processed densitysliced image.
- Select algorithms to print.
- Use your hardcopy control file and print the selected algorithms.

Printing in Batch Script

Process a tiff image to use it as the picture in the icon of your batch script button. Name the tiff file as JS_batch_print_16x16.tif and save it in the 'ERMAPPER/icons' directory.

Note: To run a batch script from a button, the batch script file has to be included in the toolbar file. Hence, add the name of the batch script you are going to create in the "test1.bar" toolbar file.

- 1 Edit the "test1.bar" file and add a new line **"JS_batch_print_16x16"**
"Example BATCH Batch_print" BATCH "JS_batch_print"

Writing the batch script

Type the following batch script in a Text Editor.

```
#####
#
# Copyright 1997 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   batch_print.erb
# Date:     4th August 1997
# Author:   Abdullah Mah
#
# Summary:  Printing multiple algorithms
#
# Details:  1. select a band FROM a dataset
#           2. set the algorithm
#           3. set the layer and enhance the image
#           4. Option to print the processed algorithm
#           5. set the system ermhe command
#           6. print the algorithm
#
include "lib/BE_Startup.erb"
ask begintitle "User Input Dataset"
    say "Please select a file and a band for a Densitysliced image"
    ask file "Dataset:" ".ers" $dataset_input
    ask bandmenu "Which band do you want to use? " $dataset_input $band
```


Chapter 8 Hardcopy printing ● Printing in Batch Script

```
Say ""
ask yesno "Print the processed algorithm" $yesnoPrint
ask end
new window
    new algorithm
    set algorithm description "Densitysliced"
    set algorithm mode pseudo
    set algorithm supersample type to bilinear
    first layer
        set layer description "Pseudocolor"
        set layer dataset to $dataset_input
        set layer input 1 to band $band
        set algorithm lut to "pseudocolor"
        copy algorithm to window
    go window
    copy algorithm from window
        first layer
        last transform
        set transform limits to actual
        copy algorithm to window
    go window
    copy algorithm from window
        first layer
        last transform
        set transform clip to 95
        copy algorithm to window
    go window
    copy algorithm from window
    $test_algo = "test.alg"
    $test_algo = $ERM TMP + "\" + $test_algo
    save algorithm $test_algo
    println $test_algo
    if ($yesnoPrint==1) then goto Print1 else goto DoNotPrint1
Print1:
    $sys_command = "ermhe -a " + $test_algo + " "+"-h D:/ERMAPPER/hardcopy/HP/
MyHCfile.hc"
    system $sys_command status
DoNotPrint1:
exit
#####
```

Tip: (i) Select a dataset and band/s in an **ask begin/end block** (ii) use **ask file** command to select a file. The syntax used is (**ask file “prompt text” “default directory” “.ers” \$filename**). (iii) Use **ask bandmenu** command to select a band from the file. The syntax used is (**ask bandmenu “prompt text” \$filename \$band_number**). (iv) Use **ask yesno** command to setup the option to print or not to print the processed algorithm. (v) There has to be current window, algorithm, layer to create an algorithm and enhance the image. Hence set them up with **new window, new algorithm, first layer** commands. (vi) Set algorithm mode and lut as, **set algorithm mode pseudo** and **set algorithm lut to “pseudocolor”**. (Note: **set algorithm mode pseudo** and **set algorithm mode to pseudo** will function the same. **To** is optional). (vii) Copy the algorithm to window and display it. (viii) Define the processed algorithm as a text file and save the algorithm. (ix) Set up the system command, use your hardcopy control file and print the algorithm.

- 2 Save the batch script file as 'JS_batch_print.erb' in the 'ERMAPPER/batch' directory
- 3 Exit the ER Mapper and call it again so that it will recognize your 'test1.bar' toolbar file in the 'ERMAPPER/config' directory
- 4 On the ER Mapper main menu click on the toolbar menu. Your 'test1.bar' will appear as 'test1' in the Toolbar menu in alphabetical order.

Tip: Your batch script file '**JS_batch_print.erb**' in the 'ERMAPPER/batch' directory should have the same name as the batch script ("JS_batch_print_16x16" "Example BATCH Batch_script" BATCH "**JS_batch_print**") in the 'test1.bar' toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file 'test1.bar', for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 5 Click the 'test1' on the Toolbar menu.
- 6 From the displayed numerous buttons click on the **batch_print** script button.
- 7 The **User Input Dataset** dialog box appears.
- 8 Click the load dataset button on the **User Input Dataset** window and select a file from the file chooser.
- 9 Select a band from the drop down list from the band selection button on the **User Input Dataset** window.
- 10 Click the OK button on the **User Input Dataset** window.
- 11 Select to print the processed algorithm.

- 12 The densitysliced image of the selected band is displayed and the algorithm is printed from the plotter.

Note: Batch scripts are run in the Batch Script Engine separate from ER Mapper. The algorithm of the processed image is copied to window and displayed.

Tip: To check the algorithm, click the **View Algorithm for Image Window** button on the main menu. The algorithm window will appear. Check the algorithm.

Printing in Wizard Script

Process a tiff image to use it as the picture in the icon of your batch script button. Name the tiff file as JS_wizard_multi_print_16x16.tif and save it in the 'ERMAPPER/icons' directory.

Note: To run a batch script from a button, the batch script file has to be included in the toolbar file. Hence, add the name of the batch script you are going to create in the "test1.bar" toolbar file.

- 1 Edit the "test1.bar" file and add a new line
"JS_wizard_multi_print_16x16" "Example Wizard_multi_print"
BATCH "JS_wizard_multi_print"

Writing the batch script

Type the following batch script in a Text Editor.

```
#####  
# Copyright 1997 Earth Resource Mapping Pty Ltd.  
# This document contains unpublished source code of  
# Earth Resource Mapping Pty Ltd. This notice does  
# not indicate any intention to publish the source  
# code contained herein.  
#  
# Script:   wizard_multi_print.erb  
# Date:     4th August 1997
```

```

# Author:   Abdullah Mah
#
# Summary:  Printing multiple algorithms in wizard
#
# Details:  1.select algorithms to be printed
#            2.assign algorithms in an array variable
#            3.setup an increment loop
#            4. apply if .. then ... else ...Conditional control
#            5.setup the system ermhe command
#            6.print the algorithms one at a time
#
include "lib/BE_Startup.erb"
# define variables
$alg[1] = ""
$alg[2] = ""
$alg[3] = ""
$increment = 1
$limit = 4
# Wizard page
Wizard begin "Multiple Print Wizard"
    title "Multiple Print Wizard - Algorithm/s Selection"
    container begin "DataEntry"
        labels_left
        say "This page allows you to choose the algorithm/s you want to
print."

        say ""
        say ""
        ask file "Select algorithm 1" ".alg" $alg[1]
        say ""
        ask file "Select algorithm 2" ".alg" $alg[2]
        say ""
        ask file "Select algorithm 3" ".alg" $alg[3]
        say ""
    container end
    container begin "PageControls"
        height_pct 12
        below "DataEntry"
        horizontal right justify
        ask action "> Next"
        ask action "Finish" goto Start_Print1
        ask action "Cancel" close goto WizardCancel
    container end
Wizard end
Start_Print1:

```

Chapter 8 Hardcopy printing ● Printing in Wizard Script

```
#check algorithm/s
    if ($alg[2] == "") and ($alg[3] == "") then goto adjust_limit1 else goto
check_algorithm1
adjust_limit1:
    $limit=2
    goto start_print2
check_algorithm1:
    if ($alg[3] == "") then goto adjust_limit2 else goto start_print2
adjust_limit2:
    $limit=3
    goto start_print2
check1:
next_algorithm:
    if $increment < $limit then goto start_print2 else goto end_print
#start printing
start_print2:
print "limit ="
print $limit
#    $sys_command = "ermhe -a " + $alg[$increment] + " "+"-h D:/ERMAPPER/hardcopy/
HP/MyHCfile.hc"
#    system $sys_command status
#    $increment = $increment + 1
#    goto next_algorithm
end_print:
WizardCancel:
Wizard close
exit
#####
```

- 2 Save the wizard script file as “JS_wizard_multi_print.erb” in the ‘ERMAPPER/batch’ directory
- 3 Exit the ER Mapper and call it again so that it will recognize your ‘test1.bar’ toolbar file in the ‘ERMAPPER/config’ directory
- 4 On the ER Mapper main menu click on the toolbar menu. Your “test1.bar” will appear as “ test1” in the Toolbar menu in alphabetical order.

Tip: Your wizard script file “**JS_wizard_multi_print.erb**” in the ‘ERMAPPER/batch’ directory should have the same name as the batch script (“**JS_wizard_multi_print_16x16**” “Example BATCH wizard_multiple print” BATCH “**JS_wizard_multi_print**”) in the “test1.bar” toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and run it again.

- 5 Click the “test1” from the Toolbar dropdown list.
 - 6 From the displayed buttons click on the **JS_wizard_multi_print** wizard button.
 - 7 The **Multiple Print Wizard - Algorithm/s Selection** dialog box appears.
 - 8 On the **Multiple Print Wizard - Algorithm/s Selection** dialog box, from the Select File button/s, select one, two or three algorithms.
 - 9 Click the **Finish** button
- The selected algorithm/s will be printed from the plotter.

Exercises:

(1) Write a wizard script with options to process a densitysliced image and/or a RGB composite image. Provide an option to select either the densitysliced or RGB composite or both images to be printed. Use system command to print the selected processed algorithm/s to print out from a plotter. Before writing the wizard script create your own hardcopy file and use it in the wizard script.

(2) Write a wizard script with options to select 5 algorithms which you have processed previously. Use system command and your hardcopy control file (which you have created before) to consecutively print out the selected one, two, three, four or all five algorithms from a plotter.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:
 - For Windows, select **Close** from the window control-menu.
 - For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).

Chapter 8 Hardcopy printing ● Exercises:

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to perform the following tasks in ER Mapper:

- Create your own hardcopy control file.
- Create a batch script to process a Densitysliced image and print it from a plotter.
- Create a wizard to print multiple algorithms using system command and the hardcopy control file you have created before.

9

Formula

Formula can be applied in batch scripting. Quick reference and full specifications of formula are given below. In batch scripting, formula are defined in strings. In applying formula in batch scripting it is necessary to set the formula first then define the input bands to be applied in the formula. For example:

```
#####  
add red layer  
set layer description "red"  
set layer dataset to $dataset1_input  
set layer formula to "i1/i2"  
set layer input 1 to band $band1  
set layer input 2 to band $band2
```

```
#####
```

The formula in the red layer is defined as “i1/i2”. After its definition input \$band1 is assigned to i1 and input \$band2 is assigned to i2. For long and complicated formula, the formula can be divided into sections and concatenated. For example:

```
#####  
add hue layer  
set layer description "hue"  
set layer dataset to $dataset_input
```



```

$formula_hue1 = "if (max(i1,i2,i3)=min(i1,i2,i3)) then 0 "
$formula_hue2 = "else ((if (i1=max(i1,i2,i3)) then ((i2-i3)/(max(i1,i2,i3)-
min(i1,i2,i3))) "
$formula_hue3 = "else if (i2=max(i1,i2,i3)) then 2 + ((i3-i1)/(max(i1,i2,i3)-
min(i1,i2,i3))) "
$formula_hue4 = "else 4 + ((i1-i2)/(max(i1,i2,i3)-min(i1,i2,i3)))*60+360)%360"
$formula_hue1234=$formula_hue1 + $formula_hue2 + $formula_hue3 +
formula_hue4

set layer formula to $formula_hue1234

set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3

#####

The long formula is divided into four sections and are assigned to $formula_hue1,
$formula_hu2, $formula_hue3 and $formula_hue4. They are then concatenated to
form the final formula which is $formula_hue1234.

```

Formula - Quick Reference

General

In Short form of an input specification (e.g. I1).

INPUTn Long form of an input specification (e.g. INPUT1).

Im..In Input list. A list of inputs (e.g. "I1..I4" is equivalent to "I1,I2,I3,I4")

Rn Short form of a region specification (e.g. R1).

REGIONn Long form of a region specification (e.g. REGION1).

DnShort form of a dataset specification (e.g. D1).

DATASETnLong form of a dataset specification (e.g. DATASET1).

'abc' Text can be specified in single quotes. Used to specify datasets and sometimes regions or bands(e.g. `ers/Australia_DTM').

bandlist Similar to input list, but can also use band numbers and/or descriptions (e.g. "B1..B3", "I1..I4,B7,'0.85_um'")

Operators

* multiplication
 /division
 + addition
 - subtraction
 - negation
 %modulus
 = is equal to
 != is not equal to
 > is greater than
 >= is greater than or equal to
 < is less than
 <= is less than or equal to
 AND logical and
 OR logical or
 NOT logical negation

Standard Functions

Usual Syntax: function(expression) or
 function(expression, expression)

ABS(x) absolute value of x
 SIN(x) sine of x (radians)
 COS(x)cosine of x (radians)
 TAN(x) tangent of x (radians)
 ASIN(x) arc sine of x (radians)
 ACOS(x) arc cosine of x (radians)
 ATAN(x) arc tangent of x (radians)
 EXP(x) e to the power of x (e**x)

LOG(x) natural log of x

LOG10(x) log to base 10 of x

POW(x,y)x to the power of y (x^{**y})

SQRT(x) square root of x

MIN(x,y,...) minimum of x, y, ...

MAX(x,y,...)maximum of x, y, ...

FLOOR(x) greatest integer less than or equal to x

CEIL(x)smallest integer greater than or equal to x

Special functions

INREGION(r) 1 if cell is inside region r1 otherwise 0.

ISNULL()1 if cell is null, otherwise 0.

CELLX() Cell column number)

CELLY() Cell row number

MAHAL(r1,i1,...) Mahalanobis classification discriminant function

PIXELX() Pixel column number

PIXELY() Pixel row number

Special constructs

If construct

IF expression THEN expression ELSE expression

Summation construct

SIGMA(input list | expression)

Dataset Statistics

Usual Syntax: STAT('dir/dataset', region, band, component)

NRBANDS() number of bands

NRROWS() number of rows

NRCOLS() number of columns

NRNULLS() number of null cells

NRNONNULLS() number of non-null cells

RMIN() region minimum

RMAX() region maximum

MEAN() mean

MEDIAN() median

COVAR() covariance

CORR() correlation

STDEV() standard deviation

STDEVNM1() standard deviation calculated with (n - 1)

MAXCELLX() maximum cell x coordinate (NRCOLS() - 1)

MAXCELLY() maximum cell y coordinate (NRROWS() - 1)

Usual Syntax: STAT(bandlist|'dir/dataset',region,band)

COV_EVAL() covariance eigenvalue

CORR_EVAL() correlation eigenvalue

PC_COV() covariance principal component (eigenvector) value

PC_CORR() correlation principal component (eigenvector) value

DETCOV() determinant of covariance matrix

DETCOR() determinant of correlation matrix

Symbolic Constants

PI p (3.14159...)

E e (2.71828...)

NULL value representing null or undefined cell

Special Constants

NRPIXROWS() number of pixel rows in the display (height)

NRPIXCOLS() number of pixel columns in the display (width)

MAXPIXELX()maximum pixel x coordinate (NRPIXCOLS() - 1)

MAXPIXELY()maximum pixel y coordinate (NRPIXROWS() - 1)

Examples

Ratio of two inputs: "INPUT1 / INPUT2"

Normalized diff veg ratio: "(I1 - I2) / (I1 + I2)"

Principal Component 1 (LandsatTM): "SIGMA(I1..I6 | I? * PC_COV(I1..I6 | , R1, I?, 1))"

Minimum Distance Classify: "-SQRT(SIGMA(I1,I2,I3,I4,I5,I6,I7|POW(I? - MEAN(,R1,I?),2)))"

Full Specifications

INPUTs

Syntax: INPUTn

In

Where: "n" is an integer ≥ 1

Examples: INPUT1, INPUT2, etc.

I1, I2, etc.

An input represents the data from a particular band of a dataset along with any pre-formula processing being done on the data (kernels or transforms).

Once the formula has been entered, the INPUTs are matched up to bands using the menus in the Band/Region Selection window pane.

In the majority of cases, INPUTs can be thought of as interchangeable with bands. However when pre-formula processing is being done, this is not the case. Note that the INPUT actually represents the band PLUS any preprocessing. Cases do occasionally arise where different inputs use the same band with different preprocessing (see the "SPOT_Sobel" algorithm for an example).

Note:A formula cannot have gaps in an sequential list of its INPUT numbers. This means that a formula like "(I3 - I1) / (I3 + I1)" would be invalid because there is an I1 and an I3, but no I2. The correct version of the formula would be:

"(I2 - I1) / (I2 + I1)".

REGIONS

Syntax: REGIONn

Rn

'text'

Where: "n" is an integer ≥ 1

"text" is a region name

Examples: REGION1, REGION2, etc.

R1, R2, etc.

'All', 'Ocean', 'Vegetation', etc.

ER Mapper gives you the ability to define regions of datasets by specifying one or more arbitrary polygons. Region specifications are used in formulas to specify certain region statistics, or within the INREGION() function.

"REGIONn" and "Rn" are generic forms of region specification which operate in a way similar to INPUTs. This region specification is not tied to a particular dataset, and the region menus in the Band/Region Selection window pane must be used to match up each generic region with one from the dataset being processed in the layer.

The 'text' form of region specification is absolute. It is not possible to change it in the Band/Region Selection window pane, and if the region does not exist in the dataset being processed, an error message will appear when you click on GO.

INPUT Lists

Input lists are used in the SIGMA construct, and the MAXLIKE function.

Syntax: Input range 1, Input range 2, Input range 3, ...

Where: An "Input range" can be of the form:

INPUTm..INPUTn (where $1 \leq m \leq n$)

Im..In (where $1 \leq m \leq n$)

INPUTn (where $n \geq 1$)

In (where $n \geq 1$)

Examples: (The following are all equivalent)

"INPUT1, INPUT2, INPUT3, INPUT4"

"I1, I2, I3, I4"

"INPUT1..INPUT4"

"I1..I4"

"INPUT1..INPUT3, INPUT4"

"I1..I3, I4"

Bands

Bands are used when specifying certain dataset statistics.

Syntax: INPUT

BANDn

Bn

`desc'

Where: INPUT is a standard INPUT specification (see section on INPUTs).

`n' is an integer ≥ 1 .

`desc' is a band description.

Examples: INPUT1, I1

BAND1, BAND2, BAND3, etc.

B1, B2, B3, etc.

`0.485_um', `Residual Gravity um/sec2', `magnetics_nanoTesla'

The INPUT form

The most common form of band specification is the INPUT. In this case the band associated with the INPUT is used.

The BANDn and Bn forms

These forms are used to specify an absolute band number. For example, BAND1 and B1 correspond to band number 1 in the dataset. If the band number is invalid, an error message will appear after you click on GO.

The `desc' form

This form uses the band description to specify the band to use. If the band does not exist in the dataset, an error message appears after you click on GO.

Band Lists

Band lists are used with the following region statistics: COV_EVAL, CORR_EVAL, PC_COV, PC_CORR, DETCOV, and DETCOR.

Syntax: Band range1, Band range 2, Band range3, ...

Where: A band range can be a input list (see section on Input Lists) or of the form:

BANDm..BANDn (where $1 \leq m \leq n$)

Bm..Bn (where $1 \leq m \leq n$)

Band (see section on Bands)

Examples: I1..I3

I1, I2, I3

B1, B2, B3

B1..B3, B5

`0.485_um', `0.56_um', `0.66_um'

BAND1, B2, I1, B6..B10, `0.485_um'

Operators

Arithmetic Operators

* Multiplication "I1 * I2"

/Division "I1 / I2"

+Addition "I1 + I2"

- (binary) Subtraction "I1 - I2"

- (unary) Negation "-I1", "-(I1 + I2)"

% Modulus "I1 % 10"

Relational and Logical Operators

Relational and logical operators perform some logical operation and return either 1 or 0 representing TRUE or FALSE. The result of these operations may be used in an IF construct, or directly as part of the formula.

= Is equal to "IF (I1 == 0) THEN NULL ELSE I1"

!= Is not equal to "IF (I1 != 0) THEN I1 ELSE NULL"

> Is greater than "IF (I1 > I2) THEN 1 ELSE 2"

>= Is greater than or equal to "IF (I1 >= I2) THEN 1 ELSE 2"

< Is less than "IF (I1 < I2) THEN 2 ELSE 1"

<= Is less than or equal to "IF (I1 <= I2) THEN 2 ELSE 1"

AND Logical and "If (I1 > 0 AND I1 < 10) THEN I1 ELSE NULL"

OR Logical or "If (I1 < 0 OR I1 > 10) THEN NULL ELSE I1"

NOT logical negation "IF (NOT I1 > I2) THEN 2 ELSE 1"

Operator Precedence

The following shows the relative precedence of the various operators in the order of decreasing precedence. Operators on the same line have equal precedence.

NOT - (unary)

* / %

+ - (binary)

< <= > >=

= !=

AND

OR

Example:

In the expression "I1 OR I2 AND I3", the AND would be evaluated first because it has a greater precedence than OR. The expression is equivalent to "I1 OR (I2 AND I3)". If you wanted the OR to be evaluated first, you would have to use brackets: "(I1 OR I3) AND I3".

Note: Brackets are used to explicitly describe the order in which sub-expressions are to be evaluated. It is recommended they be used in cases where they are not strictly necessary, but where they improve the readability of the formula.

Standard Functions

Syntax: function(expression) or

function(expression1, expression2) or

function(expression1, expression2, expression3, ...)

(depending on the function)

ABS(x) absolute value of x

SIN(x) sine of x (radians)

COS(x) cosine of x (radians)

TAN(x) tangent of x (radians)

ASIN(x) arc sine of x (radians)

ACOS(x) arc cosine of x (radians)

ATAN(x) arc tangent of x (radians)

EXP(x) e to the power of x (e^x)

LOG(x) natural log of x

LOG10(x) log to base 10 of x

POW(x,y) x to the power of y (x^y)

SQRT(x) square root of x

MIN(x,y,...) minimum of x, y, ...

MAX(x,y,...) maximum of x, y, ...

FLOOR(x) greatest integer less than or equal to x

CEIL(x) smallest integer greater than or equal to x

Arguments to functions can be any valid expression.

Examples: abs(I1)

abs(-1)

abs(I1 * I2)

abs(sin(I1))

abs(IF I1 > I2 THEN I1 ELSE I2)

The trigonometric functions, for example SIN(x), require "x" to be in radians. The conversion for degrees to radians is: angle (radians) = angle (degrees) * (p/180).

Special Functions

INREGION()

Syntax: INREGION(region)

Where: "region" is a standard region specification. (See section on REGIONS).

Examples: INREGION(R1)

INREGION(REGION1)

INREGION('Forest')

The INREGION() function returns 1 if the current cell being processed is within the specified region, otherwise it returns 0. A typical formula might look like:

"IF INREGION(R1) THEN I1 ELSE NULL"

The ISNULL() Function

Syntax: ISNULL()

ISNULL() returns 1 if the cell is null, otherwise it returns 0.

To test for nulls, use ISNULL(). For example, suppose you have an algorithm with two Pseudocolor layers. The two layers are identical except that the first layer includes a large average filter. On its own, the averaging filter tends to shrink edges of an image and substitute nulls. To maintain the size of the image, you can use:

IF ISNULL(i1) THEN i2 ELSE i1

The CELLX() and CELLY() Functions

Syntax: CELLX()

CELLY()

CELLX() and CELLY() return the x and y (column and row) coordinates of the current cell being processed in the formula.

The coordinates begin with zero. For example in a dataset 100 cells wide by 200 cells high, the x coordinates would be from 0-99 and the y coordinates would be from 0-199).

A formula only processes non-null cells (cells for which a known value exists). This means that coordinates for null cells or for cells outside the dataset will never be encountered.

The MAHAL() Function

Syntax: MAHAL(region, input list)

Where: "region" is a standard region specification. See section on REGIONS.

"input list" is a standard input list. (See section on Input Lists).

Examples: MAHAL('Urban', I1, I2, I3)

MAHAL(R1, I1..I6)

MAHAL(REGION2, I1, I2..I10)

MAHAL is the Mahalanobis classification discriminant function. This can be used as a classifier on its own, or as part of the formula for the Maximum Likelihood discriminant function.

The MAHAL function computes the following:

$\text{transpose}(x - m) * \text{covmat} * (x - m)$

Where: covmat -> covariance matrix

x -> input vector

m -> mean vector

(from Richards, J. A., 1986, Remote Sensing Digital Analysis: An Introduction, Springer-Verlag, Berlin. pp. 185-186).

The mean vector and covariance matrix are extracted from the 'region' statistics for the bands associated with each of the INPUTs.

The PIXELX() and PIXELY() Functions

Syntax: PIXELX()

PIXELY()

PIXELX() and PIXELY() return the x and y (column and row) coordinates of the first pixel where the current cell being processed will appear.

If the image is not being supersampled there will be a 1:1 cell:pixel correspondence, otherwise the PIXELX() and PIXELY() values will be replicated in the supersampling.

For example, if supersampling 1 to 2 (i.e. if the window is twice the size of the dataset), PIXELX() would return:

0 0 2 2 4 4 6 6...

instead of

0 1 2 3 4 5 6 7...

Note: Generally when using PIXELX() and PIXELY() in a formula, you will only cover the area where the dataset appears - not necessarily the whole window.

For example, if you have a dataset that only appears in a portion of the window (say from top left (200,250) to bottom right (300,350) of a 400x400 window, then you will only get PIXELX() values from 200-300 and PIXELY() values from 250-350.

In special cases where the formula does not contain any INPUTs, INREGION()s, CELLX()s, or CELLY()s, the overlay dataset is ignored for purposes of registration, and all pixels are processed within the formula.

Special Constructs

The IF THEN ELSE Expression

Syntax: IF expression THEN expression ELSE expression

Examples: IF (I1 > I2) THEN 1 ELSE 2

IF (I1 == 0) THEN NULL ELSE I1

The IF THEN ELSE expression is evaluated as follows:

If the "IF" expression is non zero, then the value of the expression will be the THEN expression, otherwise it will be the ELSE expression.

Examples: The value of "IF 1 THEN 2 ELSE 3" would be 2.

The value of "IF 0 THEN 2 ELSE 3" would be 3.

IF THEN ELSE expressions can be nested. For example,

IF I1 THEN I1 ELSE (IF I2 THEN I2 ELSE 0)

Use brackets to clarify order of evaluation. In particular, be aware of the following:

"IF 1 THEN 2 ELSE 5 + 1" = 2 is equivalent to:

"IF 1 THEN 2 ELSE (5 + 1) = 2 not equivalent to:

"(IF 1 THEN 2 ELSE 5) + 1" = 3

The Sigma (Summation) Construct

Syntax: SIGMA(input list | expression)

Examples: "SIGMA(I1, I2, I3 | I?)"

"SIGMA(I1..I7 | I? - MEAN(R1,I?))"

The SIGMA construct is evaluated as follows:

For each input in the input list, all occurrences of I? are replaced in the sigma expression. The resulting expression is evaluated and added to the total.

For example:

"SIGMA(I1,I2,I3 | I? - MEAN(R1,I?))" is equivalent to

"(I1 - MEAN(R1,I1)) + (I2 - MEAN(R1,I2)) + (I3 - MEAN(R1,I3))"

The input list may contain the following structure: "Im..In". This is equivalent to "Im, I(m+1), I(m+2), ... In".

Examples: "I1..I3" is the same as "I1, I2, I3".

"I1..I3,I5" is the same as "I1, I2, I3, I5"

Dataset and Region Statistics

The general form for a dataset or region statistic in a formula is

STAT(dataset, region, band, component)

Where:

"STAT" is the statistic name.

"dataset" is text specifying the dataset directory and file name separated by a slash (/). (For example, `ers/Australia_DTM.ers'). The default is the overlay dataset. Alternatively, specifying D1, D2 etc. lets you specify the specific dataset using the Datasets pane.

"region" is a standard REGION specification (see section on REGIONS). The default is the `All' region.

"band" is a standard band specification (see Bands). The default is band 1.

"component" is an integer ≥ 1 and \leq the number of bands in the dataset.

If one of the arguments is not specified, it is set to a default value. Commas MUST be entered if preceding arguments are not specified. For example:

STAT(, region, band, 1) - dataset defaults to overlay dataset.

STAT(,band, 1) - dataset defaults to overlay dataset and region defaults to `All'.

STAT(,, 1) - dataset defaults to overlay dataset, region defaults to `All', and band defaults to band 1.

NRBANDS()

The number of bands in the dataset.

Syntax: NRBANDS(dataset)

Examples: NRBANDS()

NRBANDS(`ers/LandsatTM_Path90Row86E.ers')

NRROWS()

The number of rows in the dataset.

Syntax: NRROWS(dataset)

Examples: NRROWS()

NRROWS(`ers/LandsatTM_Path90Row86E.ers')

NRCOLS()

The number of columns in the dataset.

Syntax: NRCOLS(dataset)

Examples: NRCOLS()

NRCOLS(`ers/LandsatTM_Path90Row86E.ers')

MAXCELLX()

The maximum cell x coordinate. The coordinates begin at zero, so this is the same as (NRCOLS() - 1).

Syntax: MAXCELLX(dataset)

Example: MAXCELLX(`ers/LandsatTM_Path90Row86E.ers')

MAXCELLY()

The maximum cell y coordinate. The coordinates begin at zero, so this is the same as (NRROWS() - 1).

Syntax: MAXCELLY()

Example: MAXCELLY(`ers/LandsatTM_Path90Row86E.ers')

NRNULLS()

The number of null cells in a particular band within a region.

Syntax: NRNULLS(dataset, region, band)

Examples: NRNULLS()

NRNULLS(`ers/LandsatTM_Path90Row86E.ers', `Ocean', B1)

NRNULLS(R1,I1)

NRNONNULLS()

The number of non null cells in a particular band within a region.

Syntax: NRNONNULLS(dataset, region, band)

Examples: NRNONNULLS()

NRNONNULLS(`ers/Australia_DTM.ers', `All', B1)

NRNONNULLS(R1,I1)

RMIN()

The minimum value for a particular band within a region.

Syntax: RMIN(dataset, region, band)

Examples: RMIN()

RMIN(`ers/Australia_DTM.ers', `All', B1)

RMIN(R1,I1)

RMAX()

The maximum value for a particular band within a region.

Syntax: RMAX(dataset, region, band)

Examples: RMAX()

RMAX(`ers/Australia_DTM.ers', `All', B1)

RMAX(R1,I1)

MEAN()

The mean value for a particular band within a region.

Syntax: MEAN(dataset, region, band)

Examples: MEAN()

MEAN(`ers/Australia_DTM.ers', `All', B1)

MEAN(R1,I1)

MEDIAN()

The median value for a particular band within a region.

Syntax: MEDIAN(dataset, region, band)

Examples: MEDIAN()

MEDIAN(`ers/Australia_DTM.ers', `All', B1)

MEDIAN(R1,I1)

STDEV()

The standard deviation for a particular band within a region (The sum of squares of distances from the mean divided by the number of cells).

Syntax: STDEV(dataset, region, band)

Examples: STDEV()

STDEV(`ers/Australia_DTM.ers', `All', B1)

STDEV(R1,I1)

STDEVNM1()

The standard deviation for a particular band within a region calculated with n - 1 (The sum of squares of distances from the mean divided by the number of cells minus one).

Syntax:STDEVNM1(dataset, region, band)

Examples: STDEVNM1()

STDEVNM1(`ers/Australia_DTM.ers', `All', B1)

STDEVNM1(R1,I1)

COVAR()

The region covariance matrix value for a particular band and component number.

Syntax: COVAR(dataset, region, band, component)

Examples: COVAR(`ers/SPOT_XS_Perth.ers', `All', B1, 1)

COVAR(R1,I1,1)

CORR()

The region correlation matrix value for a particular band and component number.

Syntax: CORR(dataset, region, band, component)

Examples: CORR(`ers/SPOT_XS_Perth.ers', `All', B1, 1)

CORR(R1,I1,1)

Dataset and Region Statistics with Band Lists

Certain statistics follow the same basic structure as the standard 'Dataset and Region Statistic', but have the added ability of specifying the bands to use to calculate the statistic.

The general form is:

STAT(band list | dataset, region, band, component)

Where: "band list" is a standard band list (see the Band List section). The band list and vertical bar are optional and may be omitted. If this is done, all bands are used.

"dataset", "region", "band", and "component" are all as for dataset and region statistics.

PC_COV()

The region covariance principal component (eigenvector) matrix value for a particular band and component number.

Syntax: PC_COV(band list | dataset, region, band, component)

Examples: PC_COV(B1..B3 | `ers/SPOT_XS_Perth.ers', `All', B1, 1)

PC_COV(I1..I6 | , R1, I1, 1)

PC_COV(R1,I1,1)

PC_CORR()

The region correlation principal component (eigenvector) matrix value for a particular band and component number.

Syntax: PC_CORR(band list | dataset, region, band, component)

Examples: PC_CORR(B1..B3 | `ers/SPOT_XS_Perth.ers', `All', B1, 1)

PC_CORR(I1..I6 | , R1, I1, 1)

PC_CORR(,R1,I1,1)

COV_EVAL()

The region covariance matrix eigenvalue for a particular band.

Syntax: COV_EVAL(band list | dataset, region, band)

Examples:COV_EVAL()

COV_EVAL(B1..B3 | `ers/SPOT_XS_Perth.ers', `All', B1)

COV_EVAL(I1..I6 | , R1, I1)

COV_EVAL(,R1,I1)

CORR_EVAL()

The region correlation matrix eigenvalue for a particular band.

Syntax: CORR_EVAL(band list | dataset, region, band)

Examples:CORR_EVAL()

CORR_EVAL(B1..B3 | `ers/SPOT_XS_Perth.ers', `All', B1)

CORR_EVAL(I1..I6 | , R1, I1)

CORR_EVAL(,R1,I1)

DETCOV()

The determinant of the region covariance matrix for a particular set of bands.

Syntax: DETCOV(band list | dataset, region)

Examples: DETCOV()

DETCOV(B1..B3 | `ers/SPOT_XS_Perth.ers', `All')

DETCOV(I1..I6 | , R1)

DETCOV(,R1)

DETCORR()

The determinant of the region correlation matrix for a particular set of bands.

Syntax: DETCORR(band list | dataset, region)

Examples: DETCORR()

DETCORR(B1..B3 | `ers/SPOT_XS_Perth.ers', `All')

DETCORR(I1..I6 | , R1)

DETCORR(,R1)

The band list form is sometimes useful when dealing with LandsatTM data to exclude band 6 from the principal components calculation.

The first principal component formula for Landsat TM excluding band 6 is as follows:

"SIGMA(I1..I6 | I? * PC_COV(I1..I6 | ,R1, I?, 1)"

(with inputs I1->I6 matched to bands 1-5, and 7)

Numeric Constants

Numeric constants may be used in formulas.

Examples:"I1 + 1"

"I1 + I2 / 2"

"I1 + I2 * 0.5"

Symbolic Constants

Certain symbolic constants are recognised and translated into their corresponding numerical values. These are:

"PI" 3.14159...

"E" 2.71828...

NULL

The special NULL keyword is available for use in formula. It is a special constant used to set a value to NULL, especially in THEN or ELSE parts of an IF statement.

For example:

IF ISNULL(i1) THEN 1 ELSE NULL

This highlights areas of null data.

Special Constants

These statistics provide information related to the size (height and width) of the displayed image in pixels.

NRPIXROWS()

The number of rows in the display.

Syntax: NRPIXROWS()

Example: NRPIXROWS().

NRPIXCOLS()

The number of columns in the display.

Syntax: NRPIXCOLS()

Example: NRPIXCOLS()

MAXPIXELX()

The maximum pixel x coordinate. The coordinates begin at zero, so this is the same as (NRPIXCOLS() - 1).

Syntax: MAXPIXELX()

Example: MAXPIXELX()

MAXPIXELY()

The maximum pixel y coordinate. The coordinates begin at zero, so this is the same as (NRPIXROWS() - 1).

Syntax: MAXPIXELY()

Example: MAXPIXELY()

Ratio and Principal Component Analysis

This chapter shows you how to use formula in batch scripting. It describes how to apply formula in batch scripting such as a simple ratio to a more complex formula of Principal Component Analysis.

Hands-on exercises

These exercises teach you how to apply formula such as rationing and Principal Component Analysis in batch scripting.

What you will learn...

After completing these exercises, you will know how to use formula in batch scripting and create ratio and Principal Components RGB composite images.

- Select a dataset.
- Set the algorithm mode to pseudo and rgb.
- Select bands for ratios.
- Set up the ratio formula.
- Select number of bands for Principal Component Analysis
- Set up the Principal Component Analysis formula.
- Enhance transforms of the ratio and the principal components

- Copy the algorithms (ratio & RGB composite of PCA) to ER Mapper and display the processed images

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

1: Setting up the batch script button

Note: To run a batch script from a button, the batch script file has to be included in the toolbar file. Hence, add the name of the batch script you are going to create in the “test1.bar” toolbar file.

- 1 Edit the “test1.bar” file and add a new line “**JS_Ratio_16x16**” “**Example BATCH_Ratio**” **BATCH “JS_Ratio”**

Note: You have added “JS_Ratio” batch script file which will refer to the “JS_Ratio.erb” batch script file in the ERMAPPER/batch directory. You will create the “JS_Ratio.erb” batch script file below.

Create an image Tiff file to be used as the picture in the icon of the batch script button

Process an image that you would like to use it as the picture in the icon of your batch script button.

Tip: You can use an exiting Tiff file from the ERMAPPER\icon directory

- 1 Create a band ratio image (TM3/TM2) using a Landsat Thematic Mapper image from the ‘Landsat_TM_year_1985.ers’ dataset.

Note: The ‘Landsat_TM_year_1985.ers’ dataset is in the ERMAPPER\dataset\Core_Datasets directory.

- 2 Follow the procedure in chapter 3 and create a 24-bit 16x 16 pixels “JS_Ratio_16x16.tif” tiff file. Place the tiff file in the ‘ERMAPPER/icons’ directory.

Write a batch script to process and display a band ratio image.

Logic:

- To write a batch script that process a band ratio image you need to select a multi-band dataset.
- Select two bands to process a band ratio image.
- Set the colormode to pseudo and select a color from the Color Lookup Table
- Set up an option for smoothing
- Process the ratio algorithm
- Copy the algorithm to window and display the ratio image.

Processing

- Include “lib/BE_Startup.erb” and set up the wizard
- Use ask begin/end block
- Set the formula for Band Ratio
- Select two bands to process band ratio.
- A section to process a Band Ratio image.

Tip: In the batch script setting the formula must precede setting the bands for Input 1 and Input 2 of the layer.

- Copy the algorithm to window and display the Band Ratio image

Writing the batch script

Type the following batch script in a Text Editor.

1:Band ratio

```
#####
#
# Copyright 1997 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_Ratio.erb
```


Chapter 10 Ratio and Principal Component Analysis ● 1:Band ratio

```
# Date:      4th August 1997
# Author:    Abdullah Mah
#
# Summary:   Applying Ratio formula in batch scripting
#
# Details:
#           1) Use ask file command to import a dataset
#           2) Use ask bandmenu command to select two bands for the ratio
#           3) Set the algorithm mode to pseudo
#           4) Use ask lutmenu to select a color from CLUT
#           5) Use ask yesno command for smoothing option
#           6) Set the layer
#           7) set the formula in the layer
#           8) set the input 1 and input 2 in the layer for the formula
#           9) Enhance the transform of the ratioed image
#           10) Copy algorithm to window and display the ratio image
#
include "lib/BE_Startup.erb"
ask begintitle "User Input Dataset for Ratio"
    say "Please select a dataset for Ratio"
    ask file "Dataset1:" ".ers" $dataset1_input
    say ""
    say "Select two bands for ratio"
    ask bandmenu "Select a band for Input 1" $dataset1_input $band1
    ask bandmenu "Select a band for Input 2" $dataset1_input $band2
    say ""
    labels_above
    ask lutmenu "Color Lookup Table" $lut
    say ""
    ask yesno "Smoothing " $yesSmoothing
ask end
new window
    first surface
    new algorithm
    set algorithm description "Pseudocolor"
    set algorithm mode pseudo
    set layer description "Pseudo"
    set layer dataset to $dataset1_input
    set layer formula to "i1/i2"
    set layer input 1 to band $band1
    set layer input 2 to band $band2
    set surface lut to $lut
    copy algorithm to window
```

```

go window
copy algorithm from window
first pseudo layer
first transform
set transform limits to actual
copy algorithm to window
go window
copy algorithm from window
first pseudo layer
first transform
set transform clip to 95
if ($yesSmoothing == 1) then goto smoothing else goto no_smoothing
smoothing:
    set algorithm supersample type to bilinear
no_smoothing:
    include "lib/Delete_All_Inactive_Layers.erb"
    copy algorithm to window
    go window
    say status "Done.\n"
    delete status
exit
#####

```

Displaying Band Ratio image

- 1 Save the batch script file as “JS_Ratio.erb” in the ‘ERMAPPER/batch’ directory
- 2 Exit the ER Mapper and call it again so that it will recognize your test1.bar toolbar file in the ‘ERMAPPER/config’ directory
- 3 On the ER Mapper main menu click on the toolbar menu. Your “test1.bar” will appear as “test1” in the Toolbar menu in alphabetical order.

Tip: Your batch script file “**JS_Ratio.erb**” in the ‘ERMAPPER/batch’ directory should have the same name as the batch script (“JS_Ratio_16x16” “Example BATCH Ratio” BATCH “**JS_Ratio**”) in the “test1.bar” toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 4 Click the “test1” from the dropdown list of the Toolbar menu.

- 5 From the displayed buttons click on the JS_Ratio batch script button.
- 6 The **User Input Dataset for Ratio** dialog box appears.
- 7 On the **User Input Dataset for Ratio** dialog box select “**Landsat_TM_year_1985.ers**” dataset from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 8 Select **B3:0.66_um** and **B1:0.485_um** of the **Landsat_TM_year_1985.ers**

Note: Band ratio TM3/TM1 highlights Fe rich sediments

- 9 Select Pseudocolor from the Color Lookup Table.
- 10 Select Smoothing option.
- 11 Click OK on the **User Input Dataset for Ratio - Image Properties** dialog box.
- 12 The band ratio image is displayed

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

1: Setting up the batch script button for PCA

Note: To run a batch script from a button, the batch script file has to be included in the toolbar file. Hence, add the name of the wizard script you are going to create in the “test1.bar” toolbar file.

- 1 Edit the “test1.bar” file and add a new line “**JS_PCformula_16x16**”
“**Example BATCH_PCformula**” BATCH “**JS_PCformula**”

Note: You have added “JS_PCformula” batch script file which will refer to the “JS_PCformula.erb” batch script file in the ‘ERMAPPER/batch’ directory. You will create the “JS_PCformula.erb” batch script file below.

Create an image Tiff file to be used as the picture in the icon of the batch script button

Process a tiff image to use it as the picture in the icon of your batch script button.

Tip: You can use an exiting Tiff file from the ERMAPPER\icon directory

- 1 Create a RGB composite with PC1(Red), PC2(Green) and PC3(Blue) image using the 'Landsat_TM_year_1985.ers' dataset.

Note: The 'Landsat_TM_year_1985.ers' dataset is in the ERMAPPER\dataset\Core_Datasets directory.

- 2 Follow the procedure in chapter 3 and create a 24-bit 16x 16 pixels "JS_PCformula_16x16.tif" tiff file. Place the tiff file in the 'ERMAPPER/icons' directory.

Write a batch script to process and display a RGB composite with PC1(Red), PC2(Green) and PC3(Blue) image.

Logic:

- To write a batch script that process RGB composite with PC1(Red), PC2(Green) and PC3(Blue) you need to select a multi-band dataset.
- Set up an option to select number of bands to be used in the Principal Component Analysis (PCA)
- Select rgb color mode
- Set up an option for smoothing
- Process the RGB composite algorithm with PC1(Red), PC2(Green) and PC3(Blue)
- Copy the algorithm to window and display it

Processing

- Include "lib/BE_Startup.erb" and set up the wizard
- Use ask begin/end block
- Set the formula for Principal Component Analysis
- A section to process a RGB composite image with PC1(Red), PC2(Green) and PC3(Blue).

Tip: In the batch script, setting the formula must precede setting the bands for Input 1, Input 2, Input 3 etc. for the principal component of the layer.

- Copy the algorithm to window and display the PCA image

Writing the batch script

Type the following batch script in a Text Editor.

2:Principal Component Analysis

```
#####

#
# Copyright 1997 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_formula_PC.erb
# Date:     4th August 1997
# Author:   Abdullah Mah
#
# Summary:  Applying Principal Component Analysis formula in batch scripting
#
# Details:
#   1) Set array index for 3/4/5/6 bands PCA options
#   2) Set PCA formulas for 3/4/5/6 bands PCA options
#   3) Use ask file command to import a dataset
#   4) Use ask listmenu (dropdown list) command to select 3/4/5/6 bands PCA
#   5) Set the algorithm mode to rgb
#   6) Use ask yesno command for smoothing option
#   7) Set the red/green/blue layers for 3/4/5/6 bands PCA options
#   8) Set the algorithm description for 3/4/5/6 bands PCA options
#   9) set the formulas for 3/4/5/6 bands PCA in the layers
#  10) Enhance the red/green/blue layers transforms of the PCA image
#  11) Copy algorithm to window and display the PCA image
#
include "lib/BE_Startup.erb"
$list_menu["0"] = "3-Bands"
$list_menu["1"] = "4-Bands"
$list_menu["2"] = "5-Bands"
$list_menu["3"] = "6-Bands"
$band1 = 1
$band2 = 2
$band3 = 3
```

```

$band4 = 4
$band5 = 5
$band6 = 7
$list_choice = $list_menu["0"]
$formula_3PC1 = "SIGMA(I1..I3|I? * PC_COV(I1..I3|,R1,I?,1))"
$formula_3PC2 = "SIGMA(I1..I3|I? * PC_COV(I1..I3|,R1,I?,2))"
$formula_3PC3 = "SIGMA(I1..I3|I? * PC_COV(I1..I3|,R1,I?,3))"
$formula_4PC1 = "SIGMA(I1..I4|I? * PC_COV(I1..I4|,R1,I?,1))"
$formula_4PC2 = "SIGMA(I1..I4|I? * PC_COV(I1..I4|,R1,I?,2))"
$formula_4PC3 = "SIGMA(I1..I4|I? * PC_COV(I1..I4|,R1,I?,3))"
$formula_5PC1 = "SIGMA(I1..I5|I? * PC_COV(I1..I5|,R1,I?,1))"
$formula_5PC2 = "SIGMA(I1..I5|I? * PC_COV(I1..I5|,R1,I?,2))"
$formula_5PC3 = "SIGMA(I1..I5|I? * PC_COV(I1..I5|,R1,I?,3))"
$formula_6PC1 = "SIGMA(I1..I6|I? * PC_COV(I1..I6|,R1,I?,1))"
$formula_6PC2 = "SIGMA(I1..I6|I? * PC_COV(I1..I6|,R1,I?,2))"
$formula_6PC3 = "SIGMA(I1..I6|I? * PC_COV(I1..I6|,R1,I?,3))"
ask begintitle "User Input Dataset for PC123 RGB composite"
    say "Please select a dataset"
    ask file "Dataset:" ".ers" $dataset_input
    say "Choose number of bands to create PC1, PC2, PC3"
    ask listmenu "No. of bands" "" $list_menu $list_choice
    say ""
    ask yesno "Smoothing " $yesSmoothing
ask end

new window
new algorithm
set algorithm mode rgb
add red layer
set layer description "red"
set layer dataset to $dataset_input
if ($list_choice == "3-Bands") then goto Rbands3 else goto Rcont1
Rbands3:
set algorithm description "PCA R(PC1)G(PC2)B(PC3) Composite - 3 bands"
set layer formula to $formula_3PC1
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
goto greenlayer
Rcont1:
if $list_choice == "4-Bands" then goto Rbands4 else goto Rcont2
Rbands4:
set algorithm description "PCA R(PC1)G(PC2)B(PC3) Composite - 4 bands"
set layer formula to $formula_4PC1

```

```

set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
goto greenlayer
Rcont2:
if $list_choice == "5-Bands" then goto Rbands5 else goto Rcont3
Rbands5:
set algorithm description "PCA R(PC1)G(PC2)B(PC3) Composite - 5 bands"
set layer formula to $formula_5PC1
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
goto greenlayer
Rcont3:
if $list_choice == "6-Bands" then goto Rbands6 else goto Rcont4
Rbands6:
set algorithm description "PCA R(PC1)G(PC2)B(PC3) Composite - 6 bands"
set layer formula to $formula_6PC1
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
set layer input 6 to band $band6
goto greenlayer
Rcont4:
# Green layer for PC2
greenlayer:
add green layer
set layer description "green"
set layer dataset to $dataset_input
if $list_choice == "3-Bands" then goto G3bands else goto Gcont1
G3bands:
set layer formula to $formula_3PC2
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
goto bluelayer
Gcont1:
if $list_choice == "4-Bands" then goto G4bands else goto Gcont2

```

```

G4bands:
set layer formula to $formula_4PC2
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
goto bluelaye
Gcont2:
if $list_choice == "5-Bands" then goto G5bands else goto Gcont3
G5bands:
set layer formula to $formula_5PC2
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
goto bluelaye
Gcont3:
if $list_choice == "6-Bands" then goto G6bands else goto Gcont4
G6bands:
set layer formula to $formula_6PC2
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
set layer input 6 to band $band6
goto bluelaye
Gcont4:
# Blue layer for PC3
bluelaye:
add blue layer
set layer description "green"
set layer dataset to $dataset_input
if $list_choice == "3-Bands" then goto B3bands else goto Bcont1
B3bands:
set layer formula to $formula_3PC3
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
goto complete3layers
Bcont1:
if $list_choice == "4-Bands" then goto B4bands else goto Bcont2

```



```

B4bands:
set layer formula to $formula_4PC3
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
goto complete3layers
Bcont2:
if $list_choice == "5-Bands" then goto B5bands else goto Bcont3
B5bands:
set layer formula to $formula_5PC3
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
goto complete3layers
Bcont3:
if $list_choice == "6-Bands" then goto B6bands else goto Bcont4
B6bands:
set layer formula to $formula_6PC3
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
set layer input 6 to band $band6
goto complete3layers
Bcont4:
complete3layers:
    copy algorithm to window
    go window
    copy algorithm from window
# Cycle through all layers setting limits to actual data limits
    first active raster layer
next_active_layer:
    last transform
    set transform limits to actual
    next active raster layer
    if ($ERROR ==0) then goto next_active_layer
    copy algorithm to window
    go window
    copy algorithm from window

```

```

# Cycle through all layers setting the transform clip to 95%
    first active raster layer
next_active_layer:
    last transform
    set transform clip to 95
    next active raster layer
    if ($ERROR ==0) then goto next_active_layer
    if ($yesSmoothing == 1) then goto smoothing else goto no_smoothing
smoothing:
    set algorithm supersample type to bilinear
no_smoothing:
    include "lib/Delete_All_Inactive_Layers.erb"
    copy algorithm to window
    go window
exit
#####

```

Displaying PC123 RGB composite image

- 1 Save the batch script file as “JS_PCformula.erb” in the ‘ERMAPPER/batch’ directory
- 2 Exit the ER Mapper and call it again so that it will recognize your test1.bar toolbar file in the ‘ERMAPPER/config’ directory
- 3 On the ER Mapper main menu click on the toolbar menu. Your “test1.bar” will appear as “test1” in the Toolbar menu in alphabetical order.

Tip: Your batch script file “**JS_PCformula.erb**” in the ‘ERMAPPER/batch’ directory should have the same name as the batch script (“JS_PCformula_16x16” “Example BATCH Ratio” BATCH “**JS_PCformula**”) in the “test1.bar” toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 4 Click the “test1” from the dropdown list of the Toolbar menu.
- 5 From the displayed buttons click on the JS_PCformula batch script button.
- 6 The **User Input Dataset for PC123 RGB composite** dialog box appears.
- 7 On the **User Input Dataset for PC123 RGB composite** dialog box select “**Newcastle_Radiometrics.ers**” dataset from the ERMAPPER/dataset/Core_Datasets directory.

- 8 Select **4-bands** as the number of bands from which you would like to generate Principal Components
- 9 Select Smoothing option.
- 10 Click OK on the **User Input Dataset for PC123 RGB composite** dialog box.
- 11 The PC1(Red), PC2(Green) and PC3(Blue) RGB composite image is displayed

Exercises:

(1) Write a batch script to process a ratio image. In the batch script select a multi-band raster dataset, select two bands from the dataset, set up the ratio formula, process and display the ratio image. Set the colormode to pseudo, choose pseudocolor from the lookuptable and gaussian equalize transform to enhance the ratio image.

(2) Write a batch script to process a RGB composite with PC2 in the Red layer, PC3 in the Green layer and PC4 in the Blue layer. In the batch script select a raster dataset with 6 or more bands, select six bands from the dataset, set up the PC2, PC3 and PC4 formulas, process and display it as a RGB (PC2-Red, PC3-Green, PC4-Blue) composite image. Enhance the transforms of the red/green/blue layers individually with clip to 98%. Use \$ERROR variable as a control to loop through the three layers. Delete the inactive layer/s before displaying the processed image.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:
 - For Windows, select **Close** from the window control-menu.
 - For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to perform the following tasks in ER Mapper:

- Create a batch script to process a ratio image.
- Create a batch script to process RGB composite image with PC1 (Red), PC2 (Green) and PC3 (Blue).

Wizard Ratio, RGB-ratio and PCA

This chapter describes how to apply formula in wizard scripting such as band ratio, RGB ratio composite and Principal Component Analysis.

Hands-on exercises

These exercises teach you how to apply formula such as band rationing, RGB ratio composite and Principal Component Analysis in wizard scripting.

What you will learn...

After completing these exercises, you will know how to use formula in wizard scripting and create ratio, RGB ratio composite and Principal Components RGB composite images.

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

1: Setting up the wizard button

Note: To run a batch script from a wizard button, the batch script file has to be included in the toolbar file. Hence, add the name of the wizard script you are going to create in the “test1.bar” toolbar file.

- 1 Edit the “test1.bar” file and add a new line
“**JS_wizard_Ratio_RGBratio_PC_16x16**” “**Example Wizard_Ratio_RGBratio_PC**” **BATCH**
“**JS_wizard_Ratio_RGBratio_PC**”

Note: You have added “JS_wizard_Ratio_RGBratio_PC” batch script file which will refer to the “JS_wizard_Ratio_RGBratio_PC.erb” batch script file in the ‘ERMAPPER/batch’ directory. You will create the “JS_wizard_Ratio_RGBratio_PC.erb” wizard script file below.

Create an image Tiff file to be used as the picture in the icon of the wizard button

Process an image that you would like to use it as the picture in the icon of your batch script wizard button.

Tip: You can use an exiting Tiff file from the ERMAPPER\icon directory

- 1 Create a RGB image of PC1(Red), PC2(Green) and PC3(Blue) using a Landsat Thematic Mapper image from the ‘Landsat_TM_year_1985.ers’ dataset.

Note: The ‘Landsat_TM_year_1985.ers’ dataset is in the ERMAPPER\dataset\Core_Datasets directory.

- 2 Follow the procedure in chapter 3 and create a 24-bit 16x 16 pixels “JS_wizard_Ratio_RGBratio_PC_16x16.tif” tiff file. Place the tiff file in the ‘ERMAPPER/icons’ directory.

Note: To use a tiff file of a size that will fit the specified area on the wizard you can create a tiff file using other software and import it into ER Mapper.

Write a wizard script to process and display a bandratio, a RGB ratio and a PCA images.

Logic:

- To write a wizard that process band ratio, RGB ratio composite and RGB composite of three Principal Components you need to select a multi-band dataset.
- Provide options to choose one of the three processing options (radio buttons) to process the data (Wizard page 1)
- Provide an option to smooth the processed image. (Wizard page 1)
- Select two bands to process band ratio. (Wizard pages 2)
- Select six bands to process RGB ratio composite. (Wizard pages 3)
- Select the number of bands for Principal Component Analysis (Wizard page 4)
- Process the image/s
- Copy the algorithm/s to window/s and display them

Processing

- Include “lib/BE_Startup.erb” and set up the wizard
- Set up default bands and define variables

Note: Set the variables and the list_menu in an array to be used in option selection through radio buttons. For example: `$list_menu["0"] = "Band Ratio";`
`$list_menu["1"] = "RGB Ratio Composite";` `$list_menu["1"] = "Principal`
`Component Analysis";` `$list_choice = $list_menu["0"]`

- Set the formula for Principal Component Analysis
- First wizard page with container begin/end blocks in the following format

```
#####
WizardPage1:                                     #label for page 1
WizardPage begin "WizardName"                     #title for first page
    title "Page1Name"                             #if a new wizard
    container begin "DataEntry" #selecting datasets

container information                             #usually image info
```

```

...                                #on first page
container end
container begin "Image" # image/s to be displayed on the wizard page
    container information
    ...
container end
container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage2
    ask action "Finish" goto Quit
    ask action "Cancel" close
container end
WizardPage end
#####

```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the first wizard begin/ end block there are three container begin/end blocks. The first container block is for data entry including processing options of Band Ratio, RGB Ratio and RGB composite of Principal Components (through radio buttons), and smoothing. The second container block is for tiff file images to be displayed on the first wizard page. The third container block is for control.

Tip: Ask action command without any control will make that particular button inactive and greyed out. For example, (ask action “Finish”) without goto Quit will make the Finish button inactive.

- Select two bands to process band ratio. (Wizard pages 2)
- Second wizard page with container begin/end blocks in the following format

```

#####
WizardPage2:                                #label for page 2
WizardPage begin "WizardName"                #title for first page
    title "Page1Name"                        #if a new wizard
    container begin "Image Properties " #selecting bands

container information                        #usually bands info
...                                          #on secondpage
container end
container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage1 #for next process option
    ask action "Finish" goto Quit
    ask action "Cancel" close

```

```

        container end
WizardPage end
#####

```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the second wizard begin/ end block there are two container begin/end blocks. The first container block is for selecting two bands to process Band Ratio image. The second container block is for control.

- Third wizard page with container begin/end blocks in the following format

```

#####
WizardPage3:                                     #label for page 3
WizardPage begin "WizardName"                   #title for first page
    title "Page1Name"                           #if a new wizard
    container begin "Image Properties " #selecting bands

container information                            #usually bands info
    ...                                          #on secondpage
    container end
container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage1 #for next process option
    ask action "Finish" goto Quit
    ask action "Cancel" close
    container end
WizardPage end
#####

```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the third wizard begin/ end block there are two container begin/end blocks. The first container block is for selecting six bands to process RGB Ratio composite image. The second container block is for control.

- The fourth wizard page with container begin/end blocks in the following format

```

#####
WizardPage3:                                     #label for page 3
WizardPage begin "WizardName"                   #title for first page
    title "Page1Name"                           #if a new wizard
    container begin "Image Properties " #selecting bands

container information                            #usually bands info

```



```

...                                #on secondpage
container end
container begin "PageControls"    #control buttons
    ask action "Next >" goto WizardPage1 #for next process option
    ask action "Finish" goto Quit
    ask action "Cancel" close
container end
WizardPage end
#####

```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the third wizard begin/ end block there are two container begin/end blocks. The first container block is for selecting number of bands you wish to process RGB composite image of Principal Components. The second container block is for control.

- A section to check the validity of datasets. If there is no dataset give a warning message and goto wizard page 1.
- A section to process a Band Ratio image.
- A section to process a RGB Ratio composite image.
- A section to process a RGB composite image of Principal Components.
- Use conditional and unconditional controls to direct the processing path/s for specific processing option.
- Copy the algorithm to window and display the Band Ratio, RGB Ratio composite , and RGB composite image of Principal Components

Writing the wizard script

Type the following wizard script in a Text Editor.

```
#####
```

1:Wizard - Band ratio, RGB-ratio & PCA

```
#####
```

```

#
# Copyright 1997 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source

```

```

# code contained herein.
#
# Script:   AM_wizard_ratio_RGBratio_PC.erb
# Date:    4th August 1997
# Author:   Abdullah Mah
#
# Summary:  Wizard to create (1)ratio (2)RGB ratio composite
#           and (3) Principal Component Analysis algorithms
#
# Details:
#   1) Set default bands for band ratio
#   2) Set default bands for RGB ratio composite
#   3) Set array index for 3/4/5/6 bands PCA options
#   4) Set PCA formulas for 3/4/5/6 bands PCA options
#   5) Use Wizard begin/end block and create multi-wizard pages
#   6) Inside the wizard begin/end blocks use container begin/end blocks
#   7) Use ask file command to import a dataset
#   8) Set radio buttons for Band Ratio, RGB ratio composite, PCA selection
#       options
#   9) Use ask listmenu (dropdown list) command to select 3/4/5/6 bands PCA
#  10) Set the algorithm mode to pseudo/rgb
#  11) Use ask yesno command for smoothing option
#  12) Set pseudo layer for band ratio
#  13) Set RGB layers for RGB ratio composite
#  14) Set the RGB layers for 3/4/5/6 bands PCA options
#  15) Set the algorithm description layer/s for band ratio and RGB ratio
#       composite
#  16) Set the algorithm descriptions for 3/4/5/6 bands PCA options
#  17) set the formulas for 3/4/5/6 bands PCA in the layers
#  18) Enhance the pseudo/RGB layers transforms of the Ratio/RGBratio/PCA
#       images
#  19) Copy algorithm to window and display the Ratio/RGBratio/PCA images
#
include "lib/BE_Startup.erb"
# set the default bands
$bandR1 = 1
$bandR2 = 2
$band1= 1
$band2= 2
$band3= 3
$band4= 4
$band5= 5
$band6= 7
$listmenu["0"] = "Band Ratio"

```

Chapter 11 Wizard Ratio, RGB-ratio and PCA ● 1:Wizard - Band ratio, RGB-ratio & PCA

```

$listmenu["1"] = "RGB Ratio Composite"
$listmenu["2"] = "Principal Component Analysis"
$list_choice = $listmenu["0"]
$list_PCAMenu["0"] = "3-Bands"
$list_PCAMenu["1"] = "4-Bands"
$list_PCAMenu["2"] = "5-Bands"
$list_PCAMenu["3"] = "6-Bands"
$list_PCAchoice = $list_PCAMenu["0"]
$formula_3PC1 = "SIGMA(I1..I3|I? * PC_COV(I1..I3|,R1,I?,1))"
$formula_3PC2 = "SIGMA(I1..I3|I? * PC_COV(I1..I3|,R1,I?,2))"
$formula_3PC3 = "SIGMA(I1..I3|I? * PC_COV(I1..I3|,R1,I?,3))"
$formula_4PC1 = "SIGMA(I1..I4|I? * PC_COV(I1..I4|,R1,I?,1))"
$formula_4PC2 = "SIGMA(I1..I4|I? * PC_COV(I1..I4|,R1,I?,2))"
$formula_4PC3 = "SIGMA(I1..I4|I? * PC_COV(I1..I4|,R1,I?,3))"
$formula_5PC1 = "SIGMA(I1..I5|I? * PC_COV(I1..I5|,R1,I?,1))"
$formula_5PC2 = "SIGMA(I1..I5|I? * PC_COV(I1..I5|,R1,I?,2))"
$formula_5PC3 = "SIGMA(I1..I5|I? * PC_COV(I1..I5|,R1,I?,3))"
$formula_6PC1 = "SIGMA(I1..I6|I? * PC_COV(I1..I6|,R1,I?,1))"
$formula_6PC2 = "SIGMA(I1..I6|I? * PC_COV(I1..I6|,R1,I?,2))"
$formula_6PC3 = "SIGMA(I1..I6|I? * PC_COV(I1..I6|,R1,I?,3))"
#          *          *          *          *          *          *          *          *          *          *
Page1:
Wizard begin "Ratio, RGB_Ratio & PCA"
    title "Ratio, RGB_Ratio & PC - Data Entry"
    container begin "DataEntry"
        container items labels_left
            say ""
            say "This allows you to create Band Ratio, RGB Ratio composite & PCA
images"
            say ""
            container items labels_above
                ask file "Select a dataset ?" ".ers" $Dataset1
                say ""
                say "Which image do you want to create ?"
                ask listmenu_exclusive "" $listmenu $list_choice
                say ""
                ask yesno "Smoothing " $yesSmoothing
            container end
        container begin "Images"
            width_pixels 64
            left "DataEntry"
            above "PageControls"
            show image "standard_icons/colordraped"
        container end
    container end

```

```

        container begin "PageControls"
        container height_pct 10
        container below "DataEntry"
        container items horizontal right justify
            ask action "Next >" goto Determine_next_page
#            ask action "Finish" goto Page1Finish
            ask action "Cancel" close goto WizardCancel
        container end
Wizard end
Determine_next_page:
if ($Dataset1 == "") then goto warning1 else goto pursuel
warning1:
say warning "Please enter a file name"
goto page1
pursuel:
    if ($list_choice == "Band Ratio") then goto BandRatioSelected else goto
NoBandRatio
BandRatioSelected:
$Goto = "Page2"
goto GotoValidatedata
NoBandRatio:
    if ($list_choice == "RGB Ratio Composite") then goto RGBRatioSelected else
goto NoRGBRatio
RGBRatioSelected:
$Goto = "Page3"
goto GotoValidatedata
NoRGBRatio:
    if ($list_choice == "Principal Component Analysis") then goto PCASelected
PCASelected:
$Goto = "Page4"
GotoValidatedata:
    goto ValidateData
Page1Finish:
    $Goto = "Finish"
    goto ValidateData
#      *      *      *      *      *      *      *      *      *      *
Page2:
new window
first surface
new algorithm
first active raster layer
set layer dataset $Dataset1
$Ratio_bcount = get layer band count
delete window

```

Chapter 11 Wizard Ratio, RGB-ratio and PCA ● 1:Wizard - Band ratio, RGB-ratio & PCA

```
if ($Ratio_bcount <2) then goto select_new_Ratio_data1 else goto Ratio1_cont
select_new_Ratio_data1:
say warning "Select dataset with 2 or > bands"
goto Page1
Ratio1_cont:
Wizard begin "Ratio, RGB_Ratio & PCA"
    title "Band Ratio - Image Properties"
    container begin "DataEntry"
        items labels_left
        say ""
        say ""
        say ""
        say "Select two bands for Band Ratio"
        say ""
        newline
        ask bandmenu "Select a band for Input 1" $Dataset1 $bandR1
        ask bandmenu "Select a band for Input 2" $Dataset1 $bandR2
    container end
    container begin "PageControls"
    container height_pct 13
    container below "DataEntry"
    container items horizontal right justify
        say "Next = Display & Process next image. Finish = Display & Quit
the program"
        newline
        ask action "Next >" goto Page2Next
        ask action "Finish" goto Page2Finish
        ask action "Cancel" close goto WizardCancel
    container end
Wizard end
Page2Next:
    $Goto = "ProcessBandRatio"
    goto ValidateData
Page2Finish:
    $Goto = "Finish"
    goto ValidateData
#      *      *      *      *      *      *      *      *      *      *
Page3:
new window
first surface
new algorithm
first active raster layer
set layer dataset $Dataset1
$bcount = get layer band count
```

```

delete window
if ($bcount <3) then goto select_new_data1 else goto RGBratio3_cont
select_new_data1:
say warning "Select dataset with 3 or > bands"
goto Page1
RGBratio3_cont:
if ($bcount ==4) then goto redefine_default_bands else goto RGBratio_cont1
redefine_default_bands:
$band1 = 2
$band2 = 4
$band3 = 2
$band4 = 3
$band5 = 4
$band6 = 3
goto RGBratio_cont2
RGBratio_cont1:
$band1 = 3
$band2 = 2
$band3 = 4
$band4 = 3
$band5 = 5
$band6 = 7
RGBratio_cont2:
Wizard begin "Ratio, RGB_Ratio & PCA"
    title "RGB Ratio - Image Properties"
    container begin "DataEntry"
        items labels_left
        say ""
        say ""
    newline
    say ""
    say "Select two bands for Red layer (ratio)"
    ask bandmenu "Select a band for Input1" $Dataset1 $band1
    ask bandmenu "Select a band for Input2" $Dataset1 $band2
    say ""
    say "Select two bands for Green layer (ratio)"
    ask bandmenu "Select a band for Input1" $Dataset1 $band3
    ask bandmenu "Select a band for Input2" $Dataset1 $band4
    say ""
    say "Select two bands for Blue layer (ratio)"
    ask bandmenu "Select a band for Input1" $Dataset1 $band5
    ask bandmenu "Select a band for Input2" $Dataset1 $band6
    container end

```

Chapter 11 Wizard Ratio, RGB-ratio and PCA ● 1:Wizard - Band ratio, RGB-ratio & PCA

```
        container begin "PageControls"
        container height_pct 13
        container below "DataEntry"
        container items horizontal right justify
            say "Next = Display & Process next image.  Finish = Display & Quit
the program"
            newline
            ask action "Next >" goto Page3Next
            ask action "Finish" goto Page3Finish
            ask action "Cancel" close goto WizardCancel
        container end
Wizard end
Page3Next:
    $Goto = "ProcessRGBRatio"
    goto ValidateData
Page3Finish:
    $Goto = "Finish"
    goto ValidateData
#      *      *      *      *      *      *      *      *      *      *
Page4:
new window
first surface
new algorithm
first active raster layer
set layer dataset $Dataset1
$PCAbcount = get layer band count
println $PCAbcount
delete window
if ($PCAbcount <3) then goto select_new_PCAdatal else goto PCA1_cont
select_new_PCAdatal:
say warning "Select dataset with 3 or > bands"
goto Page1
PCA1_cont:
if ($PCAbcount ==3) then goto redefine_bands_default1 else goto PCA_bands_cont1
redefine_bands_default1:
$list_PCAMenu["0"] = "3-Bands"
$list_PCAMenu["1"] = ""
$list_PCAMenu["2"] = ""
$list_PCAMenu["3"] = ""
goto PCA_pursue_process
PCA_bands_cont1:
if ($PCAbcount ==4) then goto redefine_bands_default2 else goto PCA_bands_cont2
redefine_bands_default2:
$list_PCAMenu["0"] = "3-Bands"
```

```

$list_PCAMenu["1"] = "4-Bands"
$list_PCAMenu["2"] = ""
$list_PCAMenu["3"] = ""
goto PCA_pursue_process
PCA_bands_cont2:
if ($PCAbcount ==5) then goto redefine_bands_default3 else goto PCA_bands_cont3
redefine_bands_default3:
$list_PCAMenu["0"] = "3-Bands"
$list_PCAMenu["1"] = "4-Bands"
$list_PCAMenu["2"] = "5-Bands"
$list_PCAMenu["3"] = ""
goto PCA_pursue_process
PCA_bands_cont3:
if ($PCAbcount >=6) then goto redefine_bands_default4 else goto PCA_bands_cont4
redefine_bands_default4:
$list_PCAMenu["0"] = "3-Bands"
$list_PCAMenu["1"] = "4-Bands"
$list_PCAMenu["2"] = "5-Bands"
$list_PCAMenu["3"] = "6-Bands"
goto PCA_pursue_process
PCA_bands_cont4:
PCA_pursue_process:
Wizard begin "Ratio, RGB_Ratio & PCA"
    title "PCA - Image Properties"
    container begin "DataEntry"
        items labels_left
        say ""
        say ""
    newline
    say ""
    say "Choose number of bands to create PC1, PC2, PC3"
    ask listmenu "No. of bands" "" $list_PCAMenu $list_PCACHoice
    say ""
    container end
    container begin "PageControls"
    container height_pct 13
    container below "DataEntry"
    container items horizontal right justify
        say "Next = Display & Process next image. Finish = Display & Quit
the program"
    newline
    ask action "Next >" goto Page4Next
    ask action "Finish" goto Page4Finish
    ask action "Cancel" close goto WizardCancel

```


Chapter 11 Wizard Ratio, RGB-ratio and PCA ● 1:Wizard - Band ratio, RGB-ratio & PCA

```

        container end
Wizard end
Page4Next:
    $Goto = "ProcessPCA"
    goto ValidateData
Page4Finish:
    $Goto = "Finish"
    goto ValidateData
#      *      *      *      *      *      *      *      *      *      *
ValidateData:
    if ($Dataset1 != "") then goto DatasetOK
    say warning "Please specify a dataset for Ratio, RGB_Ratio or PCA"
    goto Page1
DatasetOK:
    if ($Goto == "Page1") then goto Page1
    if ($Goto == "Page2") then goto Page2
    if ($Goto == "Page3") then goto Page3
    if ($Goto == "Page4") then goto Page4
    if ($Goto == "ProcessBandRatio") then goto CheckBandRatio
    if ($Goto == "ProcessRGBRatio") then goto CheckRGBRatio
    if ($Goto == "ProcessPCA") then goto CheckPCA
    if ($Goto == "Finish") then goto CheckBandRatio
#      *      *      *      *      *      *      *      *      *      *
CheckBandRatio:
    if ($list_choice=="Band Ratio") then goto beginBandRatio else goto CheckRGBRatio

beginBandRatio:
    new window
    first surface
    new algorithm
    set algorithm description "Band Ratio"
    first pseudo layer
    set layer description "Pseudo"
    set layer dataset to $Dataset1
    set layer formula to "(I1 - RMIN(,R1,I1)) / (I2 - RMIN(,R1,I2))"
    set layer input 1 to band $bandR1
    set layer input 2 to band $bandR2
    copy algorithm to window
    go window
    copy algorithm from window
        first surface
        first pseudo layer
        first transform

```

```

        set transform limits to actual
        copy algorithm to window
    go window
    copy algorithm from window
        include "lib/Clip_99_All_Active_Layers.erb"
if ($yesSmoothing == 1) then goto smoothing else goto no_smoothing
smoothing:
    set algorithm supersample type to bilinear
no_smoothing:
include "lib/Delete_All_Inactive_Layers.erb"
    copy algorithm to window
    go window
wizard close
    if ($Goto == "ProcessBandRatio") then goto Page1
    if ($Goto == "Finish") then goto quitprogram
#      *      *      *      *      *      *      *      *      *      *
CheckRGBRatio:
if ($list_choice == "RGB Ratio Composite") then goto beginRGBRatio else goto
CheckPCA
beginRGBRatio:
    new window
    first surface
    new algorithm
    set algorithm description "RGB Ratio Composite"
    set algorithm mode rgb
    add red layer
set layer description "Red"
    set layer dataset to $Dataset1
set layer formula to "(I1 - RMIN(,R1,I1)) / (I2 - RMIN(,R1,I2))"
    set layer formula to "i1/i2"
    set layer input 1 to band $band1
    set layer input 2 to band $band2
    add green layer
    set layer description "Green"
    set layer dataset to $Dataset1
    set layer formula to "(I1 - RMIN(,'All',I1)) / (I2 - RMIN(,'All',I2))"
    set layer input 1 to band $band3
    set layer input 2 to band $band4
    add blue layer
    set layer description "Blue"
    set layer dataset to $Dataset1
    set layer formula to "(I1 - RMIN(,'All',I1)) / (I2 - RMIN(,'All',I2))"
    set layer input 1 to band $band5
    set layer input 2 to band $band6

```

```

        copy algorithm to window
        go window
        copy algorithm from window
    if ($yesSmoothing == 1) then goto smoothing1 else goto no_smoothing1
smoothing1:
    set algorithm supersample type to bilinear
no_smoothing1:
include "lib/Clip_99_All_Active_Layers.erb"

        copy algorithm to window
        go window
wizard close
    if ($Goto == "ProcessRGBRatio") then goto Page1
    if ($Goto == "Finish") then goto quitprogram
#      *      *      *      *      *      *      *      *      *      *
CheckPCA:
if ($list_choice == "Principal Component Analysis") then goto beginPCA else goto
CheckControl
beginPCA:
    new window
    new algorithm
    set algorithm mode rgb
    add red layer
    set layer description "red"
    set layer dataset to $Dataset1
    if ($list_PCchoice == "3-Bands") then goto Rbands3 else goto Rcont1
Rbands3:
    set algorithm description "PCA R(PC1)G(PC2)B(PC3) Composite - 3 bands"
    set layer formula to $formula_3PC1
    set layer input 1 to band $band1
    set layer input 2 to band $band2
    set layer input 3 to band $band3
    goto greenlayer
Rcont1:

    if ($list_PCchoice == "4-Bands") then goto Rbands4 else goto Rcont2
Rbands4:
    set algorithm description "PCA R(PC1)G(PC2)B(PC3) Composite - 4 bands"
    set layer formula to $formula_4PC1
    set layer input 1 to band $band1
    set layer input 2 to band $band2
    set layer input 3 to band $band3
    set layer input 4 to band $band4
    goto greenlayer

```

```

Rcont2:

if ($list_PCChoice == "5-Bands") then goto Rbands5 else goto Rcont3
Rbands5:
set algorithm description "PCA R(PC1)G(PC2)B(PC3) Composite - 5 bands"
set layer formula to $formula_5PC1
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
goto greenlayer
Rcont3:
if ($list_PCChoice == "6-Bands") then goto Rbands6 else goto Rcont4
Rbands6:
set algorithm description "PCA R(PC1)G(PC2)B(PC3) Composite - 6 bands"
set layer formula to $formula_6PC1
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
set layer input 6 to band $band6
goto greenlayer
Rcont4:
# Green layer for PC2
greenlayer:
add green layer
set layer description "green"
set layer dataset to $Dataset1
if $list_PCChoice == "3-Bands" then goto G3bands else goto Gcont1
G3bands:
set layer formula to $formula_3PC2
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
goto bluelayer
Gcont1:
if $list_PCChoice == "4-Bands" then goto G4bands else goto Gcont2
G4bands:
set layer formula to $formula_4PC2
set layer input 1 to band $band1
set layer input 2 to band $band2

```

```

set layer input 3 to band $band3
set layer input 4 to band $band4
goto bluelayer
Gcont2:
if $list_PCChoice == "5-Bands" then goto G5bands else goto Gcont3
G5bands:
set layer formula to $formula_5PC2
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
goto bluelayer
Gcont3:
if $list_PCChoice == "6-Bands" then goto G6bands else goto Gcont4
G6bands:
set layer formula to $formula_6PC2
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
set layer input 6 to band $band6
goto bluelayer
Gcont4:
# Blue layer for PC3
bluelayer:
add blue layer
set layer description "green"
set layer dataset to $Dataset1
if $list_PCChoice == "3-Bands" then goto B3bands else goto Bcont1
B3bands:
set layer formula to $formula_3PC3
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
goto complete3layers
Bcont1:
if $list_PCChoice == "4-Bands" then goto B4bands else goto Bcont2
B4bands:
set layer formula to $formula_4PC3
set layer input 1 to band $band1
set layer input 2 to band $band2

```

```

set layer input 3 to band $band3
set layer input 4 to band $band4
goto complete3layers
Bcont2:
if $list_PCChoice == "5-Bands" then goto B5bands else goto Bcont3
B5bands:
set layer formula to $formula_5PC3
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
goto complete3layers
Bcont3:
if $list_PCChoice == "6-Bands" then goto B6bands else goto Bcont4
B6bands:
set layer formula to $formula_6PC3
set layer input 1 to band $band1
set layer input 2 to band $band2
set layer input 3 to band $band3
set layer input 4 to band $band4
set layer input 5 to band $band5
set layer input 6 to band $band6
goto complete3layers
Bcont4:
complete3layers:
copy algorithm to window
go window
copy algorithm from window
# Cycle through all layers setting limits to actual data limits
first active raster layer
next_active_layer:
last transform
set transform limits to actual
next active raster layer
if ($ERROR == 0) then goto next_active_layer
copy algorithm to window
go window
copy algorithm from window
# Cycle through all layers setting the transform clip to 95%
first active raster layer
next_active_layer:
last transform

```

```

        set transform clip to 95
        next active raster layer
        if ($ERROR ==0) then goto next_active_layer
if ($yesSmoothing == 1) then goto smoothing2 else goto no_smoothing2
smoothing2:
        set algorithm supersample type to bilinear
no_smoothing2:
include "lib/Delete_All_Inactive_Layers.erb"
#include "lib/Clip_99_All_Active_Layers.erb"
        copy algorithm to window
        go window
CheckControl:
wizard close
        if ($Goto == "ProcessPCA") then goto Page1
        if ($Goto == "Finish") then goto quitprogram
#      *      *      *      *      *      *      *      *      *      *
quitprogram:
        wizard close
        say status "Done.\n"
        delete status
WizardCancel:
exit
#####

```

Displaying the Band Ratio image

- 1 Save the batch script file as “JS_wizard_Ratio_RGBratio_PC.erb” in the ‘ERMAPPER/batch’ directory
- 2 Exit the ER Mapper and call it again so that it will recognize your test1.bar toolbar file in the ‘ERMAPPER/config’ directory
- 3 On the ER Mapper main menu click on the toolbar menu. Your “test1.bar” will appear as “ test1” in the Toolbar menu in alphabetical order.

Tip: Your batch script file “**JS_wizard_Ratio_RGBratio_PC.erb**” in the ‘ERMAPPER/batch’ directory should have the same name as the batch script (“JS_wizard_Ratio_RGBratio_PC_16x16” “Example BATCH densitysliced” BATCH “**JS_wizard_Ratio_RGBratio_PC**”) in the “test1.bar” toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 4 Click the “test1” on the Toolbar menu. Numerous buttons will be displayed on the Toolbar.
- 5 From the displayed buttons click on the JS_wizard__Ratio_RGBRatio_PC wizard script button.
- 6 The **Ratio, RGB_Ratio & PC - Data Entry** dialog box appears. This is the first wizard page.

Note: **Ratio, RGB_Ratio & PC - Data Entry** is the title of the wizard and you can change it as you wish in the wizard script.

- 7 On the **Ratio, RGB_Ratio & PC - Data Entry** dialog box select **Pseudocolor draped** option
- 8 On the **Ratio, RGB_Ratio & PC - Data Entry** dialog box select “**Landsat_TM_year_1985.ers**” dataset for the color layer from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 9 On the **Ratio, RGB_Ratio & PC - Data Entry** dialog box select “**Landsat_TM_year_1985.ers**” dataset for the intensity layer from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 10 Select the **smoothing** option
- 11 Click the **Next** button
- 12 **Band Ratio - Image Properties** dialog box appears.
- 13 **B1:0.485_um** and **B2:0.56_um** of the **Landsat_TM_year_1985.ers** are selected for Input 1 and Input 2 respectively by default.
- 14 Select band 3 B3:0.66_um for Input 1 and band 1 B1:0.485_um for Input 1.

Tip: Band ratio TM3/TM1 is to highlight Fe rich sediments

- 15 Click the Next button on the **Band Ratio - Image Properties** dialog box.

Note: Clicking Next button will display the band ratio image and takes you back to wizard page 1 so that you can choose another image processing option from it. Clicking Finish button will display the band ratio image and quit the program.

- 16 The band ratio image is displayed

Displaying the RGB Ratio composite image

#####

Note: You have clicked the **Next** button on the **Band Ratio - Image Properties** dialog box and displayed the band ratio image. By clicking the Next button you have decided to continue the process and create another image from the three image processing options.

- 1 The **Ratio, RGB_Ratio & PC - Data Entry** dialog box reappears. This is the first wizard page.
- 2 On the **Ratio, RGB_Ratio & PC - Data Entry** dialog box select RGB Ratio Composite option.
- 3 On the **Ratio, RGB_Ratio & PC - Data Entry** dialog box select “**Landsat_TM_year_1991.ers**” dataset for RGB Ratio Composite from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 4 Select the **smoothing** option
- 5 Click the **Next** button on the **Ratio, RGB_Ratio & PC - Data Entry** dialog box
- 6 **RGB Ratio - Image Properties** dialog box appears.
- 7 For **Red Layer B3:0.66_um** as **Input 1**, **B2:0.56_um** as **Input 2**, for **Green Layer B4:0.83_um** as **Input 1**, **B3:0.66_um** as **Input 2**, and for **Blue Layer B5:1.65_um** as **Input 1**, **B7:2.215_um** as **Input 2** by default are selected.
- 8 Click the Next button on the **RGB Ratio - Image Properties** dialog box.
- 9 The RGB Ratio composite image is displayed.

Displaying the RGB composite image with Principal Components

#####

Note: You have clicked the **Next** button on the **RGB Ratio - Image Properties** dialog box and displayed the RGB ratio image. By clicking the Next button you have decided to continue the process and create another image from the three image processing options.

- 1 The **Ratio, RGB_Ratio & PC - Data Entry** dialog box reappears. This is the first wizard page.
- 2 On the **Ratio, RGB_Ratio & PC - Data Entry** dialog box select Principal Component Analysis option.
- 3 On the **Ratio, RGB_Ratio & PC - Data Entry** dialog box select “**Landsat_TM_year_1985.ers**” dataset for RGB Composite of Principal Components from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 4 Select the **smoothing** option
- 5 Click the **Next** button on the **Ratio, RGB_Ratio & PC - Data Entry** dialog box
- 6 **PCA - Image Properties** dialog box appears.
- 7 Choose 6-bands from which you would like to generate Principal Components.

Note: Number 7.000000 is printed out on the Batch Engine Output dialog box to remind you the dataset you have selected consists of seven bands. Close the Batch Engine Output dialog box.

- 8 Click the **Finish** button on the **PCA - Image Properties** dialog box.
- 9 The RGB composite of PC1 (Red), PC2(Green) and PC3(Blue) image is displayed.

Note: The Band Ratio, RGB Ratio composite and RGB composite of PC1(Red), PC2(Green) and PC3(Blue) are displayed in three windows.

Exercise:

(1) Write a wizard script to process a Ratio, a RGB Ratio composite and a RGB Principal Components image. In the wizard script set up radio buttons and provide options to select and process Ratio, RGB Ratio composite or RGB Principal Components images. You need to set up a wizard page for each image processing technique to select image properties . Use **ask action** command and conditional **If .. then .. else goto** and unconditional **goto** controls to navigate the processing path of each of the processing technique.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:

Chapter 11 Wizard Ratio, RGB-ratio and PCA ● Exercise:

- For Windows, select **Close** from the window control-menu.
- For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to perform the following tasks in ER Mapper:

- Create multi-page wizard.
- Write wizard scripts for Band Ratio, RGB Ratio composite and RGB composite with PC1(Red), PC2(Green) and PC3(Blue) algorithms and display them.

Multi-tasks Wizard

This chapter describes how to write a multi-tasks wizard script. The multi-tasks include densityslicing, sunshading, colordrapping and 3-D perspective.

Multi-tasks in a wizard

The multi-tasks wizard uses a single band to create densitysliced, sunshaded, colordraped and 3-D perspective images. Densityslicing and sunshading use one band whereas colordrapping and 3-D perspective use the same band but in two layers, one as color layer and the other as intensity layer. All four processing options use the same band and hence you can run either selectively choose only one option or two or three or all the four processing options at the same time creating four images in four windows. An option to link the four windows is provided.

Hands-on exercises

What you will learn...

After completing these exercises, you will know how to create multi-page wizards, select a dataset, select a band from the dataset, assign the band to pseudo layer and an intensity layer, enhance the transforms of the bands and display the algorithms in multiple windows.

- Create multi-page wizard

- Select a dataset
- Set the algorithm mode to pseudo
- Use wizard begin/end, container begin/end blocks, ask action, unconditional and conditional controls
- Choose bands for pseudocolor and intensity layers
- Enhance transforms of the bands in the Batch Engine
- Copy the algorithms to ER Mapper and display the processed images in multiple windows

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

1: Setting up the wizard button

Note: To run a batch script from a wizard button, the batch script file has to be included in the toolbar file. Hence, add the name of the wizard script you are going to create in the “test1.bar” toolbar file.

- 1 Edit the “test1.bar” file and add a new line
**“JS_wizard_multi_tasks_16x16” “Example Wizard_multi_tasks”
BATCH “JS_wizard_multi_tasks”**

Note: You have added “JS_wizard_multi_tasks” batch script file which will refer to the “JS_multi_tasks.erb” batch script file in the ‘ERMAPPER/batch’ directory. You will create the “JS_wizard_multi_tasks.erb” wizard script file below.

Create an image Tiff file to be used as the picture in the icon of the wizard button

Process an image that you would like to use it as the picture in the icon of your batch script wizard button.

Tip: You can use an exiting Tiff file from the ERMAPPER\icon directory

- 1 Create a pseudocolor draped image using a magnetic image from the “Newcastle_Magnetics.ers” dataset. (The Newcastle_Magnetics.ers dataset is in the ERMAPPER\dataset\Core_Datasets directory)
- 2 Follow the procedure in chapter 3 and create a 24-bit 16x 16 pixels “JS_wizard_multi_tasks_16x16.tif” tiff file. Place the tiff file in the ‘ERMAPPER/icons’ directory.
- 3 Also create 24-bit 64x 64 pixels MagneticsPseudo64x64, MagneticsShaded64x64, MagneticsColordraped64x64, Magnetics3D64x64 tiff files. Place the tiff files in the ‘ERMAPPER/icons/standard_icons’ directory.

Note: The 64x64 pixels tiff files will be used on the wizard page. To use a tiff file of a size that will fit the specified area on the wizard you can create a tiff file using other software and import it into ER Mapper.

Write a multi_task wizard script to process and display a densitysliced, a sunshaded, a colordraped and a 3_D perspective images.

Logic:

- To write a multi-task wizard that will process densitysliced, sunshaded, colordraped and 3_D perspective images at the same time you need to select only one dataset. Hence, select a dataset that has a height perspective such as DTM, Magnetics, Gravity or PC1. (Wizard page 1)
- Provide options to choose a single or multiple options to display the colordraped images in 3-D perspective. (Wizard page 1)
- Provide an option to link the windows. (Wizard page 1)
- Select a band to be processed. (Wizard pages 2)
- Provide an option to smooth the processed images. (Wizard page 2)
- Provide an option to invert the data values of the band if the band used is a Two Way Time (TWT) seismic data. (Wizard page 2)
- Process the image/s
- Copy the algorithm/s to window/s and display them

Processing

- Include “lib/BE_Startup.erb” and set up the wizard
- Set up default band and define variables
- Set up the first wizard page with container begin/end blocks in the following format

```
#####
```

```
WizardPage1:
```

```
#label for page 1
```

Chapter 12 Multi-tasks Wizard ● 1: Setting up the wizard button

```
WizardPage begin "WizardName"                                #title for first page
    title "Page1Name"                                         #if a new wizard
    container begin "DataEntry" #selecting datasets

container information                                         #usually image info
    ...                                                       #on first page
container end

    container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage2
    ask action "Finish" goto Quit
    ask action "Cancel" close

container end

container begin "Image" # image/s to be displayed on the wizard page
    container information
    ...
container end

WizardPage end
#####
```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the first wizard begin/ end block there are three container begin/end blocks. The first container block is for data entry including processing options pseudo colordrape or RGB colordrape (through radio buttons), 3-D perspective and smoothing. The second container block is for control. The third container block is for tiff file images to be displayed on the first wizard page.

- Set up the second wizard page with container begin/end blocks in the following format

```
#####
WizardPage2:                                                #label for page 2
WizardPage begin "WizardName"                                #title for first page
    title "Page1Name"                                         #if a new wizard
    container begin "Image Properties " #selecting bands

container information                                         #usually bands info
    ...                                                       #on secondpage
container end

container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage1 #for next process option
    ask action "Finish" goto Quit
    ask action "Cancel" close
```

```

        container end
WizardPage end
#####

```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the second wizard begin/ end block there are two container begin/end blocks. The first container block is for selecting bands to process pseudo colordrape image. The second container block is for control.

- Set up the third wizard page with container begin/end blocks in the following format

```

#####
WizardPage3:                                     #label for page 3
WizardPage begin "WizardName"                   #title for first page
    title "Page1Name"                            #if a new wizard
    container begin "Image Properties " #selecting bands

container information                            #usually bands info
    ...                                          #on secondpage
    container end
container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage1 #for next process option
    ask action "Finish" goto Quit
    ask action "Cancel" close
    container end
WizardPage end
#####

```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the third wizard begin/ end block there are two container begin/end blocks. The first container block is for selecting bands to process RGB colordrape image. The second container block is for control.

- A section to check the validity of datasets. If there is no dataset give a warning message and goto wizard page 1.
- A section to process densitysliced image
- A section to process sunshaded image
- A section to process colordraped image
- A section to process 3D perspective image

- Use conditional and unconditional controls to direct the processing path/s for specific processing option.
- Link windows if the window linking option is chosen
- Apply smoothing if the smoothing option is chosen
- Invert data values if the data is Two Way Time seismic data
- Copy the algorithm to window and display the densitysliced, sunshaded, colordraped and 3_D perspective images

Writing the wizard script

Type the following wizard script in a Text Editor.

```
#####  
  
# Copyright 1997 Earth Resource Mapping Pty Ltd.  
# This document contains unpublished source code of  
# Earth Resource Mapping Pty Ltd. This notice does  
# not indicate any intention to publish the source  
# code contained herein.  
#  
# Script:   GeophysicsWizard.erb  
# Date:     Wednesday Dec 18th WST 1996  
# Author:   David Hayward  
#  
# Summary:  Wizard to create the common types of geophysical images  
#           (e.g. for magnetics, gravity, seismic, etc. data).  
#  
# Details:  
#  
include "lib/BE_Startup.erb"  
$yesnoDensitySliced = 0  
$yesnoShaded = 0  
$yesnoColourdraped = 0  
$yesnoPerspective = 0  
$yesnoGeolinked = 0  
# set the default band  
$band = 1.0  
# use the default lut  
$lut = get preference "Pseudo Lut Name" "pseudocolor"  
# set if processing seismic two-way time data  
$yesnoTwoWayTime = 0  
# set if we want smoothing on supersampled images  
$yesnoSmoothing = 1
```

```

$SupersampleType = bilinear
# parameters for image window creation
$WindowWidth = 400
$WindowHeight = 400
$WindowXOffset = 0
$WindowYOffset = 0
$WindowXIncrement = 20
$WindowYIncrement = 20
#      *      *      *      *      *      *      *      *      *      *
Page1:
Wizard begin "Common Geophysical Images Wizard"
  title "Common Geophysical Images Wizard"
  container begin "DataEntry"
    labels_left
    say "This allows you to create the common types of"
    say "magnetic and gravity images. You need to have"
    say "imported the data before using this wizard."
    say ""
    say "Which images do you want to create?"
    items newline
    show image "MagneticsPseudo16x16"
    items horizontal
    ask yesno "Density Sliced (Pseudocolour) " $yesnoDensitySliced
    newline
    show image "MagneticsShaded16x16"
    ask yesno "Shaded " $yesnoShaded
    newline
    show image "MagneticsColourdraped16x16"
    ask yesno "Colourdraped" $yesnoColourdraped
    newline
    show image "Magnetics3D16x16"
    ask yesno "3D Perspective" $yesnoPerspective
    newline
    say ""
    newline
    container items labels_above
    ask file "Which dataset do you want to use?" ".ers" $Dataset
    newline
    say ""
    newline
    ask yesno "Link the window(s)" $yesnoGeolinked
  container end
  container begin "PageControls"

```

Chapter 12 Multi-tasks Wizard ● 1: Setting up the wizard button

```
        height_pct 12
        below "DataEntry"
        horizontal right justify
#        ask action "< Back"
        ask action "Next >" goto Page1Next
        ask action "Finish" goto Page1Finish
        ask action "Cancel" close goto WizardCancel
    container end
    container begin "Images"
        width_pixels 64
        left "DataEntry"
        above "PageControls"
        show image "standard_icons/Wizards/MagneticsPseudo64x64"
        show image "standard_icons/Wizards/MagneticsShaded64x64"
        show image "standard_icons/Wizards/MagneticsColourdraped64x64"
        show image "standard_icons/Wizards/Magnetics3D64x64"
    container end
Wizard end
Page1Next:
    $Goto = "Page2"
    goto ValidateData
Page1Finish:
    $Goto = "Finish"
    goto ValidateData
#    *    *    *    *    *    *    *    *    *    *
Page2:
Wizard begin "Common Geophysical Images Wizard"
    title "Common Geophysical Images Wizard - Image Properties"
    container begin "DataEntry"
        labels_left
        say "This page allows you to specify additional properties for the"
        say "image(s) to be created."
        say ""
        say ""
        ask bandmenu "Which band do you want to use? " $Dataset $band
        ask yesno "Band is seismic two-way time data." $yesnoTwoWayTime
        say ""
        say ""
        ask lutmenu "Colour lookup table to use: " $lut
        say ""
        say ""
        ask yesno "Apply smoothing when zoomed in or creating hardcopy."
$yesnoSmoothing
    container end
```

Chapter 12 Multi-tasks Wizard ● 1: Setting up the wizard button

```
container begin "PageControls"
    height_pct 12
    below "DataEntry"
    horizontal right justify
    ask action "< Back" goto Page2Back
    ask action "Finish" goto Page2Finish
    ask action "Cancel" close goto WizardCancel
container end
Wizard end
Page2Back:
    $Goto = "Page1"
    goto ValidateData
Page2Finish:
    $Goto = "Finish"
    goto ValidateData
#      *      *      *      *      *      *      *      *      *      *
ValidateData:
CheckImageSpecified:
    if ($yesnoDensitySliced or $yesnoShaded or $yesnoColourdraped or
$yesnoPerspective) then goto CheckDataset
    say warning "Please specify the type(s) of image to create."
    goto Page1
CheckDataset:
    if ($Dataset != "") then goto DatasetOK
    say warning "Please specify a dataset"
    goto Page1
DatasetOK:
    if ($Goto == "Page1") then goto Page1
    if ($Goto == "Page2") then goto Page2
WizardFinish:
    if ($yesnoColourdraped == 0) then goto LutOk
    if ($lut != "greyscale" and $lut != "newgrey") then goto LutOk
    say warning "Please specify a non-grey lookup table for use with the colour
draped image."
    goto Page2
LutOk:
    if ($yesnoSmoothing) then goto SmoothingOn
    $SupersampleType = nearest# turn smoothing off
SmoothingOn:
    wizard close
    if ($yesnoTwoWayTime) then goto TwoWayTimeFormula
    $Formula = "I1"
    goto CheckDensitySliced
TwoWayTimeFormula:
```

Chapter 12 Multi-tasks Wizard ● 1: Setting up the wizard button

```
$Formula = "-I1"
#      *      *      *      *      *      *      *      *      *      *
CheckDensitySliced:
    if ($yesnoDensitySliced == 0) then goto CheckShaded
    say status "Creating Density Slice (Pseudocolour) image...\n"
    new algorithm
    set algorithm description "Density Sliced"
    set algorithm mode pseudo
    set algorithm supersample type to $SupersampleType
    first surface
    set surface lut $lut
    first layer
    set layer description "Density Sliced"
    set layer dataset to $Dataset
    set formula to $Formula
    set layer input 1 to band $band
    new window $nWindowXOffset $nWindowYOffset $nWindowWidth $nWindowHeight
    $nWindowXOffset = $nWindowXOffset + $nWindowXIncrement
    $nWindowYOffset = $nWindowYOffset + $nWindowYIncrement
    copy algorithm to window
    go window
    copy algorithm from window
    first layer
    last transform
    set transform limits to actual
    copy algorithm to window
    go window
    if ($yesnoTwoWayTime) then goto SkipDensitySliceLimitsAdjustment
    copy algorithm from window
    first layer
    last transform
#      *      *      *      *      *      *      *      *      *      *
    set clip to 99
    set to gaussian equalize
    copy algorithm to window
    go window
SkipDensitySliceLimitsAdjustment:
    if ($yesnoGeolinked == 0) then goto CheckShaded
    set window geolink mode to window
#      *      *      *      *      *      *      *      *      *      *
CheckShaded:
    if ($yesnoShaded == 0) then goto CheckColourdraped
    say status "Creating Shaded image...\n"
    new algorithm
```

```

set algorithm description "Shading"
set algorithm mode pseudo
set algorithm lut "greyscale"
set algorithm supersample type to $SupersampleType
first layer
set layer description "Shaded"
set layer dataset to $Dataset
set formula to $Formula
set layer input 1 to band $band
new window $nWindowXOffset $nWindowYOffset $nWindowWidth $nWindowHeight
$nWindowXOffset = $nWindowXOffset + $nWindowXIncrement
$nWindowYOffset = $nWindowYOffset + $nWindowYIncrement
copy algorithm to window
go window
copy algorithm from window
first layer
# This is the current (5.5) workaround to ensure we don't operate on
# subsampled data (set up a dummy filter with the process at dataset
# resolution flag set).
add convolution filter
set filter rows to 1
set filter cols to 1
set filter matrix 1 1 1
set filter postsampled false
last transform
set transform limits to actual
set layer shading on
copy algorithm to window
go window
if ($yesnoGeolinked == 0) then goto CheckColourdraped
set window geolink mode to window
# * * * * *
CheckColourdraped:
if ($yesnoColourdraped == 0) then goto CheckPerspective
say status "Creating Colour Draped image...\n"
new algorithm
set algorithm description "Colourdraped"
set algorithm mode pseudo
set algorithm supersample type to $SupersampleType
set surface lut $lut
first layer
set layer description "Colour"
set layer dataset to $Dataset

```

Chapter 12 Multi-tasks Wizard ● 1: Setting up the wizard button

```
set formula to $Formula
set layer input 1 to band $band
new window $nWindowXOffset $nWindowYOffset $nWindowWidth $nWindowHeight
$nWindowXOffset = $nWindowXOffset + $nWindowXIncrement
$nWindowYOffset = $nWindowYOffset + $nWindowYIncrement
copy algorithm to window
go window
copy algorithm from window
first layer
last transform
set transform limits to actual
duplicate layer
set layer description "Shading"
set layer type to intensity
set layer shading on
# This is the current (5.5) workaround to ensure we don't operate on
# subsampled data (set up a dummy filter with the process at dataset
# resolution flag set).
add convolution filter
set filter rows to 1
set filter cols to 1
set filter matrix 1 1 1
set filter postsampled false
# turn layer off while redisplaying so we can adjust limits on colour layer
turn layer off
copy algorithm to window
go window
if ($yesnoTwoWayTime) then goto SkipColourDrapeClip
# Clip the transform on the colour layer.
copy algorithm from window
first layer
last transform
# set clip to 99
set clip to 99
set to gaussian equalize
SkipColourDrapeClip:
# set the current layer to the shaded layer so they can bring up the
# shading dialogue and use it directly
last layer
turn layer on
copy algorithm to window
go window
if ($yesnoGeolinked == 0) then goto CheckPerspective
set window geolink mode to window
```

```

#      *      *      *      *      *      *      *      *      *      *
CheckPerspective:
    if ($yesnoPerspective == 0) then goto FinishUp
    say status "Creating 3D Perspective image...\n"
    new algorithm
    set algorithm description "3D Perspective"
    set algorithm mode pseudo
    set algorithm supersample type to $SupersampleType
    first layer
    set layer description "Colour"
    set layer dataset to $Dataset
    set formula to $Formula
    set layer input 1 to band $band
    new window $nWindowXOffset $nWindowYOffset $nWindowWidth $nWindowHeight
    $nWindowXOffset = $nWindowXOffset + $nWindowXIncrement
    $nWindowYOffset = $nWindowYOffset + $nWindowYIncrement
    copy algorithm to window
    go window
    copy algorithm from window
    first layer
    last transform
    set transform limits to actual
    copy algorithm to window
    go window
    copy algorithm from window
    first layer
    # create the height layer
    duplicate layer
    next layer
    set layer description "Height"
    set layer type to Height
    if ($yesnoTwoWayTime) then goto SkipPerspectiveClip
    # Clip the transform on the colour layer.
    first layer
    last transform
#      set clip to 99
    set to gaussian equalize
SkipPerspectiveClip:
    set view mode perspective
    copy algorithm to window
    go window
    if ($yesnoGeolinked == 0) then goto FinishUp
    set window geolink mode to window

```


Chapter 12 Multi-tasks Wizard ● 1: Setting up the wizard button

```
#      *      *      *      *      *      *      *      *      *      *
FinishUp:
    # if have a density slice image only then pop up transform
    if ($yesnoDensitySliced == 1 and $yesnoShaded == 0 and $yesnoColourdraped ==
0 and $yesnoPerspective == 0) then goto PopTransformDialogue
    # if have a density slice or perspective image then do nothing
    if ($yesnoDensitySliced == 1 or $yesnoPerspective == 1) then goto Done
    # if no shaded or colourdraped then do nothing
    if ($yesnoShaded == 0 and $yesnoColourdraped == 0) then goto Done
PopShadeDialogue:
    # if we get here we have shaded and/or colourdraped but nothing else
    open sunangle window
    goto Done
PopTransformDialogue:
    open transform window
    goto Done
Done:
    say status "Done.\n"
    delete status
WizardCancel:
exit
#####
```

Using multi-task wizard to process and display densitysliced, sunshaded, colordraped and 3_D perspective image in 4 windows

- 1 Save the batch script file as “JS_wizard_multi_tasks.erb” in the ‘ERMAPPER/batch’ directory
- 2 Exit the ER Mapper and call it again so that it will recognize your test1.bar toolbar file in the ‘ERMAPPER/config’ directory
- 3 On the ER Mapper main menu click on the toolbar menu. Your “test1.bar” will appear as “ test1” in the Toolbar menu in alphabetical order.

Tip: Your batch script file “**JS_wizard_multi_tasks.erb**” in the ‘ERMAPPER/batch’ directory should have the same name as the batch script (“JS_RGB_541” “Example Wizard_multi_tasks” BATCH “**JS_wizard_multi_tasks**”) in the “test1.bar” toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 4 Click the “test1” on the Toolbar menu. Numerous buttons will be displayed on the Toolbar.
- 5 From the displayed buttons click on the JS_wizard_multi_tasks wizard script button, the **Common Geophysical Images Wizard** dialog box appears. This is the first wizard page.

Note: **Common Geophysical Images Wizard** is the title of the wizard and you can change it as you wish in the wizard script.

- 6 On the **Common Geophysical Images Wizard** dialog box select all the **Density Sliced (Pseudocolor), Shaded, Colordraped and 3D Perspective** options
- 7 On the **Common Geophysical Images Wizard** dialog box select “**Newcastle_Magnetics.ers**” dataset for the intensity layer from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 8 Select the **Link the window(s)** option
- 9 Select the **3-D Perspective** option
- 10 Click the **Next** button

Note: You can also click the Finish button to process the algorithms, display them and quit the program. Clicking the Next button gives you more options to choose in the second wizard page.

- 11 **Common Geophysical Images Wizard - Image Properties** dialog box appears.
- 12 **B1:magnetrics** of the **Newcastle_Magnetics.ers** by default is selected.

Note: For dataset with more than one band, select the band of your choice

- 13 Choose a color from the Color lookup table. Choose Pseudocolor.
- 14 Select the smoothing option.
- 15 Click the Finish button on the **Common Geophysical Images Wizard - Image Properties** dialog box.

Note: If you click the < **Back** button, it will take you back to the first wizard page and allows you to choose a different dataset

- 16 The densitysliced, sunshaded, colordraped and 3_D perspective images are displayed in 4 windows. The windows are linked and the bilinear smoothing filter is applied to all the images.

Exercises:

(1) Write a wizard script to process images of densitysliced, sunshaded, colordraped and 3-D perspective . In the wizard script provide options to choose either one or more than one or all the processing techniques. Set up a wizard page for each image processing technique to select image properties. Use **ask action** command and conditional **If .. then .. else goto** and unconditional **goto** controls to navigate the processing path of each of the processing technique.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:
 - For Windows, select **Close** from the window control-menu.
 - For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to perform the following tasks in ER Mapper:

- Create multi_task wizard.
- Write a multi_tasks wizard script for densitysliced, sunshaded, colordraped and 3D perspective algorithms and display them in four image windows

Wizard: Colordrape images in multisurfaces

This chapter shows you how to write a multipage wizard script, import multiple datasets, build pseudo and RGB colordraped algorithms in multi surfaces.

About wizards creating images in multi-surfaces

In writing wizard scripts to create images in multi-surfaces, multiple wizard pages are necessary and hence 'page controls' are critical. It is important to define the processing path clearly for each option to create an image in a surface. If necessary 'dummy variables' may be defined to use as controls. '**If ---- then ---- else goto--**' conditional and '**Goto--**' unconditional controls are used and care should be taken to avoid using un-necessary identical labels that the conditional and unconditional controls refer to. 'Goto--' controls search labels which they refer to, from top to bottom of the script.

Wizards are made up of one or more pages that step a user through a task. They guide the user to making choices, providing any necessary information and suggesting appropriate choices when possible. They are especially useful for complex tasks or tasks requiring a number of steps. General batch script commands mentioned in the last chapter are applicable in writing wizard scripts.

As in batch script, wizard scripts execute the commands through batch engine - quite separate from ER Mapper itself. Hence, to display an algorithm developed in a wizard script it has to be copied to ER Mapper and do a Go in the window.

Hands-on exercises

These exercises teach you how to create wizards with multiple pages, import multiple datasets, select bands from the datasets and assign them to layers, enhance transforms of the bands and display the processed images in multisurfaces.

What you will learn...

After completing these exercises, you will know how to create multi-page wizards, select multiple datasets, select bands from the datasets, assign the bands to layers, enhance the transforms of the bands and display the algorithms in multisurfaces.

- Create multi-page wizard
- Import two datasets
- Set the algorithm mode to pseudo and rgb
- Use wizard begin/end, container begin/end blocks, ask action, unconditional and conditional controls
- Choose bands for pseudocolor, RGB and intensity layers
- Enhance transforms of the bands in the Batch Engine
- Copy the algorithms (pseudo & RGB color draped) to ER Mapper and display the processed images in two surfaces

Before you begin...

Before beginning these exercises, make sure all ER Mapper image windows are closed. Only the ER Mapper main menu should be open on the screen.

1: Setting up the wizard button

Note: To run a batch script from a wizard button, the batch script file has to be included in the toolbar file. Hence, add the name of the wizard script you are going to create in the “test1.bar” toolbar file.

- 1 Edit the “test1.bar” file and add a new line **“JS_wizard_colordrape”**
“Example Wizard_colordrape” BATCH “JS_wizard_colordrape”

Note: You have added “JS_wizard_colordrape” batch script file which will refer to the “JS_wizard_colordrape.erb” batch script file in the ‘ERMAPPER/batch’ directory. You will create the “JS_wizard_colordrape.erb” wizard script file below.

Create an image Tiff file to be used as the picture in the icon of the wizard button

Process an image that you would like to use it as the picture in the icon of your batch script wizard button.

Tip: You can use an exiting Tiff file from the ‘ERMAPPER/icon’ directory

- 1 Create a pseudocolor draped image using a magnetic image from the “Newcastle_Magnetics.ers” dataset.

Note: The Newcastle_Magnetics.ers dataset is in the ERMAPPER\dataset\Core_Datasets directory.

- 2 Follow the procedure in chapter 3 and create a 24-bit 16x 16 pixels “JS_pseudocolor_drape_16x16.tif” tiff file. Place the tiff file in the ‘ERMAPPER/icons’ directory.
- 3 Also create a 24-bit 64x 64 pixels “JS_pseudocolor_drape_64x64.tif” tiff file. Place the tiff file in the ‘ERMAPPER/icons/standard_icons’ directory.

Note: The 64x64 pixels tiff file will be used on the wizard page. To use a tiff file of a size that will fit the specified area on the wizard you can create a tiff file using other software and import it into ER Mapper.

Write a wizard script to process and display a pseudocolor colordrape and a RGB colordrape images in two surfaces.

Logic:

- To create a colordrape image you need two layers. A pseudocolor layer or a RGB composite and an intensity layer. Hence, select a dataset for the pseudocolor layer or a RGB composite. The pseudocolor layer or a RGB composite can be any raster type image whereas the intensity layer should be an image with height perspective. To display colordrape images in multisurfaces and in 2-D mode, on the intensity layer you apply the sunangle to highlight structure and overdrape the pseudocolor layer or a RGB composite on top of it. To display colordrape images in multisurfaces and in 3-D perspective you need a height layer. If there is no height layer then the intensity layer will be taken as the height layer.
- Select a dataset that has a height perspective such as DTM, Magnetics, Gravity or PC1. (Wizard page 1)
- Provide an option to display the colordraped images in 3-D perspective. (Wizard page 1)
- Provide an option to smooth the colordraped images. (Wizard page 1)
- Select band/s for pseudo or RGB composite and intensity layers. (Wizard pages 2 & 3)
- Process the colordrape image/s
- Copy the algorithm/s to window and display them

Processing

- Include “lib/BE_Startup.erb” and set up the wizard
- Set up default bands and define variables

Note: Set the variables and the list_menu in an array to be used in option selection through radio buttons. For example: `$list_menu["0"] = "Pseudocolordraped"; $list_menu["1"] = "RGBcolordraped"; $list_choice = $list_menu["0"]`

- First wizard page with container begin/end blocks in the following format

```
#####
WizardPage1:                                     #label for page 1
WizardPage begin "WizardName"                   #title for first page
    title "Page1Name"                           #if a new wizard
    container begin "DataEntry" #selecting datasets

container information                            #usually image info
    ...                                         #on first page
container end
container begin "Image" # image/s to be displayed on the wizard page
    container information
    ...
container end
container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage2
    ask action "Finish" goto Quit
    ask action "Cancel" close
container end
WizardPage end
#####
```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the first wizard begin/ end block there are three container begin/end blocks. The first container block is for data entry including processing options pseudo colordrape or RGB colordrape (through radio buttons), 3-D perspective and smoothing. The second container block is for tiff file images to be displayed on the first wizard page. The third container block is for control.

- Second wizard page with container begin/end blocks in the following format

```
#####
WizardPage2:                                     #label for page 2
WizardPage begin "WizardName"                   #title for first page
    title "Page1Name"                           #if a new wizard
    container begin "Image Properties " #selecting bands

container information                            #usually bands info
    ...                                         #on secondpage
container end
container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage1 #for next process option
    ask action "Finish" goto Quit
    ask action "Cancel" close
```



```

        container end
WizardPage end
#####

```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the second wizard begin/ end block there are two container begin/end blocks. The first container block is for selecting bands to process pseudo colordrape image. The second container block is for control.

- Third wizard page with container begin/end blocks in the following format

```

#####
WizardPage3:                                     #label for page 3
WizardPage begin "WizardName"                   #title for first page
    title "Page1Name"                            #if a new wizard
    container begin "Image Properties " #selecting bands

container information                            #usually bands info
    ...                                           #on secondpage
    container end
container begin "PageControls" #control buttons
    ask action "Next >" goto WizardPage1 #for next process option
    ask action "Finish" goto Quit
    ask action "Cancel" close
    container end
WizardPage end
#####

```

Note: If the subsequent wizards have the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. Within the third wizard begin/ end block there are two container begin/end blocks. The first container block is for selecting bands to process RGB colordrape image. The second container block is for control.

- A section to check the validity of datasets. If there is no dataset give a warning message and goto wizard page 1.
- A section to process pseudo colordrape image
- A section to process RGB composite colordrape image
- Use conditional, unconditional and dummy (variable) controls to direct the processing path/s for specific processing option.

- If the option is to display 2 images in 2 surface in 2-D mode set 1st surface transparency to 0
- If the option is to display 2 images in 2 surface in 3-D mode set surface Zoffsets to 6000 and -6000

Note: Zoffset is relative to the image size

- Enhance the transforms of the band/s
- Delete inactive layers
- Copy the algorithm to window and display the pseudo/RGB colordraped image/s

Writing the wizard script

Type the following wizard script in a Text Editor.

```
#####
#
# Copyright 1997 Earth Resource Mapping Pty Ltd.
# This document contains unpublished source code of
# Earth Resource Mapping Pty Ltd. This notice does
# not indicate any intention to publish the source
# code contained herein.
#
# Script:   AM_wizard_colordrape.erb
# Date:     4th August 1997
# Author:   Abdullah Mah
#
# Summary:  Wizard to create pseudo and RGB colordraped algorithms
#           and display them in 2 surfaces
#
# Details:
#           1) Use Wizard begin/end block and create multi-wizard pages
#           2) Inside the wizard begin/end blocks use container begin/end blocks
#           3) Use ask file command to select two datasets for colordrape images
#           4) Select option/s to create Pseudocolordrape or RGBcolordrape image/s
#           5) Use ask bandmenu command to select a band for pseudocolordrape layer
#           6) Use ask bandmenu command to select three bands for RGBcolordrape RGB
#           layers
#           6) Use ask bandmenu command to select a band for intensity layer
#           7) Set the algorithm mode to pseudo/rgb
#           8) Use ask yesno command for smoothing option
#           8) Use ask yesno command for 3-D perspective option
```

Chapter 13 Wizard:Colordrape images in multisurfaces ● 1: Setting up the wizard button

```
#          9) Use ask action commands to navigate the process
#          11) If 2-D mode and 2 images in 2 surface set 1st surface transparency to 0
#          11) If 3-D mode and 2 images in 2 surface set surface Zoffsets to 6000 and -
6000
#          12) Enhance the transforms of the band/s
#          13) Delete inactive layers
#          14) Copy the algorithm to window and display the pseudo/RGB colordraped
image/s
#
include "lib/BE_Startup.erb"
# set the default bands & define variables
$band = 1
$band1 = 1
$band2 = 2
$band3 = 3
$band_intensity=1
$yesnoPseudocolordraped= 0
$yesnoRGBcolordraped = 0
$Goto1 = "dummy1"
$Goto2 = "dummy2"
$test_surface = "test_surface1"
$list_menu["0"] = "Pseudocolordraped"
$list_menu["1"] = "RGBcolordraped"
$list_choice =$list_menu["0"]
#          *          *          *          *          *          *          *          *          *
Page1:
Wizard begin "Pseudo & RGB colordrape"
    title "Pseudo & RGB colordrape - Data Entry"
    container begin "DataEntry"
        container items labels_left
        say "This allows you to create Pseudo & RGB colordrape images in 2
surfaces"
        say ""
        say "Which images do you want to create?"
        ask listmenu_exclusive "" $list_menu $list_choice
        say ""
        container items labels_above
        ask file "Select a dataset for color layer?" ".ers" $Dataset1
        newline
        container items labels_above
        ask file "Select a dataset for intensity layer? (DTM, Magnetics,
Gravity or PC1)" ".ers" $Dataset2
        newline
        ask yesno "3-D Perspective" $yesnoPerspective
        ask yesno "Smoothing " $yesSmoothing
```

```

        container end
        container begin "Images"
            width_pixels 64
            left "DataEntry"
            above "PageControls"
            show image "standard_icons/colordraped"
            show image "standard_icons/RGBdraped"
        container end
        container begin "PageControls"
        container height_pct 12
        container below "DataEntry"
        container items horizontal right justify
            ask action "Next >" goto determine_next_page
            ask action "Finish"
            ask action "Cancel" close goto WizardCancel
        container end
    Wizard end

    determine_next_page:
        if ($list_choice=="Pseudocolordraped") then goto Page1Next
        if ($list_choice=="RGBcolordraped") then goto Page2Next
        say warning "Please specify the type(s) of image to create."
        goto Page1

    Page1Next:
        $Goto = "Page2"
        goto ValidateData

    Page2Next:
        $Goto = "Page3"
        goto ValidateData

#      *      *      *      *      *      *      *      *      *      *
Page2:
Wizard begin "Pseudo & RGB colordrape"
    title "Pseudo colordrape - Image Properties"
    container begin "Pseudo colordrape - DataEntry"
        container items labels_left
        say "Select a band for pseudo layer and a band for intensity layer."
        say ""
        ask bandmenu "Select a band for pseudo layer? " $Dataset1 $band
        say ""
        ask bandmenu "Select a band for intensity layer? " $Dataset2
$band_intensity
    container end
    container begin "PageControls"
    container height_pct 15
    container below "Pseudo colordrape - DataEntry"

```

Chapter 13 Wizard:Colordrape images in multisurfaces ● 1: Setting up the wizard button

```
        container items horizontal right justify
        say "Next = To display and continue.      Finish = To display and quit."
        newline
        ask action "Next" goto NextPage1
        ask action "Finish" goto Quit1
        ask action "Cancel" close goto WizardCancel
    container end
Wizard end
Quit1:
    if ($list_choice=="Pseudocolordraped") then goto DisplayQuit1 else goto
FinalQuit1
DisplayQuit1:
    $Goto ="Finish1"
    $Goto1="Finish11"
    goto cont11
FinalQuit1:
    $Goto="Finish"
cont11:
    goto ValidateData
NextPage1:
    $Goto = "Finish1"
    goto CheckPseudocolordraped
#      *      *      *      *      *      *      *      *      *      *
Page3:
Wizard begin "Pseudo & RGB colordrape"
    title "RGB colordrape - Image Properties"
    container begin "RGB colordrape - DataEntry"
        container items labels_left
        say "Select 3 bands for RGB composite and a band for intensity
layer."
        say ""
        ask bandmenu "Select a band for Red? " $Dataset1 $band1
        ask bandmenu "Select a band for Green? " $Dataset1 $band2
        ask bandmenu "Select a band for Blue? " $Dataset1 $band3
        say ""
        ask bandmenu "Select a band for intensity layer? " $Dataset2
$band_intensity
    container end
    container begin "Page3Controls"
    container height_pct 15
    container below "RGB colordrape - DataEntry"
    container items horizontal right justify
    say "Next = To display and continue.      Finish = To display and quit."
    newline
```

```

        ask action "Next"    goto NextPage2
        ask action "Finish" goto Quit2
        ask action "Cancel" close goto WizardCancel
    container end
Wizard end
Quit2:
    if ($list_choice=="RGBcolordraped") then goto DisplayQuit2 else goto
FinalQuit2
    DisplayQuit2:
        $Goto ="Finish2"
        $Goto2="Finish22"
        goto cont22
FinalQuit2:
    $Goto="Finish"
cont22:
    goto ValidateData
NextPage2:
    $Goto = "Finish2"
    goto CheckRGBcolordraped
#      *      *      *      *      *      *      *      *      *      *
ValidateData:
    wizard close
    if ($list_choice=="Pseudocolordraped" or $list_choice=="RGBcolordraped")
then goto CheckDataset
    say warning "Please specify the type(s) of image to create."
    goto Page1
CheckDataset:
    if ($Dataset1 != "") then goto DatasetOK
    say warning "Please specify datasets for colordrape and intensity"
    goto Page1
DatasetOK:
    if ($Goto == "Page1") then goto Page1
    if ($Goto == "Page2") then goto Page2
    if ($Goto == "Page3") then goto Page3
    if ($Goto == "Finish") then goto FinishUp
    if ($Goto == "Finish1") then goto CheckPseudocolordraped
    if ($Goto == "Finish2") then goto CheckRGBcolordraped
#      *      *      *      *      *      *      *      *      *      *
CheckPseudocolordraped:
    if ($list_choice!="Pseudocolordraped") then goto CheckRGBcolordraped
    say status "Creating Density Slice (Pseudocolour) image...\n"
    if ($test_surface != "test_surface3") then goto newsurface0 else goto
cont_surface0
cont_surface0:

```

```

        add new surface
        last surface
        turn surface on
        set algorithm description "Pseudo colordraped"
        first layer
        set layer description "Pseudo"
        set layer dataset to $Dataset1
        set layer input 1 to band $band

        goto cont_oldsurface0
newsurface0:
        new window
        first surface
        new algorithm
        set algorithm description "Pseudo colordraped"
        first layer
        set layer description "Pseudo"
        set layer dataset to $Dataset1
        set layer input 1 to band $band
cont_oldsurface0:
        add intensity layer
        set layer description "intensity"
        set layer dataset to $Dataset2
        set layer input 1 to band $band_intensity
        set layer shading on
        if ($test_surface != "test_surface3") then goto DoNotSetTransparency1 else
goto SetTransparency1
SetTransparency1:
        first surface
        set surface transparency to 1
DoNotSetTransparency1:
        set view mode to 2d
        copy algorithm to window
        go window
        copy algorithm from window
            last surface
            first pseudo layer
            last transform
            set transform limits to actual
            first intensity layer
            next transform
            set transform limits to actual

```

```

        copy algorithm to window
        go window
    copy algorithm from window
        last surface
        first pseudo layer
        last transform
        set transform limits to 95.0
        first intensity layer
        next transform
        set transform limits to 99.9
if ($yesSmoothing == 1) then goto smoothing1 else goto no_smoothing1
smoothing1:
    set algorithm supersample type to bilinear
no_smoothing1:
    if ($yesnoPerspective ==1) then goto YesPerspective1 else goto
NoPerspective1
YesPerspective1:
    set view mode to perspective
    if ($test_surface != "test_surface3") then goto NoZOffset1 else goto
SetZOffset1
SetZOffset1:
    first surface
    set surface transparency to 0
    set surface Zoffset to 6000
    last surface
    set surface Zoffset to -6000
NoZOffset1:
NoPerspective1:
include "lib/Delete_All_Inactive_Layers.erb"
    copy algorithm to window
    go window
    $test_surface = "test_surface2"
    if ($Goto1 == "Finish11") then goto FinishUp
    $Goto = "Page1"
    goto ValidateData
#      *      *      *      *      *      *      *      *      *
CheckRGBcolordraped:
    if ($list_choice!="RGBcolordraped") then goto FinishUp
    say status "Creating RGB colordraped image...\n"
    if ($test_surface != "test_surface2") then goto newsurface else goto
cont_surface
newsurface:
    new window
    new algorithm

```


Chapter 13 Wizard:Colordrape images in multisurfaces ● 1: Setting up the wizard button

```
        goto cont_newsurface
cont_surface:
    add new surface
    last surface
    turn surface on
cont_newsurface:
    set algorithm description "RGB colordraped"
    add red layer
    set layer description "Red"
    set layer dataset to $Dataset1
    set layer input 1 to band $band1
    add green layer
    set layer description "green"
    set layer dataset to $Dataset1
    set layer input 1 to band $band2
    add blue layer
    set layer description "blue"
    set layer dataset to $Dataset1
    set layer input 1 to band $band3
    add intensity layer
    set layer description "intensity"
    set layer dataset to $Dataset2
    set layer input 1 to band $band_intensity
    set layer shading on
set view mode to 2d
    copy algorithm to window
    go window
    copy algorithm from window
# Cycle through all layers setting the transform limits to actual
    first active raster layer
    if ($ERROR !=0) then goto no_algorithm
    next_active_layer:
        last transform
        set transform limits to actual
        next active raster layer
        if ($ERROR ==0) then goto next_active_layer
no_algorithm:
# Copy algorithm to window and display the image
    copy algorithm to window
    go window
    copy algorithm from window
    first red layer
    last transform
```

```

        set transform limits to actual
        first green layer
    next transform
        set transform limits to actual
        first blue layer
    next transform
        set transform limits to actual
        first intensity layer
    next transform
        set transform limits to actual
        if ($test_surface != "test_surface2") then goto DoNotSetTransparency2 else
goto SetTransparency2
SetTransparency2:
    first surface
    set surface transparency to 1
DoNotSetTransparency2:
    last surface
    last algorithm
    set algorithm mode to rgb
# Copy algorithm to window and display the image
    copy algorithm to window
    go window
    copy algorithm from window
    first red layer
    last transform
    set transform clip to 95
    first green layer
    next transform
    set transform clip to 95
    first blue layer
    next transform
    set transform clip to 95
    first intensity layer
    next transform
    set transform clip to 99.9
    if ($yesnoPerspective ==1) then goto YesPerspective2 else goto
NoPerspective2
YesPerspective2:
    set view mode to perspective
    if ($test_surface != "test_surface2") then goto NoZOffset2 else goto
SetZOffset2
SetZOffset2:
    first surface
    set surface transparency to 0

```

```

        set surface Zoffset to 6000
        last surface
        set surface Zoffset to -6000
NoZOffSet2:
NoPerspective2:
if ($yesSmoothing == 1) then goto smoothing2 else goto no_smoothing2
smoothing2:
        set algorithm supersample type to bilinear
no_smoothing2:
include "lib/Delete_All_Inactive_Layers.erb"
        copy algorithm to window
        go window
        $test_surface = "test_surface3"
        if ($Goto2 == "Finish22") then goto FinishUp
        $Goto = "Page1"
        goto ValidateData
#      *      *      *      *      *      *      *      *      *      *
FinishUp:
        if ($list_choice=="Pseudocolordraped" or $list_choice=="RGBcolordraped")
then goto Done
        say warning "Please specify the type(s) of image to create."
        goto Page1
Done:
        say status "Done.\n"
        delete status
WizardCancel:
exit
#####

```

Displaying the pseudo colordrape image in 3-D perspective in the first surface

- 1 Save the batch script file as “JS_wizard_colordrape.erb” in the ‘ERMAPPER/batch’ directory
- 2 Exit the ER Mapper and call it again so that it will recognize your test1.bar toolbar file in the ‘ERMAPPER/config’ directory
- 3 On the ER Mapper main menu click on the toolbar menu. Your “test1.bar” will appear as “ test1” in the Toolbar menu in alphabetical order.

Tip: Your batch script file “**JS_wizard_colordrape.erb**” in the ‘ERMAPPER/batch’ directory should have the same name as the batch script (“**JS_RGB_541**” “Example BATCH densitysliced” BATCH “**JS_wizard_colordrape**”) in the “test1.bar” toolbar file. You can edit your batch script file and run it by clicking the button without exiting from ER Mapper. Only when you modify the toolbar file “test1.bar”, for the ER Mapper to recognize the modified toolbar file, it is necessary to exit from ER Mapper and call it again.

- 4 Click the “test1” on the Toolbar menu. Numerous buttons will be displayed on the Toolbar.
- 5 From the displayed buttons click on the JS_wizard_script wizard script button.
- 6 The **Pseudo & RGB colordrape - Data Entry** dialog box appears. This is the first wizard page.

Note: **Pseudo & RGB colordrape - Data Entry** is the title of the wizard and you can change it as you wish in the wizard script.

- 7 On the **Pseudo & RGB colordrape - Data Entry** dialog box select **Pseudocolordraped** option
- 8 On the **Pseudo & RGB colordrape - Data Entry** dialog box select “**Newcastle_Radiometric.ers**” dataset for the color layer from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 9 On the **Pseudo & RGB colordrape - Data Entry** dialog box select “**Newcastle_Magnetics.ers**” dataset for the intensity layer from the ‘ERMAPPER/dataset/Core_Datasets’ directory.
- 10 Select the **smoothing** option
- 11 Select the **3-D Perspective** option
- 12 Click the **Next** button
- 13 **Pseudo colordrape - Image Properties** dialog box appears.
- 14 **B1:Total_Count_cps** of the **Newcastle_Radiometrics.ers** for the color layer and **B1:magnetrics** of the **Newcastle_Magnetics.ers** for the intensity layer by default are selected.

Note: For dataset/s with more than one band, select the band/s of your choice

- 15 Click the Next button on the **Pseudo colordrape - Image Properties** dialog box.
- 16 The 3_D pseudo colordraped image is displayed

Note: By not selecting the 3-D Perspective option the pseudo colordraped image will be displayed in 2-D . If you click the Finish button the pseudo colordraped image will be displayed and quit the program.

Displaying the RGB composite colordrape image in 3-D perspective in the second surface

To create RGB colordraped image requires a RGB composite and an intensity layer. Generally, of a particular area, a DTM image used as an intensity layer is a separate dataset and TM image used in creating a RGB composite is another dataset. Similarly Magnetism used as an intensity layer in colordrapping is also a separate dataset and radiometrics used to create an RGB composite is another dataset. Hence, in creating RGB colordraped images normally requires two sets of data. This exercise is the extension of the above wizard scripting exercise and describes how to import two datasets, process and display them as pseudocolor and RGB colordraped images on two surfaces.

Note: You have clicked the **Next** button on the **Pseudo colordrape - Image Properties** dialog box and displayed the pseudo colordrape image in the first surface. By clicking the Next button you have decided to continue the process and create another colordrape image - RGB composite colordrape image- and display it in the second surface. Clicking the **Next** button on the **Pseudo colordrape - Image Properties** dialog box will take you back to the first wizard page for you to choose datasets for further processing.

- 1 The **Pseudo & RGB colordrape - Data Entry** dialog box will appear. This is the first wizard page.

Note: **Pseudo & RGB colordrape - Data Entry** is the title of the wizard and you can change it as you wish in the wizard script.

- 2 On the **Pseudo & RGB colordrape - Data Entry** dialog box select RGBcolordraped option.

- 3 On the **Pseudo & RGB colordrape - Data Entry** dialog box select **"Newcastle_Radiometric.ers"** dataset for RGB composite from the 'ERMAPPER/dataset/Core_Datasets' directory.
- 4 On the **Pseudo & RGB colordrape - Data Entry** dialog box select **"Newcastle_Magnetics.ers"** dataset for the intensity layer from the 'ERMAPPER/dataset/Core_Datasets' directory.
- 5 Select the **smoothing** option
- 6 Select the **3-D Perspective** option
- 7 Click the **Next** button on the **Pseudo & RGB colordrape - Data Entry** dialog box
- 8 **RGB colordrape - Image Properties** dialog box appears.
- 9 **B1:Total_Count_cps**, **B2:Potassium_cps** and **B3:Uranium_cps** of the **Newcastle_Radiometrics.ers** for the Red, Green and Blue layers of RGB composite and **B1:magnetics** of the **Newcastle_Magnetics.ers** for the intensity layer by default are selected.
- 10 From the band selection, select **B2:Potassium_cps** for **Red** layer, **B4:Thorium_cps** for **Green** layer and **B3:Uranium_cps** for **Blue** layer.
- 11 Click the **Finish** button on the **RGB colordrape - Image Properties** dialog box.
- 12 The 3_D pseudo colordraped and RGB colordraped images are displayed in the first and second surfaces respectively and quit the program.

Note: Images are displayed first in 2-D mode and then 3-D images are displayed

Note: The ZOffset is set for the first surface 6000 and the second surface - 6000. ZOffset between surfaces is proportional to the width of the dataset used. After the images are displayed you can call the Algorithm window and reset the ZOffset from the Surface Tab page.

Note: By not selecting the 3-D Perspective option the RGB colordraped image will be displayed in 2-D. In displaying the pseudo colordrape and RGB colordrape images in two surface in 2-D mode, 100% transparency is applied to the first surface image so that you can view the second image in the second surface. In the batch scripting language the transparency is setup in decimal. Therefore 1 is equal to 100% transparency. 0.5 is equal to 50% transparency. To adjust the transparency click the **View Algorithm for Image Window** button



on the main menu window. The Algorithm appears. On the Algorithm window click the Surface Tab and adjust the Transparency %.

Note: You can select and display RGB colordraped image first and then display Pseudo colordraped image in the second surface

Exercises:

(1) Write a wizard script to process a pseudocolordrape image and a RGB colordrape image and display the two images in two surfaces. Set up a wizard page for each image processing technique to select image properties . Use **ask action** command and conditional **If .. then .. else goto** and unconditional **goto** controls to navigate the processing path of each of the processing technique. You will need to set up options to choose either to display the processed image/s in 2-D or 3-D mode. If it is 3-D mode you need to set up ZOffset of the two processed images in the two surfaces.

Close all image windows and dialog boxes

- 1 Close all image windows using the window system controls:
 - For Windows, select **Close** from the window control-menu.
 - For Unix systems, press right mouse button on the window title bar, and select **Close** or **Quit** (for systems with both options, select **Quit**).

Only the ER Mapper main menu should be open on the screen.

What you learned...

After completing these exercises, you know how to perform the following tasks in ER Mapper:

- Create multi-page wizard.

- Write wizard scripts for pseudo colordrape and RGB colordrape algorithms and display them in two surfaces

Scripting reference

This chapter lists and describes all the operators, functions, variables, keywords and commands used in the ER Mapper batch scripting language.

Note: All variable names, label, and keywords are case insensitive.

Operators

The full suite of standard operators is available.

#	Specifies a comment line. For example, # this is a comment
= * / + - %	Standard arithmetic operators: assignment, multiplication, division, addition, subtraction, modulus
!= <> > >= < <= ==	Standard comparison operators: not equal to, not equal to, greater than, greater than or equal to, less than, less than or equal to, equal to
~	Specifies the number of decimal places to print for numbers. For example, print ~3 \$var prints \$var to 3 decimal places.
()	For assigning operator precedence.

[] Array specifier.

Concatenation

The Batch engine can add numbers to strings. Numbers are converted into text and joined to the end of strings.

Example

```
$number = 6
$string = "datasetname"
$string = $string + $number + ".ers"
```

Mathematical functions

<code>sin(x)</code> , <code>cos(x)</code> , <code>tan(x)</code>	The sine, cosine and tangent of the angle x, with the angle specified in radians.
<code>asin(x)</code> , <code>acos(x)</code> , <code>atan(x)</code>	The arcsine, arccosine and arctangent of the angle x, with the angle specified in radians.
<code>pow(x,y)</code>	The number x raised to the power of y.
<code>log(x)</code> , <code>exp(x)</code>	The natural log and exponent of x.
<code>sqrt(x)</code>	The square root of x.
<code>floor(x)</code>	The number x rounded down.
<code>ceil(x)</code>	The number x rounded up.
<code>min(x,y)</code> , <code>max(x,y)</code>	The minimum and maximum of two numbers x and y.
<code>abs(x)</code>	The absolute value of x.

Variables

Variables have a \$ leading character, followed by a letter and then alphanumeric characters (including underscores).

Syntax: \$variablename

Memory is allocated dynamically as assignments are made.

```
$var = 1
$var2 = $var
```

ER Mapper supports global name spaces for variables; i.e values set in included scripts affect parent scripts.

Memory is freed as variables are deleted or when the script exits.

There are a number of types. Once a variable has been defined, its type becomes fixed. For example, in the first statement below, the assignment `$a = 0` defines the variable `$a` as a number variable. The second statement `$a = "Hello"` tries to assign a string to the number variable and produces an error.

```
$a = 0
$a = "Hello"
```

The following types of variable are available.

Number

A floating point number. Numerical variables can use the full suite of arithmetic operators: `- * / + - %`.

For example,

```
$var = 1 + 1
```

is equivalent to

```
$var = 2
```

```
$var2 = $var * 100
```

is equivalent to

```
$var2 = 200
```

String

Strings are enclosed in quotes. Allowable arithmetic operators are: `++`

```
$hello = "Hello "
```

```
$world = "World"
```

```
$hello_world = $hello + $world
```

gives the result

```
"Hello World"
```

YesNo

Allowable values are:

- yes or 1
- no or 0

Mode

Allowable values are:

- pseudocolor (or pseudo)
- rgb
- hsi

LayerType

Allowable values are:

- pseudocolor

	<ul style="list-style-type: none">• red• green• blue• hue• saturation (or sat)• intensity (or int)• classification (or class)• classdisplay• link• height
CoordSys	Allowable values are: <ul style="list-style-type: none">• raw• en• ll
MosaicType	Allowable values are: <ul style="list-style-type: none">• overlay• feather
PageViewMode	Allowable values are: <ul style="list-style-type: none">• normal• layout
TransformType	Allowable values are: <ul style="list-style-type: none">• linear• exp• log• hist
ViewMode	Allowable values are: <ul style="list-style-type: none">• 2d• perspective• flythu
FilterType	Allowable values are: <ul style="list-style-type: none">• convolution• threshold• user
SuperSampleType	Allowable values are: <ul style="list-style-type: none">• nearest

References

- bilinear

You can also have string references to a:

- Window
- Algorithm
- Layer
- Formula
- Transform
- Filter

Color

A three part number variable which can be specified in two ways:

- As a set of three numbers which represent the red, green and blue components of the color. The syntax is:

```
$Color = [colorval] r,g,b
```

For example,

```
$Color = 124,5,5
```

```
$Color = colorval 124,5,5
```

- As a string containing one of the named colors listed in the ER Mapper color chooser. The syntax is:

```
$Color = colorval "known_color"
```

For example,

```
$Color = colorval "red"
```

CellType

Allowable values are:

- uint8
- uint16
- uint32
- int8
- int16
- int32
- ieee4
- ieee8

MachineType

Can be used for creating system command paths, etc.
Allowable values are:

- sun4
- sun5
- irix5
- decalpha
- hp
- win32

Arrays

You can have arrays of any type.

`$array[1]`

The array index can be a variable: `$array[$var]`. For example,

```
$count = 3
$filename[$count] = "vegetation"
```

Multi-dimensional arrays are supported. For example,

```
$array[cars][red] = 1
```

Note: You can specify the array index to start at any number, including a negative value. We do, however, recommend that you restrict this to 0 or 1.

Page size options

The page size options listed in the page setup dialog are special strings that are recognised by ER Mapper. They can be included in a listmenu chooser. For example,

```
$p_size_array[0] = "Custom"
$p_size_array[1] = "US Letter"
$p_size_array[2] = "US Legal"
$p_size_array[3] = "A1"
$p_size_array[4] = "A2"
$p_size_array[5] = "A3"
ask listmenu "Choose a page size from the list" "Page
size chooser" $p_size_array $p_size_choice
```

Keywords

ER Mapper supports the following keywords

above	absolute	action	active
actual	add	algorithm	all
annotation	as	ask	at
autovary_value	azimuth	background	band
bandchooser	bandmenu	batch	batch
begin	below	blue	border
bottom	build	cell	center
centre	class	classdisplay	classification
clip	close	color	color_blue
color_green	color_red	colorchooser	colorval
colour	colour_blue	colour_green	colour_red
colourchooser	colourval	cols	columns
constraints	container	contents	coord
coordinate	copy	count	current
dataset	datasets	datum	defaults
delete	description	dirname	down
duplicate	edit	editable	elevation
else	end	equalise	equalize
exists	exit	extension	extention
extents	false	file	fileext
filename	filter	first	fit
format	formula	free	from
gaussian	geolink	geoposition	get
getenv	go	goto	green
hardcopy	height	horizontal	hsi
hue	if	image	in

Chapter 14 Scripting reference ● Keywords

include	info	init	int
intensity	items	job	justify
last	layer	left	limits
link	listmenu	listmenu_exclusive	load
lut	lutmenu	main	match
matrix	mode	monocolor	monocolour
mosaic	move	name	new
newline	next	none	number
off	on	open	out
overview	page	parameters	params
path	point	pointer	postsampled
preference	preferences	previous	print
println	process	profile	program
projection	pseudo	pseudocolor	raster
realtime3d	red	relative	rgb
right	roam	roam	rows
sarturation	sat	save	say
scale	scattergram	screen	select
sep	separator	set	setenv
setup	shading	show	size
sizeX	sizey	sleep	space
split	status	sunangle	supersample
surface	system	tempname	text
then	threshold_value	timestamp	title
to	top	transform	transparency
traverse	true	truecolor	truecolour
turn	type	up	userfile
userfunc	vector	version_number	version_string
vertical	view	virtual	warning

width	width_pct	width_pixels	window
wizard	wizardpage	yesno	zoffset

Flow control

Labels

For labelling the program for ‘goto’s. (A ‘goto’ always refers to a label within the same file; i.e local name space. This means that ER Mapper can distinguish between a label in the main code and the same on in an included file. Labels can be used for conditional or unconditional control.

Syntax: labelname:

For example,

```
get_filenames:
```

Controls

goto Unconditional control. For example,

```
label1:
goto label1
```

if ... then ... else Conditional control. For example,

```
label1:
$var = 1
if ($var == 1) then goto label1
if ($var == 2) then goto label1 else
goto label2
```

Explicit exit

```
if ($var == 1) then goto carryon
exit
carryon:
```

Note: If you are running the batch script from the command line using **ermapper -b**, the script must exit with a return code > 0 (e.g. `exit 1`) to ensure that the ER Mapper application also exits. If the script exits with `exit` or `exit 0`, then the command line prompt will "hang" until you physically stop the ER Mapper application.

Looping

```
$count = 0
increment:
$count = $count + 1
if ($count <= 10) goto increment
```

Including files

Files must be specified by their absolute path or a path relative to the `ERMAPPER\batch` directory. Please note that ER Mapper supports forward (/) slashes for PC and Unix directory paths. It supports backward slashes (\) on PCs only. To maintain portability between platforms it is advisable to use forward (/) slashes.

Syntax: `include "filename"`

Example:

```
include "lib/BE_Startup.erb"
```

Error reporting

Some commands return an error code and text which are stored in the following variables:

`$ERROR` stores the error code

- 0 - successful
- 1 - unsuccessful

`$ERROR_TEXT` stores the error text message

Commands that return an error code include: `previous`, `next`, `first`, `last`, `load`, `save`

Script Commands - Alphabetical listing

This section lists all the script commands alphabetically, and describes their operation and syntax.

absolute|relative path

Specifies whether the variable returned by ‘ask file’ or ‘ask link’ will be an absolute filepath or relative to the current directory. The default is absolute.

Syntax: **absolute|relative path**

Example

```
relative path
ask file "File to contour:" "$dsname ".ers,.alg" $dsname
```

add (new) layer

Adds a new layer of the given type after the current layer in the current surface.

Syntax: **add \$stype|pseudocolor|red|green|blue|hue
|saturation|intensity|height|classification|classdisplay|link layer**

\$stype Layer type variable: see “Variables” on page 240 for allowed values.

Examples

```
add pseudocolor layer
```

This is the same as

```
new pseudocolor layer
add layer
```

add (new) transform (to layer)

Adds a transform of the given type after the current transform. If no type is specified, it defaults to linear.

Syntax: **add [\$ttype|linear|exp|log|hist] transform**

\$ttype Transform type variable: value can be linear, exp, log or hist

Examples

```
add transform
add linear transform
```

add filter (to layer)

Adds the current or specified filter to the current layer after the current filter.

Syntax: **add [\$ftype|convolution|threshold|user] filter|\$fil**

\$ftype Filter type variable: value can be convolution, threshold or user.

Examples

```
add $fil
add user filter
```

add formula (to layer)

Adds the current or specified formula to the current layer.

Syntax: **add formula|\$for**

Example

```
add $form1
```

add layer (to surface)

Adds the current or specified layer to the current surface, at the end of the layer list.

Syntax: **add layer|\$lay**

Examples

```
add layer
add $new_layer
```

add surface (to algorithm)

Adds the specified surface to the current algorithm.

Syntax: **add \$srf**

Example

```
add $add_surface
```

add transform (to layer)

Adds the specified transform after the current transform in the current layer.

Syntax: **add \$tra**

Example

```
add $another_transform
```

add transform point

Adds a point to the current or specified transform at x, y.

Syntax: **add transform|\$tra point \$x \$y**

\$x \$y Numbers, representing point coordinates

Example

```
add transform point 20 40
```

add transform to layer input

Adds the specified transform after the current transform in the given stream input of the current layer.

Syntax: **add \$tra to layer input \$count**

\$count Number representing layer input number

Example

```
$count = 2
add $tral to layer input $count
add $tral to layer input 2
```

algorithm2 = algorithm1

Sets \$alg2 to refer to \$alg1.

Syntax: **\$alg2 = \$alg1**

Note: They refer to the same object, so deleting one of them will cause them both to become invalid. Use the copy command to create separate objects.

ask action

Defines a button and the action to be carried out if it is selected by a user.

Syntax:	Ask Action "<i>Button_Name</i>" [[Close] Goto <i>Label</i>]
"<i>Button_Name</i>"	The text to appear on the button.
Close	(Optional). Closes the wizard before carrying out the 'Goto Label'.
Goto <i>Label</i>	(Optional). Specifies the label to go to in the script if the user presses the button. If omitted, the button is made inactive and shaded out.

Example

The following define the buttons used on the first page of a multi-page wizard:

Note: In the above example, the "Back" button will be greyed out because no ***GoTo Label*** is specified.

Similarly, an action button can be put inside the user area of a form, for example:

```
Ask Action "Configure ..." Goto Configure_form
```

There is a 'Close' parameter on the 'Ask Action' command, as well as an explicit "Wizard Close" command. They are designed with different circumstances in mind. The 'Close' parameter on the 'Ask Action' command is primarily designed for quick error pop-up windows, which generally report an error to the user, then pop down, and go straight back to the page that needs more data entered.

ask bandchooser

Creates a chooser dialog which lists the descriptions for all the bands in the specified file.

Syntax: **ask bandchooser** "*prompt text*" "*title*" *p_dsh_name* [*single_band_flag* *consec_flag*] *\$bandchoice*

"prompt text" The text which appears to the left of the drop down list.

"title" Title of the chooser dialog.

p_dsh_name The dataset header file (.ers) name.

single_band_flag (Optional) Specifies the number of bands that can be selected. Can be either:

- 0 - multiple bands may be selected
- 1 - only one band may be selected

If it is not included it is set to 0 by default. If it is included the *consec_flag* must also be included. If it is set to 1, the *consec_flag* should be set to 0.

consec_flag (Optional) Automatically selects the both the real and imaginary bands when one or the other is selected. This is designed specifically for specifying bands for FFT. Can be either:

- 0 - has no effect
- 1 - select real and imaginary bands for FFT

If it is not included it is set to 0 by default. If it is included the *single_band_flag* must also be included. If it is set to 1, the *single_band_flag* should be set to 0.

Note: *single_band_flag* and *consec_flag* are optional, but must be used together (i.e. either none or both must be specified)

\$band_choice The variable into which the selected band or bands are stored as a string.

Examples:

```
ask bandchooser "Band Menu" "Band Menu Chooser" $filename1 0 1
$BANDSTRING
```

```
ask bandchooser "Band Menu" "Band Menu Chooser" $filename1 $BANDSTRING
```

Note: This stores the choice as a string e.g. 1-3,5,7 and is intended for use with FFT and classification wizards (i.e. wizards that run an executable which takes a band list as an input parameter).

ask bandmenu

Creates a drop down list of the descriptions for all the bands in the specified file. The band list in the process diagram in the Algorithm dialog is an example of this.

Syntax:	ask bandmenu " <i>prompt text</i> " <i>p_dsh_name</i> <i>\$bandchoice</i>
"prompt text"	The text which appears to the left of the drop down list.
p_dsh_name	The dataset header file (.ers) name.
\$bandchoice	The band number of the chosen band.

Example:

```
ask bandmenu "My menu" $filename $band_choice
```

ask colorchooser|ask colourchooser

Adds a color chooser button that will open the standard color chooser.

Syntax:	ask colorchooser " <i>prompt text</i> " " <i>title</i> " <i>\$color</i> ask colourchooser " <i>prompt text</i> " " <i>title</i> " <i>\$color</i>
"prompt text"	The text which appears to the left of the drop down list.
"title"	The title of the chooser.
\$color	The ColorType variable in which the name of the chosen color is stored.

Example

```
ask colorchooser "Choose a color" "My color title" $colorchoice
```

ask datum

Adds a file chooser button that will open the standard datum chooser.

Syntax:	ask datum " <i>label</i> " <i>\$datum_name</i>
"label"	The title of the chooser.
\$datum_name	A variable in which the chosen datum from the list is stored.

ask directory

Adds a button that will open a directory chooser.

Syntax: **ask directory "label" \$dir_name**

"label" The title of the chooser.

\$dir_name A variable in which the chosen directory is stored. You can preset this as a default directory.

Example

```
$dir_name = "C:\ermapper\examples"  
ask directory "Directory:" $dir_name
```

ask file

Adds a file chooser button to the alphanumeric entry field.

Syntax	ask file "prompt text" "default_directory" "default_file" ".ext" \$file_name
"prompt text"	The text appears to the left of the entry box. If you don't want a prompt string use empty quotes: ""
"default_directory"	(Optional). The default directory to show in the file chooser, relative to the default directory for the file type. Omit this field if you specify a default file.
"default_file"	(Optional). A default file to show in the file chooser. Use the absolute path name or path relative to the default directory for the file type.
".ext"	A comma separated list of file extensions of the files to be listed in the file chooser. Specify ".ALLRASTER" if you want to list all image files supported by ER Mapper.
\$file_name	A text variable for storing the file name selected using the file chooser.

Example:

```
ask file "Input algorithm or dataset:" ".ers,.alg" $alg_name
```

Note: You can only use the “default_file” parameter to specify .ers files. To specify another file type as a default, initialize the \$file_name variable with the default file name. For example:

```
$outfile = "c:\temp\output.cc8"
ask file "Output dataset: " " " ".cc8" $outfile
```

ask gridlayermenu

Creates a drop down selection list of the descriptions for all the layers in the specified gridding project file.

Syntax: **ask gridlayermenu** "*prompt text*" *\$project_file*
\$layerchoice

"prompt text" The text which appears to the left of the drop down list.

\$project_file String containing the path and name of the gridding project file.

\$layerchoice The layer number of the chosen layer.

Example:

```
ask gridlayermenu "Grid Layer:" $grid_project_filename  
$grid_layer
```

ask hardcopy

Asks the user to specify a hardcopy device. The two different types of drivers that are available on the pc platform are erm_driver (ermapper hardcopy control files) and win32_driver (win32 printer drivers). The driver_type option is ignored for unix platforms (erm_driver is used regardless of the driver type).

Syntax: **ask hardcopy** erm_driver|win32_driver "*prompt text*"

erm_driver Add button to open the Default Hardcopy chooser

win32_driver Add button to open the Windows Print Setup dialog.

"prompt text" The text which appears above the entry box.

Example

```
ask hardcopy win32_driver "Please specify a printer driver"
```

ask link

Adds a dynamic link chooser button to the alphanumeric entry field.

Syntax	ask link "prompt text" chooser_program \$var
"prompt text"	The text that appears to the left of the entry box. If you don't want a prompt string use empty quotes:""
chooser_program	The dynamic link chooser program to run. This field is equivalent to the sixth parameter in the dynamic link menu file. See sections “Menu entry parameters” on page 233 and “Link chooser parameter” on page 235.
\$var	A variable for storing the chosen data.

Example:

```
ask link " " "$CHOOSER=arc_chooser $DEFAULT" $ws_file
```

ask listmenu

Adds a list containing entries of any defined type.

Syntax: **ask listmenu** "*prompt text*" "*title*" *\$array_name*
\$choice

"prompt text" The text which appears to the left of the drop down list.

"title" The title of the chooser.

\$array_name The name of the array which holds the choices to be listed in the chooser.

\$choice A variable in which the chosen item from the list is stored.

You can only select one option. You can have lists of the following types (or a mixture of them):

- String
- Value
- LayerType
- Algorithm Mode
- ViewMode
- Coordinate Space Type
- Mosaic Type
- CellType
- TransformType
- SuperSampling Type
- Filter Type
- Color
- Page Constraint Type

The page constraint information is the same as that used in the Page setup dialog.

Most items are listed in the choosers exactly as they are defined in the \$array_name elements. However, in cases when the resulting lists would not be easy to understand some other property is used instead. The cases this applies to and the property that is listed in the chooser for each case are shown below.

<u>Variable type</u>	<u>Property listed in chooser</u>
Window/Destination	title
Algorithm	name
Stream	desc
Formula	name
Filter	name
Surface	name

You can't have lists of transforms.

If the index for the pointer is null, the item will not show up in the list.

Example

```
$ARRAY_VARIABLE[1] = red
$ARRAY_VARIABLE[2] = green
$ARRAY_VARIABLE[3] = blue
...
# set the default
$ARRAY_VARIABLE_CHOICE = $ARRAY_VARIABLE[2]
ask listmenu "Choose a layer type from the list" "Layer Chooser"
$ARRAY_VARIABLE $ARRAY_VARIABLE_CHOICE
```

ask listmenu_exclusive

The same as **ask listmenu**, but all the chooser entries are listed in the dialog with exclusive radio buttons (not a drop down list). Thus, this is best used with small numbers of options only.

Syntax:	ask listmenu_exclusive " <i>prompt text</i> " <i>\$Array_name</i> <i>\$choice</i>
"prompt text"	The text which appears to the left of the drop down list.
\$array_name	The name of the array which holds the choices to be listed in the chooser.
\$choice	A variable in which the chosen item from the list is stored.

ask lutmenu

Adds a list of color lookup tables to choose from and stores the selection in a string variable.

Syntax:	ask lutmenu " <i>prompt text</i> " <i>\$lut_choice</i>
"prompt text"	The text which appears to the left of the list.
\$lut_choice	The name of the chosen lookup table is stored in a string variable.

Example

```
ask lutmenu "Color Lookup Table" $lut_choice
```

ask projection

Adds a file chooser button that will open the standard projection chooser.

Syntax:	ask projection " <i>label</i> " <i>\$proj_name</i>
"label"	The title of the chooser.
\$proj_name	A variable in which the chosen projection from the list is stored.

ask text|number

Adds an alphanumeric entry box to the container.

Syntax**ask text "prompt text" \$text_input****ask number "prompt text" \$number_input****"prompt text"**

The text appears to the left of the entry box. If you don't want a prompt string use empty quotes: ""

\$text_input

A text variable.

\$number_input

A number variable.

Example:

```
ask text "Please enter your name:"$name
```

ask yes/no

Adds a check box to the container.

Syntax:**absolute|relative path****"prompt text"**

The text appears to the left of the check box. If you don't want a prompt string use empty quotes: ""

\$yesno_input

A Yes/No variable.

Example:

```
ask yesno "Center Horizontally" $do_center_horiz
```

build absolute filespec

Builds the absolute file specification from a given relative file specification and its parent directory

Syntax:**build absolute filespec <parent_dir> <rel_file>****parent_dir**

The path and name of the parent directory

rel_file

The name and path of the file relative to parent_dir

Example

```
println build absolute filespec "c:/ermapper/examples"
"Data_Types\\Airphoto\\RGB.alg"
```

build file path

Builds a complete path for a file from up to 4 given elements by inserting the correct file separators.

Syntax: **build file path** *<element_1>* [*<element_2>* [*<element_3>* [*<element_4>*]]]

element_ The path element.

Example

```
println build file path "c:\\ermapper\\examples" "Data_Types"
"Airphoto" "RGB.alg"
```

build relative filespec

Builds file specification relative to its parent directory from a given absolute file specification.

Syntax: **build relative filespec** *<parent_dir>* *<abs_file>*

parent_dir The path and name of the parent directory

abs_file The name and absolute path of the file

Example

```
println build relative filespec "c:/ermapper/examples"
"c:\\ermapper\\examples\\Data_Types\\Airphoto\\RGB.alg"
```

Color keywords

You can use the **color_red**, **color_green** and **color_blue** keywords to get the rgb (0-255) components of a color variable.

Example

```
$bg_color = 230,23,56
$red = get $bg_color color_red
$blue = get $bg_color colour_blue
```

container right|left justify

This is used to justify a button or a row of buttons in a container. It will not affect other item types such as lists and files as these are placed so as to make best use of the existing space.

Syntax: [Container] right|left justify

Example:

```
container right justify
  ask action "< Back"
  ask action "Next >"
```

container above|below|left|right

Optional command specifies how this container is to be placed in relation to another container. The default is for the container to be below the previously defined container.

Syntax: [Container] Above|Below|Left|Right "Name"

“Name” String with name of other container

Example:

```
container below "con1"
```

container begin|end

A container block defines the size, contents and layout of a single ‘container’ or ‘pane’ in a wizard page. Any number of containers can be defined though realistically there will be only two or three per page.

Syntax**container begin “Name”****.....****container end****"Name"**

Name is the name of the container within the script. Each of the containers on a single page (within a single Wizard block) must have a different name.

Example:

```
container begin "con2"
  container height_pct 20
  container below "con1"
    ask action "< Back"
    ask action "Next >" goto wizard_page2
    ask action "Cancel" close goto wizard_page2
  container end
```

container items

Optional command specifies the direction of placement of the *items in a container* defined between this command and the next Container Items command. If omitted, the items will be placed vertically.

Syntax:**[Container] [Items] Horizontal | Vertical | Newline****Examples**

```
Container Items Horizontal
Horizontal
```

container labels

Optional command specifies where the labels for an item in the container will appear. The default is ‘above’.

Syntax:**[Container] Labels_above | Labels_left****Example:**

```
Container labels_above
```

container width|height

Optional command specifies the container width and height as a percentage of the remaining space.

Syntax: **[Container] width_pct | height_pct
\$Percentage_number**

\$Percentage_number Number, representing percentage value.

Examples

```
Container width_pct 30
width_pct 30
```

convert file separator

Converts the file separators in a given file specification from English to native and vice versa.

Syntax: **convert <filespec> to english|native sep|separator**

filespec File specification to be converted.

Example

```
$fspec = "c:/ermapper/examples/Data_Types/Airphoto/RGB.alg"
convert $fspec to native sep
```

copy algorithm

Copies the current or specified algorithm. The current algorithm pointer is updated to point to the new algorithm.

Syntax: **copy algorithm|\$alg**

Examples

```
copy algorithm to window
copy $alg to window
$alg_copy = copy $alg # Note: not the same as $alg_copy = $alg
$alg_copy = copy algorithm
```

copy algorithm from window

Copies the algorithm from the current or specified window to the batch engine and set the current algorithm pointer to point to it.

Syntax: **copy algorithm from window|\$win**

Examples

```
copy algorithm from window
copy algorithm from $win
```

copy algorithm to window

Copies the specified algorithm or that indicated by the current algorithm pointer to the current window in ER Mapper. Aborts if there is no current window

Syntax: **copy algorithm|\$alg to window**

Examples

```
copy $original_algorithm to window
copy algorithm from window
```

copy filter

Makes a copy of the current or specified filter but does not insert the new filter into a layer.

Syntax: **copy filter|\$fil**

Example

```
$fil = copy filter
```

copy formula

Makes a copy of the current or specified formula, but does not add it to a layer.

Syntax: **copy formula|\$for**

Example

```
$for2 = copy formula
```

copy layer

Copies the current or specified layer but does not insert it into a surface.

Syntax: **copy layer|\$lay**

Examples

```
$temp_layer = copy layer
```

copy surface

Copies the current or specified surface, but does not add the new surface to any algorithm

Syntax: **copy surface|\$srf**

Example

```
$new_surface = copy $srf1
```

copy transform

Makes a duplicate of the current or specified transform but does not insert it into the current algorithm.

Syntax: **copy transform|\$tra**

Examples

```
copy $tral  
$tra2 = copy $tral
```

copy window

Makes a duplicate of the current or specified window and its algorithm, and then runs the algorithm to display the duplicate. The current window pointer is updated to point to the new window.

Syntax: **copy window|\$win**

Examples

```
copy window  
copy $win
```

current algorithm

Sets the pointer to the current algorithm. Can also be used to check whether an algorithm currently exists

Syntax: `current algorithm`

Examples

```
$curr_alg = current algorithm  
if current algorithm then goto have_algorithm
```

current filter

Sets the pointer to the current filter.

Syntax: `current filter`

Example

```
$fil = current filter
```

current formula

Sets the pointer to the current formula.

Syntax: `current formula`

Example

```
$for1 = current formula
```

current layer

Sets the pointer to the current layer. This can also be used to check whether a current layer exists.

Syntax: `current layer`

Examples

```
$layer2 = current layer  
if current layer then goto layer_ok
```

current surface

Sets the pointer to the current surface.

Syntax: **current surface**

Example

```
$curr_surface = current surface
```

current transform

Sets the pointer to the current transform.

Syntax: **current transform**

Example

```
$tral = current transform
```

current window

Sets pointer to the current window. This can also be used to check whether an image window currently exists.

Syntax: **current window**

Examples

```
$win = current window  
if current window then goto window_ok
```

delete algorithm

Deletes all references to an algorithm. If the algorithm deleted was the current algorithm, the current algorithm pointer will be set to point to the next algorithm if one exists, or the previous one, or aborts.

Syntax: **delete algorithm|\$alg**

Examples

```
delete algorithm  
delete $change_algorithm
```

delete directory|file

Delete a directory or file. If the file name has an “.ers” extension, the raster file is also deleted.

Syntax:	delete \$dir \$file
\$dir	String with path and directory name
\$file	String with path and file name

Example

```
$temp_file = "examples/tempfile.ers"  
delete $temp_file
```

delete filter

Deletes the filter and all references to it.

Syntax:	delete filter \$fil
----------------	----------------------------

Example

```
delete filter
```

delete formula

Deletes the current formula in the current layer.

Syntax:	delete formula
----------------	-----------------------

Example

```
delete formula
```

delete layer

Deletes the current or specified layer. When all layers in a surface are deleted, the surface will also be deleted unless it is the only surface on the algorithm.

Syntax:	delete layer \$lay
----------------	---------------------------

Examples

```
delete $layer1  
delete layer
```

delete status (dialog)

Deletes the status dialog. Note that there is only one status dialog for the batch engine.

Syntax: **delete status**

Example

```
delete status
```

delete surface

Deletes the current or specified surface from the current algorithm.

Syntax: **delete surface|\$srf**

Examples

```
delete $add_surface  
delete surface
```

delete transform

Deletes the current or specified transform, and all references to it.

Syntax: **delete transform|\$tra**

Example

```
delete $tral
```

delete window

Deletes the current or specified window and its algorithm. If the one deleted was current it updates the current window and current algorithm to the next window and next algorithm. If there is no next window it sets them to the previous window and previous algorithm.

Syntax: **delete window|\$win**

Examples

```
delete window  
delete $win
```

duplicate filter

Copies the current filter, and inserts the copy after the current filter.

Syntax: **duplicate filter**

Examples

```
duplicate $fil  
$fill = duplicate filter
```

duplicate formula

Duplicates the current formula, and inserts the copy after the current formula.

Syntax: **duplicate formula**

Examples

```
$form1 = duplicate formula  
duplicate layer
```

duplicate layer

Duplicates the current layer in the current surface. The new layer is inserted after the current layer.

Syntax: **duplicate layer**

Examples

```
$lay1 = duplicate layer  
duplicate layer
```

duplicate surface

Duplicates the current or specified surface within the current algorithm.

Syntax: **duplicate surface|\$srf**

Examples

```
$new_surface = duplicate surface  
duplicate surface
```

duplicate transform

Copies the current transform in the current layer, and inserts the copy after the current transform.

Syntax: **duplicate transform**

Example

```
duplicate transform
```

edit file

Edit the given file name in a text editor. The file is created if it doesn't exist.

Syntax: **edit \$filename**

\$filename String with path and text file name, including extension.

Example

```
$toolbarname = "c:\ermapper\batch\toolbar.erb"  
edit $toolbarname
```

exists

Verify that a specified file exists.

Syntax: **\$filename exists**

\$filename String with path and name of file.

Example

```
if "c:\temp\temp.alg" exists then goto file_exists
```

exit

Exit the batch process and return to ER Mapper.

Syntax: **exit** [**\$exit_no**]

\$exit_no Integer with return code. Only used if you are running the batch script from the command line using **ermapper -b**. The script must exit with a return code > 0 (e.g. `exit 1`) to ensure that the ER Mapper application also exits. If the script exits with `exit` or `exit 0`, then the command line prompt will "hang" until you physically stop the ER Mapper application.

Example

```
if ($var == 1) then goto carryon
exit
carryon:
```

File name, extension and path keywords

You can use the **filename**, **dirname** and **fileext** keywords to respectively return the file name, path and extension of a given file specification. If there is no extension, filename or path component it returns a null string ""

Examples

```
$string = "\examples\Shared_Data\airphoto.ers"
$fname = filename $string# returns "airphoto.ers"
$dirname = dirname $string# returns "examples\Shared_Data"
$ext = fileext $string# returns ".ers"
```

Note: If `$string = "c:\",` then `dirname $string` would return `"c:\"`; i.e it retains the trailing slash for volumes.

Note: If `$string = "/examples/Shared_Data/airphoto.ers"` then `dirname $string` would return `"examples/Shared_Data"`.

filter2 = filter1

Assigns specified filter (e.g. \$fil1) to new variable (e.g. \$fil2)

Syntax: **\$fil2 = \$fil1**

Note: \$fil2 and \$fil1 refer to the same object, so deleting \$fil1 will invalidate \$fil2. Use the copy or duplicate command to create a new object.

Example

```
$fil2 = $fil1
```

first|last|next|previous algorithm

Changes the current algorithm pointer to point to the first, last, next or previous algorithm.

Syntax: **first|last|next|previous algorithm**

Examples

```
next algorithm
```

```
$alg2 = last algorithm
```

first|last|next|previous filter

Sets the current filter to the first, last, next or previous filter in the current layer. Optionally selects an input filter, and/or a filter type.

Syntax: **first|last [input \$n] [\$ftype] filter**
next|previous [\$ftype] filter

\$n Number, representing input number

\$ftype Filter type variable: value can be convolution, threshold or user.

Example

```
last input 2 filter
```

```
next threshold filter
```

first|last|next|previous formula

Sets the current formula to the first, last, next or previous formula in the current layer.

Syntax: **first|last formula**
 next|previous formula

Example

```
last formula
next formula
```

first|last|next|previous layer

Sets the current layer pointer to point to the specified layer in the current surface. Sets \$ERROR to 1 if fails; otherwise 0.

Syntax: **first|last|next|previous layer**

Examples

```
$layer1 = first layer
first layer
```

first|last|next|previous transform

Sets the current transform to the first, last, next or previous transform in the current layer. Optionally select an input transform, and/or a transform type.

Syntax: **first|last [input \$n] [\$ttype] transform**
 next|previous [\$ttype] transform

\$n Number representing layer input number

\$ttype Transform type variable: value can be linear, exp, log or hist

Examples

```
last transform
$lasttran = last input 2 transform
next linear transform
$nextlin = next linear transform
```

first|last|next|previous window

Updates the current window pointer to point to the oldest, newest, next oldest or next newest window open. Sets \$ERROR to 1 if fails; otherwise 0.

Syntax: **first|last|next|previous window**

Examples

```
next window
```

```
$winlast = last window
```

first|last|next|previous surface

Selects a surface within the current algorithm and makes it current.

Syntax: **first|last|previous|next surface**

Examples

```
first surface
```

```
next surface
```

fit page to hardcopy

Sets the page width and height of the current or specified algorithm to that of the currently specified hardcopy device.

Syntax: **fit [\$alg] page to hardcopy**

Example

```
fit $alg page to hardcopy
```

format value precision

Formats a number to have a specified number of digits after the decimal point.

Syntax: **format \$input_value \$precision**

\$input_value Number, representing original data to be formatted

\$precision Number, representing required number of digits after the decimal point.

Example

```
$arg1_formatted = format $template_tl_e_extent 3
```

get (page) contents topleft|bottomright

Gets the page contents extents coordinates.

Syntax: **get [\$alg] contents topleft|bottomright
northings|latitude|meters_y**

**get [\$alg] contents topleft|bottomright
eastings|longitude| meters_x**

Example

```
$template_tl_e_extent = get $alg contents topleft eastings
```

get algorithm coordsys

Gets the algorithm's coordinate system type.

Syntax: **get algorithm|\$alg coordsys**

Examples

```
$coordsys = get algorithm coordsys
```

```
$coordsys = get $alg coordsys
```

get algorithm datum|projection

Gets the algorithm's datum or projection type.

Syntax: `get algorithm|$alg datum|projection`

Examples

```
$datum = get algorithm datum
```

```
$proj = get $alg projection
```

get algorithm description

Gets the current or specified algorithm's description.

Syntax: `get algorithm|$alg description`

Examples

```
$desc = get algorithm description
```

```
$desc = get $alg description
```

get algorithm layer count

Gets the number of layers in the current or specified algorithm.

Syntax: `get algorithm|$alg layer count`

Examples

```
$count = get algorithm layer count
```

```
$count = get $new_alg layer count
```

get algorithm lut (color table)

Gets the algorithm's pseudocolor LUT name.

Syntax: `get algorithm|$alg lut`

Example

```
$alg_lut = get algorithm lut
```

get algorithm mode

Gets the algorithm's mode. Returns "pseudo", "rgb" or "hsi".

Syntax: `get algorithm|$alg mode`

Examples

```
$algorithm_mode = get algorithm mode  
if (get algorithm mode == pseudo) then goto newmode
```

get algorithm mosaic type

Gets the algorithm's mosaic type. Returns overlay or feather.

Syntax: `get algorithm|$alg mosaic type`

Example

```
$mtype = get algorithm mosaic type
```

get algorithm topleft|bottomright (extents)

Gets the algorithm's extents. Returns number.

Syntax: `get algorithm|$alg topleft|bottomright eastings|
longitude|meters_x
get algorithm|$alg topleft|bottomright northings|
latitude|meters_y`

Example

```
$extent1 = get algorithm bottomright longitude
```

get algorithm units|rotation

Gets the algorithm's units or rotation. Rotation is in decimal degrees, units is a units string.

Syntax: `get algorithm|$alg units|rotation`

Examples

```
$alg_units = get algorithm units  
$alg_rotate = get $alg rotation
```

get English file separator

Returns the standard English file separator used by the host workstation or PC. This would be “\” on a Win32 (PC) platform or “/” on a Unix platform.

Syntax: **get english file sep|separator**

Example

```
println get english file sep
```

get file size

Gets the given files size in bytes. \$filename is an absolute path name.

Syntax: **get \$filename size**

\$filename String with path and file name

Example

```
$filename = "c:\ermapper\examples\myfile.alg"
$filesize = get $filename size
```

get filter description

Gets the current or specified filter’s description.

Syntax: **get filter|\$fil description**

Examples

```
$desc = get filter description
```

```
$desc = get $fil1 description
```

get (usercode) filter params

Gets the current or specified user filter’s parameters.

Syntax: **get filter|\$fil params**

Example

```
$param = get $filter1 params
```

get filter postsampled (flag)

Gets the current or specified filter's post sampled process flag. Returns boolean value.

Syntax: **get filter|\$fil postsampled**

Example

```
$fpost = get filter postsampled
```

get filter rows|cols

Gets the number of rows/columns in the current or specified filter.

Syntax: **get filter|\$fil rows|cols**

Examples

```
$frow = get filter rows
```

```
$fcol = get $fill cols
```

get filter scale|threshold (value)

Gets the scale or threshold for the current or specified filter. Valid only on convolution and threshold filters.

Syntax: **get filter|\$fil scale|threshold**

Example

```
$value = get filter threshold
```

get filter userfile (filename)

Gets the current or specified user filter source file name. Valid only on usercode filters.

Syntax: **get filter|\$fil userfile**

Example

```
$fnme = get filter userfile
```

get (usercode) filter userfunc (filename)

Gets the current or specified user filter function (.dll) name. Valid only on usercode filters.

Syntax: **get filter|\$fil userfunc**

Example

```
$funcname = get filter userfunc
```

get filter type

Gets the current or specified filter's type. Returns convolution, threshold or user.

Syntax: **get filter|\$fil type**

Example

```
$ftype = get filter type
```

get free space (on disk)

Gets the amount of free space, in bytes, on the given disk.

Syntax: **get \$diskname free space**

\$diskname String with path and directory name

Example

```
$diskname = "c:\ermapper\examples"  
$free_space = get $diskname free space
```

get layer azimuth|elevation

Gets the layer sunshading azimuth or elevation. Returns degrees as a number.

Syntax: **get layer|\$layazimuth|elevation**

Example

```
$angle = get layer elevation.
```

get layer band count

Gets the number of bands in the current or specified layer.

Syntax: `get layer|$lay band count`

Example

```
$band_count = get layer band count
```

get layer band description

Gets the nominated band description of the current or specified layer.

Syntax: `get layer|$lay band $n description`

\$n Number, representing band number

Example

```
$band_desc = get layer band 1 description
```

get layer cell sizex|sizey

Reads the current or specified layer cell horizontal or vertical size.

Syntax: `get layer|$lay cell sizex|sizey`

Examples

```
$cell_sizex = get $layer1 cell sizex
```

```
$cell_sizey = get layer cell sizey
```

get layer cell type

Gets the current or specified layer's image cell type.

Syntax: `get layer|$lay cell type`

Example

```
$lay_celltype = get layer cell type
```

get layer color

Gets the current or specified layer color. Valid only on link layers.

Syntax: **get [layer|\$lay] color**

Examples

```
$layer_color = get color
$layer_color = get $layer color
```

get layer coordinates

Returns the EN or LL layer coordinates from the given cellX and cellY values.

Syntax: **get layer x_coordinate y_coordinate from \$cellX \$cellY**

\$cellX \$cellY Integers with X and Y cell coordinate values

Examples

```
$tlx = get layer x_coordinate from $StartColumn $StartRow
$tlx = get layer y_coordinate from $StartColumn $StartRow
$brx = get layer x_coordinate from $EndColumn $EndRow
$bry = get layer y_coordinate from $EndColumn $EndRow
```

get layer dataset (filename)

Gets the current or specified layer's image name.

Syntax: **get layer|\$lay dataset**

Examples

```
$dsname = get layer dataset
if (get layer dataset != "") then goto make_cdrape
```

get layer description

Gets the current or specified layer's description.

Syntax: **get layer|\$lay description**

Example

```
$lay_desc = get $layer1 description
```

get layer editable (flag)

Gets the current or specified layer's editable flag. Valid only on link layers. Returns true or false.

Syntax: **get layer|\$lay editable**

Example

```
$edit = get layer editable
```

get layer edit|init program (name)

Gets the current or specified layer's edit|init program name.

Syntax: **get layer|\$lay edit|init program**

Example

```
$init_program = get layer init program
```

get layer formula

Gets a copy of the current or specified layer's formula. Returns a string with the formula.

Syntax: **get layer|\$lay formula**

Example

```
$form = get $layer1 formula
```

get layer input filter count

Gets the number of input filters in the current or specified layer input.

Syntax: **get layer|\$lay input \$index filter count**

\$index Number representing input number.

Example

```
$count = get layer input $index filter count
```

get layer link extension

Gets the link file extension string for the current or specified layer.

Syntax: **get layer|\$lay link extension**

Examples

```
$l_ext = get $layer1 link extension
```

```
if (get layer link extension == ".erv") then goto process_vector
```

get layer output transform count

Gets the number of output transforms in the current or specified layer.

Syntax: **get layer|\$lay output transform count**

Example

```
$count = get $layer1 output transform count
```

get layer shading

Gets the current or specified layer's shading value. Returns on or off

Syntax: **get layer shading**

Example

```
$shaded = get layer shading
```

get layer type

Gets the current or specified layer's type. Returns a layer type value. See "Variables" on page 240.

Syntax: `get layer|$lay type`

Example

```
$ltype = get layer type
```

get layer|\$lay input count

Gets the number of inputs in the current or specified layer.

Syntax: `get layer|$lay input count`

Example

```
$count = get layer input count
```

get native path|file separator

Get the path or file separator used natively by the host workstation or PC.

Syntax: `get native path|file sep|separator`

Example

```
println get native path separator  
println get native file sep
```

get page constraints

Get the page constraints of the current or specified algorithm. Returns zoom, page, border or scale.

Syntax: `get [$alg] page constraints`

Example

```
$page_constraint = get $alg page constraints
```

get page scale

Gets the page scale of the current or specified algorithm.

Syntax: `get [$alg] page scale`

Example

```
$page_scale = get $alg page scale
```

get page size

Gets the page size of the current or specified algorithm. Returns string with page size; e.g. “US Letter”.

Syntax: `get [$alg] page size`

Example

```
$page_size = get $alg page size
```

get page topleft|bottomright (extents)

Gets the page extents coordinates.

Syntax: `get [$alg] page topleft|bottomright
northings|latitude|meters_y
get [$alg] page topleft|bottomright
eastings|longitude| meters_x`

Example

```
$tl_e_extent = get $alg page topleft eastings
```

get page top|bottom|left|right border

Gets the page borders of the current or specified algorithm.

Syntax: `get [$alg] page top|bottom|left|right border
mm|inches`

Example

```
$page_top = get $alg page top border inches
```

get (algorithm) page view mode

Gets the page view mode (normal or page layout) of the current or specified algorithm.

Syntax: **get [\$alg] page view mode**

Example

```
$pvmode = get page view mode
```

get page width|height

Gets the specified page parameter of the current or specified algorithm,

Syntax: **get [\$alg] page width|height inches|mm**

Example

```
$page_width = get $alg page width mm
```

get preference

Gets the value of the preference name from the preference file. Returns specified default value if entry does not exist.

Syntax: **get preference "*name*"
\$default_string|\$default_number[boolean|integer|double]**

"*name*" String with name of entry in preference file. Suggested standard format is "Class:Name:Variable."

\$default_string String with default value for entry

\$default_number Number with default value for entry

Example

```
$ImageVersion = get preference "Wizard:Image:Version" 1.0
```

get preference (color)

Gets the color preference entry in the user's preference file. Set to specified default value if the entry does not exist.

Syntax:	get preference "colorname" \$red \$green \$blue get preference "colorname" colorval \$default_color \$red,\$green,\$blue
"colorname"	Color preference entry name
\$red \$green \$blue	RGB numeric values for the default color.
\$default_color	Default color specification in the form of a string (e.g. "red"), an RGB triple (e.g. 255,0,0 for red), or a variable which has been set to a color specification.

Examples

```

$pref_color = get preference "my_color_pref" $default_color
$pref_color = get preference "my_color_pref" colorval
$default_color
$pref_color = get preference "my_color_pref" colorval "green"
$pref_color = get preference "my_color_pref" colorval 100,203,240
$pref_color = get preference "my_color_pref" 100 203 240
$pref_color = get preference "my_color_pref" 100,203,240

```

get surface description

Gets the current or specified surface description.

Syntax:	get surface \$srf description
----------------	--------------------------------------

Examples

```

$desc = get surface description
$desc = get $srf1 description

```

get surface layer count

Gets the number of layers in the current or specified surface.

Syntax: **get surface|\$srf layer count**

Examples

```
$layers = get surface layer count
```

```
$layers = get $srf layer count
```

get transform (limits)

Gets the current or specified transform's limits. Returns a number.

Syntax: **get transform|\$tra input|output min|max**

Example

```
$inmax = get transform input max
```

get transform type

Gets the current or specified transform type. Returns linear, exp, log or hist.

Syntax: **get transform|\$tra type**

Example

```
$trantype = get transform type
```

get (algorithm) view mode

Gets the view mode of the current or specified algorithm.

Syntax: **get [\$alg] view mode**

Example

```
$vmode = get view mode
```

get (ER Mapper) version

Returns the ER Mapper version number as a string or as a number.

Syntax: **get version_string|version_number**

Examples

```
$version_string = get version_string# returns "5.7"
```

getenv (environment variable)

Get the given environment variable. This is from the current environment within ER Mapper.

Syntax: **getenv \$envname**

\$envname String with name of environment variable

Examples

```
$machine_type = getenv "ERM_MACHINE_TYPE"
```

```
$println "ERMScripts =" + getenv "ERMScripts"
```

go algorithm

Runs the current or specified algorithm. This causes histograms etc. to be updated.

Syntax: **go algorithm|\$alg [width height][match]**

width height Resolution at which to run algorithm (default 400 400)

match If specified, runs a histogram match.

Examples

```
go algorithm
```

```
go algorithm 50 50
```

```
go $alg1 100 100
```

```
go algorithm 400 400 match
```

go background window

Runs the algorithm in the current or specified window as a background task.; i.e performs a non-blocking go on the window.

Syntax: **go background window|\$win**

Examples

```
go background window
go background $win
```

go window

Runs the algorithm in the current or specified window.

Syntax: **go window|\$win**

Examples

```
go window
go $win
```

layer active

Checks whether the current layer is active, and returns TRUE (1) for active and FALSE (0) for inactive

Syntax: **layer active**

Examples

```
$layer_is_on = layer active
```

layer2 = layer1

Assigns specified layer (e.g. \$lay1) to another variable (e.g. \$lay2)

Note: \$lay2 and \$lay1 refer to the same object, so deleting \$lay1 will invalidate \$lay2. Use copy or duplicate layer to have different objects.

Syntax: **\$lay2 = \$lay1**

Example

```
$lay2 = $lay1
```

listdir

Lists the contents of the specified directory into an array, with each element being one file. If a recursive listing is made, then the resultant strings will contain the full path, otherwise they will just be the filename.

Syntax:	listdir \$directory [\$extension] [recursive]
\$directory	String defining the directory to list
\$extension	Optional string limiting the list to files with a specific extension (eg. "*.ers")
recursive	Specify this keyword to descend recursively into sub-directories, and list all files in the directory tree. This returns filenames with their full path.

Examples

```
$array = listdir "e:\images" ".ers" recursive
```

load algorithm

Loads the given algorithm. Sets the current algorithm pointer to point to it. Sets \$ERROR to 1 if it fails; otherwise 0.

Syntax:	load algorithm \$name
\$name	Algorithm filename; e.g. myalg.alg. Must be specified relative to the current directory or include the absolute path.

Examples

```
$template_alg_name = "c:\ERMAPPER\examples\templates\some.alg"
$template_alg_name = load algorithm $template_alg_name

$myfile = "..\..\examples\tutorial\my.alg"
load algorithm $myfile
```

load filter (filename)

Loads the current or specified filter from the given (.ker) file. The file name must include the absolute or relative path.

Syntax: **load filter|\$fil \$fname**

\$fname String with path and name of filter (.ker) file

Examples

```
$fname = "filters\myfilter.ker"
load filter $fname

load $fil "filters\myfilter.ker"
```

load formula (filename)

Loads the current or specified formula file.

Syntax: **load formula|\$for from \$fname**

\$fname String, with path and name of formula (.frm) file

Examples

```
load formula from "ratio\clay_ratio.frm"

$for = load formula from "ratio\clay_ratio.frm"
```

load layer formula (filename)

Loads the formula file into the current or specified layer's formula.

Syntax: **load layer|\$lay formula \$fname**

\$fname String, with formula filename (.frm) and path.

Example

```
$form = "ratio\clay_ratio.frm"
load $layer1 formula $form
```

match transform [to \$layer]

Matches the output transforms of all layers to the current or specified layer in the same surface.

The algorithm must already contain output transforms for the layers being matched. Use the slower **go algorithm match** command if the transforms do not exist.

Syntax: **match transform [to \$layer]**

Examples

```
$temp_layer = first active raster layer
match transform to $temp_layer

first active raster layer
match transform
```

move layer

Moves the current or specified layer within the current surface to the given position within its surface.

Syntax: **move layer|\$lay [to] up|down|top|bottom**

Examples

```
move layer to top
move $layer up
```

move surface

Moves the current surface within the current algorithm.

Syntax: **move surface up|down|top|bottom**

Example

```
move surface bottom
```

new algorithm

Creates a new (empty) algorithm. This will have 1 pseudocolor layer. The current algorithm pointer is updated to point to the new algorithm

Syntax: **new algorithm**

Examples

```
new algorithm
$alg = new algorithm
```

new filter

Creates a new filter, but does not add it to any layer.

Syntax: **new \$ftype|convolution|threshold|user filter**

\$ftype Filter type variable: value can be convolution, threshold or user.

Example

```
$fill = new convolution filter
```

new formula

Creates a new formula.

Syntax: **new formula**

Example

```
$for1 = new formula
```

new layer

Creates a new layer of the given type. This creates a floating layer which does not belong to any surface in any algorithm. To add it to a surface use add layer. This could be useful if you want to create a layer and duplicate it and change the processing before adding it.

Syntax: **new \$layer_type|pseudocolor|red|green|blue|hue|saturation| intensity|height|classification|classdisplay|link layer**

\$layer_type Layer type variable: see “Variables” on page 240 for allowed values.

Examples

```
$ltype = height
new $ltype layer

new pseudocolor layer

$new_layer = new pseudocolor layer
```

new surface

Creates a new surface but does not add it to any algorithm.

Syntax: **new surface**

Examples

```
new surface

$add_surface = new surface
```

new transform

Creates a new transform which then becomes the current transform.

Syntax: **new \$ttype|linear|exp|log|hist transform**

\$ttype Transform type variable: value can be linear, exp, log or hist

Examples

```
new linear transform

%tran2 = new linear transform
```

new window

Opens a new image window, with a default algorithm. The current window pointer points to the new window. The current algorithm pointer points to the new algorithm. You can specify the position and size of the window.

Syntax: **new window [x y w h]**

x, y The x,y coordinate of the top left corner of the window in screen pixels.

w, h The width and height of the window in screen pixels.

Examples

```
new window
$win = new window 0 0 600 600
new window $Xoffset $Yoffset $windowwidth $windowheight
```

next [active][raster|vector] layer (type)

Moves the pointer to the next layer of the type specified. With layers there are the additional keywords ‘next’ and ‘active’.

Syntax: **next [active][raster|vector] [\$layer_type] layer**

\$layer_type Layer type variable: see “Variables” on page 240 for allowed values.

Examples

```
next layer
next active layer
next active raster layer
next active vector layer
next red layer
```

open status (dialog)

Creates (if necessary) and opens the status dialog. This is useful for when the user has pressed the close button on the status dialog. If "title" is included it is used as the title of the progress dialog.

The wizard GUI may have a button labelled **Status**.

Syntax: **open status ["title"]**

"title" The text which appears in the title bar of the status dialog box.

Example

```
...
ask action "Status" goto OpenStatus
...
OpenStatus:
open status
goto wizard_page_1
```

open window

Open the designated dialog box on screen. If the dialog box is iconised, it will be un-iconised.

Syntax: **open main|algorithm|transform|filter|formula|
sunangle|page
setup|annotation|defaults|preferences|
geoposition|processinfo|datasetinfo|scattergram|
traverse|realtime3d|job|profile|coordinate window**

Example

```
open sunangle window
```

print

Write to an output dialog (default) or file without a Carriage Return or Line Feed.

Syntax: **print** “*String*”|*Number*/\$*var*

\$var: Variable with string or number. By default, numbers are printed to 6 decimal places.

The print and println commands write to an output dialog by default. Alternatively, you can print out to a file using:

```
set output to $filename
```

The print or println commands in a batch script after this set command will create the file if it doesn’t already exist and write the output to the end of the file.

Example

```
$text="Hello World"
print $text

print "Hello World"
```

println

Write to an output dialog (default) or file with a Carriage Return or Line Feed.

Syntax: **println** “*String*”|*Number*/\$*var*

\$var: Variable with string or number. By default, numbers are printed to 6 decimal places.

The print and println commands write to an output dialog by default. Alternatively, you can print out to a file using:

```
set output to $filename
```

The print or println commands in a batch script after this set command will create the file if it doesn’t already exist and write the output to the end of the file.

Example

```
$text="Hello World"
println $text

println "Hello World"
```

save algorithm

Saves the current or specified algorithm with the given name. Sets \$ERROR to 1 if fails; otherwise 0.

Syntax:**save algorithm|\$alg \$name****\$name**

Algorithm filename; e.g. myalg.alg. Must be specified relative to the current directory or include the absolute path.

Examples

```
$alg_name = "c:\ERMAPPER\examples\templates\some.alg"
save algorithm $alg_name
save $alg $alg_name
```

save algorithm as dataset

Saves the current or specified algorithm as a dataset, with the name stored in the variable \$dsname. If the optional parameters are not set, the default values which would display in the Save as Dataset dialog will be used.

Syntax:	save algorithm \$alg as dataset \$dsname [\$celltype \$null_value \$nr_cells \$nr_lines]
\$dsname	String representing the path and name of dataset (.ers) file.
\$cell_type	Cell type variable: value can be uint8, uint16, uint32, int8, int16, int32, ieee4 or ieee8. e.g \$cell_type = uint16 (not “uint16”)
\$null_value	String representing the null_value to be used. It may be set to “none”.
\$nr_cells	Number representing the number of columns of the image to include.
\$nr_lines	Number representing the number of rows of the image to include.

Examples

```

save $alg as dataset $dsname

$celltype = ul6int
save $alg as dataset $dsname $celltype $null_value $nr_cells
$nr_lines

save as dataset $dsname

save algorithm as dataset $dsname $celltype $null_value $nr_cells
$nr_lines

save $alg as dataset $dsname $celltype "none" $nr_cells $nr_lines

```

save algorithm as virtual dataset

Saves the current or specified algorithm as a virtual dataset with the given name. Sets \$ERROR to 1 if it fails; otherwise 0.

Syntax: **save algorithm|\$alg as virtual dataset \$name**

\$name Virtual dataset filename; e.g. myvds.ers. Must be specified relative to the current directory or include the absolute path.

Examples

```
$vds_name = "c:\ERMAPPER\examples\tutorial\somevds.ers"
save algorithm as virtual dataset $vds_name

save $alg as virtual dataset $vds_name
```

save filter (filename)

Saves the current or specified filter to the given (.ker) file.

Syntax: **save filter|\$fil \$fname**

\$fname String with path and name of filter (.ker) file

Example

```
$fname = "filters\myfilter.ker"
save filter $fname
```

save formula (filename)

Saves the current or specified formula to the formula file.

Syntax: **save formula|\$for to \$fname**

\$fname String, with path and name of formula (.frm) file

Example

```
save formula to "ratio\clay_ratio.frm"
```

save layer formula (filename)

Saves the current or specified layer's formula to the formula file.

Syntax: **save layer|\$lay formula \$fname**

\$fname String with formula filename (.frm) and path.

Example

```
save $layer1 formula $form
```

say status

Updates the status dialog with a message (if string) or updates the percent done gauge to number (between 0 and 100%) or both.

Syntax: **say status \$number|\$string**

say status \$number \$string

\$number Number, representing percentage done

\$string String with status message

Examples

```
$percent = 80
say status $percent "Opening image window\n"
say status 100 "Finished"
```

say warning

Opens an ER Mapper warning dialog box

Syntax: **say warning "text"**

Example

```
checkdataset:
if ($filename!= "") then goto DatasetOK
    say warning "Please enter a filename"
    goto wizard_page_1
DatasetOK:
#carry on
```

select algorithm

Changes the current algorithm pointer to point to the specified algorithm.

Syntax: `select $alg`

Examples

```
select $window_alg
```

select filter

Sets the current filter to the specified filter.

Syntax: `select $fil`

Example

```
select $fil
```

select formula

Sets the current formula to the specified formula.

Syntax: `select $for`

Example

```
select $for1
```

select layer

Sets the current layer pointer to point to the specified layer.

Syntax: `select $lay`

Example

```
select $layer1
```

select surface

Selects the given surface and makes it the current surface.

Syntax: `select $srf`

Example

```
select $window_surface
```

select transform

Sets the current transform to the specified transform.

Syntax: `select $tra`

Example

```
select $transl
```

select window

Updates the current window pointer to point to the given window.

Syntax: `select $win`

Example

```
select $win
```

set (formula) input to band

Sets the current formula input to the image band given.

Syntax: `set input $n1 to band $n2`

\$n1 \$n2 Numbers, representing input and band numbers

Example

```
set input 1 to band 1
```

set (page) contents bottomright from topleft

Calculates and sets the bottom right contents extents coordinate, based on the topleft coordinate, page size, borders, and scale. This functionality makes the implementation of map templates much easier because you do not need to change the page layout to accommodate a different image.

Syntax: `set [$alg] contents bottomright from topleft`

Example

```
set $alg contents bottomright from topleft
```

set (page) contents extents to algorithm extents

Set the contents extents to the algorithm extents (where **algorithm** is the current algorithm). This is equivalent to the "Snapshot" button on the Page Setup dialogue.

Syntax: `set contents extents to [algorithm|$alg] extents`

Example

```
set contents extents to $alg extents
```

set (page) contents topleft|bottomright

Sets the page contents extents coordinates to that specified by \$value.

Syntax: `set [$alg] contents topleft|bottomright
northings|latitude|meters_y [to] $value`
`set [$alg] contents topleft|bottomright
eastings|longitude|meters_x [to] $value`

\$value Number, representing extent value

Example

```
set $alg contents topleft eastings to $template_tl_e_extent
```

set page autovary_value

This can be used to calculate one of the page setup variables based on the Constraints setting. For example:

- When the constraint is "Fixed Page: Extents from Zoom", this command sets the scale to the largest that will fit on the page.
- When the constraint is "Auto Vary: Page", this command sets the page size to whatever is needed to fit in the desired scale and borders.
- When the constraint is "Auto Vary: Borders", this command sets the bottom and/or right border sizes to whatever is needed to accommodate the specified scale and page size.
- When the constraint is "Auto Vary: Scale", this command sets the scale to whatever is needed to create a map with the specified page and border sizes.

Syntax: `set [$alg] page autovary_value`

Example

```
set page autovary_value
```

set algorithm background (color)

Changes the algorithm's background color to the given RGB values or to a value specified by a variable.

Syntax: **set algorithm|\$alg background to \$red \$green \$blue**
set [algorithm|\$alg] background to colorval
\$colorvariable

\$red \$green \$blue RGB numeric values for the required color.

\$colorvariable Color specification in the form of a string (e.g. "red"), an RGB triple (e.g. 255,0,0 for red), or a variable which has been set to a color specification.

Examples

```
set algorithm background to 250 245 50

$red = 200
$green = 150
$blue = 30
set $alg background to $red $green $blue

set algorithm background to colorval 250 245 50

set background to color "red"

$background_color = "white"
set $alg background to colorval $background_color
```

set algorithm coordsys

Sets the algorithm's coordinate system type to the given type.

Syntax: **set algorithm|\$alg coordsys to \$csys|raw|en|ll**
\$csys Coordsys type variable: Value can be raw, en or ll

Examples

```
$csystem = en
set algorithm coordsys to $csystem

set $alg coordsys to en
```

set algorithm datum|projection

Sets the algorithm's datum or projection to the given type. This will cause layers of an incompatible type to be turned off within the algorithm.

Syntax: **set algorithm|\$alg datum to \$datumname**
 set algorithm|\$alg projection to \$projname

\$datumname String representing required datum

\$projname String representing required projection.

Examples

```
$datum = "NAD27"  
set algorithm datum to $datum  
  
set algorithm datum to "NAD27"  
  
set $alg projection to "NUTM11"
```

set algorithm description

Changes the current or specified algorithm description to the given text.

Syntax: **set algorithm|\$alg description to \$text**

\$text Algorithm description text string

Examples

```
$text = "This algorithm"  
set $alg description to $text  
  
set algorithm description to "Pseudocolor Aspect"
```

set algorithm lut (color table)

Changes the pseudocolor LUT for the first surface in the algorithm to the given lut file.

Syntax:	set algorithm \$alg lut [to] \$lutname
\$lutname	String representing color lookup table filename. It should be within the ERMAPPER\lut directory and specified within quotes but without the file extension.

Examples

```
$lut = "greyscale"
set $alg lut to $lut
set algorithm lut to "greyscale"
```

set algorithm mode

Changes the processing mode for the first surface in the algorithm to the given mode.

Syntax:	set algorithm \$alg mode to \$mode pseudo rgb hsi
\$mode	String variable representing required mode, “pseudo”, “rgb” or “hsi”

Examples

```
set algorithm mode to rgb
$alg_mode = "hsi"
set $alg mode to $alg_mode
```

set algorithm mosaic type

Sets the mosaic type (where different datasets overlay) to the given type. Overlay overwrites earlier layers with latter ones; Feather blends the border.

Syntax: **set algorithm|\$alg mosaic type to \$mtype|overlay|feather**

\$mtype Mosaic type variable: overlay or feather

Examples

```
$mtype = feather
set algorithm mosaic type $mtype
set algorithm mosaic type to feather
```

set algorithm supersample type

Sets the current or specified algorithm supersample type.

Syntax: **set algorithm|\$alg supersample type to \$supertype|nearest|bilinear**

\$supertype SuperSample type variable: value can be nearest or bilinear

Examples

```
$supertype = bilinear
set algorithm supersample type to $supertype
set $alg supersample type to nearest
```

set algorithm topleft|bottomright (extents)

Sets the current or specified algorithm topleft and bottomright extent fields in easting/northing, latitude/longitude or raw coordinate systems.

Syntax: **set algorithm|\$alg topleft|bottomright
eastings|longitude|meters_x to \$number**

**set algorithm|\$alg topleft|bottomright
northings|latitude|meters_y to \$number**

\$number Number, representing extents value.

Examples

```
$tl_east = 475912.476235
set algorithm topleft eastings to $tl_east
set algorithm topleft eastings to 475912.476235
```

set algorithm units|rotation

Sets the units or rotation to the given value.

Syntax: **set algorithm|\$alg units to \$units**
set algorithm|\$alg rotation to \$value

\$units String representing units: “meters”, “feet”, “yards”, “links”, “chains”, “roods”, “breal units” or “natural”

\$value Numerical value representing rotation in degrees

Examples

```
$units = "natural"
set algorithm units to $units
set $alg units to "natural"
$rotate = 0
set $alg rotation to $rotate
set algorithm rotation to 0
```

set filter description

Changes the current or specified filter description to the given text.

Syntax: **set filter|\$fil description to \$text**

\$text Filter description text string

Examples

```
$text = "This filter"
set $fil description to $text

set filter description to "High_pass"
```

set filter matrix (element)

Sets an element of the current or specified filter matrix. Valid only on convolution and threshold filters.

Syntax: **set filter|\$fil matrix \$x \$y \$number**

\$x \$y Numbers, representing coordinates of element in matrix

\$number Number, value to set in element

Examples

```
$number = 4
set filter matrix 1 1 $number

set filter matrix 1 1 1
```

set (usercode) filter params

Set the current or specified user filter parameter string. Valid only on usercode filters.

Syntax: **set filter|\$fil params to \$paramstring**

\$paramstring String, with filter parameters

Example

```
$params = "$SYS_STDOUT"
set filter params to $params
```

set filter postsampled (flag)

Sets whether the current or specified filter can process resampled data, or source data.

Syntax: `set filter|$fil postsampled true|false`

Example

```
$set filter postsampled false
```

set filter rows|cols

Sets the number of rows/columns for the current or specified filter to \$number. \$number must be odd.

Syntax: `set filter|$fil rows|cols [to] $number`

\$number Number, representing number of rows or columns.
Must be odd

Example

```
$filrows = 3
set filter rows to $filrows
set $fill cols to 3
```

set filter scale|threshold

Sets the scale or threshold for the current or specified filter. Valid only on convolution and threshold filters.

Syntax: `set filter|$fil scale|threshold [to] $value`

\$value Number, representing scale or threshold value

Example

```
$value = 1
set filter scale to $value
set filter scale to 1
```

set filter type

Sets the filter type of the current or specified filter.

Syntax: **set filter|\$fil type [to] \$ftype|convolution|threshold|user**

\$ftype Filter type variable: value can be convolution, threshold or user.

Example

```
$filtype = threshold
set filter type to $filtype

set $fil1 type to threshold
```

set (usercode) filter userfile (filename)

Sets the current or specified user filter source file, for a C usercode filter. Valid only on usercode filters.

Syntax: **set filter|\$fil userfile [to] \$filename**

\$filename String with path and name of filter source (.c) file

Example

```
$fname = "usercode\kernel\c\newfilter.c"
set filter userfile to $fname

set $fil2 userfile to "usercode\kernel\c\newfilter.c"
```

set (usercode) filter userfunc (filename)

Sets the current or specified user filter function (.dll) name. Valid only on usercode filters.

Syntax: **set filter|\$fil userfunc to \$funcname**

\$funcname String with path and name of filter function (.dll) file

Examples

```
$funcname = "usercode\kernel\c\win32\newfilter.dll"
set filter userfunc to $funcname

set $fil2 userfunc to "usercode\kernel\c\win32\newfilter.dll"
```

set formula

Sets the current or specified formula (\$formula is a string).

Syntax: **set formula|\$for to \$formula**

\$formula String, with formula

Examples

```
$formula = "i1 + i2"  
set formula to $formula  
  
set $for1 to "i1 + i2"
```

set layer azimuth|elevation

Sets the sun shade azimuth/elevation to the given value in degrees.

Syntax: **set layer|\$lay azimuth|elevation to \$deg**

\$deg Number, representing azimuth or elevation in degrees.

Example

```
set layer azimuth to 45
```

set layer color

Sets the current or specified layer color to the given RGB or color values. Valid only on link layers.

Syntax:	set layer \$lay color to \$red \$green \$blue set layer color to colorval \$colorvariable
\$red \$green \$blue	RGB numeric values for the required color.
\$colorvariable	Color specification in the form of a string (e.g. "red"), an RGB triple (e.g. 255,0,0 for red), or a variable which has been set to a color specification.

Examples

```

set layer color to 1.0 1.0 1.0
set $lay1 color to 1.0,1.0,1.0
$lay_color = "red"
set layer color to colorval $lay_color
set $layer1 color to colorval "red"
set layer color to colorval 123,155,100

```

set layer dataset (filename)

Sets the image file name for the current or specified layer to \$dsname.

Syntax:	set layer \$lay dataset to \$dsname
\$dsname	String specifying the image name (within quotes), including the file extension. The path should be either absolute or relative to the current directory

Example

```

$dsname = "examples\Shared_Data\testlayer.ers"
set $layer1 dataset to $dsname

```

set layer description

Changes the layer description to the text given.

Syntax: **set layer|\$lay description to \$text**

\$text String with layer description

Examples

```
$lay_desc = "First red"
set layer description to $lay_desc
set $layer1 description to "Second red"
```

set layer editable (flag)

Sets the editable flag for current or specified layer. Valid only for link layers.

Syntax: **set layer|\$lay editable [to] true|false**

Example

```
set layer editable to true
```

set layer edit|init program (name)

Sets the current or specified layer's edit/init program to the given name. Valid only on link layers.

Syntax: **set layer|\$lay edit|init program to \$fname**

\$fname String, with program name.

Examples

```
set layer edit program to "erm_link"
set layer init program to "ermininit_oracle"
```

set layer formula

Sets the current or specified layer formula.

Syntax:	set layer \$lay formula to \$formula
\$formula	String, with formula.

Examples

```
$form = "i1 + i2"
set layer formula to $form

set layer formula to "i1 + i2"
```

set layer input to band

Sets the designated layer input to the specified band number

Syntax:	set layer \$lay input \$n1 to band \$n2
\$n1 \$n2	Number

Examples

```
set layer input 1 to band 1

set layer $input_no to band $band_no
```

set layer link extension

Sets the link file extension of the current or specified layer to the given string.

Syntax:	set layer link extension to \$extent
\$extent	String with link extension. This field can take three forms:
""	no extension.
".erv"	a normal file extension.
"erv_chooser"	a program to generate a list of choices from which the user must choose one.

Example

```
set layer link extension to ".erv"
```

set layer link type to monocolour|truecolour

Sets the link type of the current or specified layer to monocolour or truecolour. Valid only on link layers. Note the English spelling of ‘colour’.

Syntax: `set layer|$lay link type to monocolour|truecolour`

Example

```
set layer link type to truecolour
```

set layer shading on|off

Turns sunshading on/off for the current or specified layer.

Syntax: `set layer|$lay shading on|off`

Example

```
set layer shading on
```

set layer type

Changes the current or specified layer to the given type.

Syntax: `set layer|$lay type to $stype|pseudocolor|red|green|blue|hue|saturation|intensity|height|classification|classdisplay|link`

\$stype Layer type variable: see “Variables” on page 240 for allowed values.

Examples

```
$lay_type = hue
set layer type to $lay_type

set $layer1 type to green
```

set page center horizontal|vertical

Centers the contents of the page horizontally or vertically.

Syntax: `set [$alg] page center horizontal|vertical`

Example

```
set $alg page center horizontal
```


set page constraints

Sets the page constraints of the current or specified algorithm.

Syntax: **set [\$alg] page constraints to \$constr|zoom|page|border|scale**

\$constr Constraints variable: value can be zoom, page, border or scale

Examples

```
$page_constraint = border
set $alg page constraints to $page_constraint

set page constraints to scale
```

set page extents from contents

Calculates and sets the page extents of the algorithm based on the contents extents, the borders, and the scale.

Syntax: **set [\$alg] page extents from contents**

Example

```
set $alg page extents from contents
```

set page scale

Sets the page scale to a specific number or the maximum scale possible.

Syntax: **set [\$alg] page scale to \$scale|max**

\$scale Number, representing scale value

Examples

```
$page_scale = 1000
set $alg page scale to $page_scale

set $alg page scale to max

set page scale to 1000
```

set page size

Sets the page size of the current or specified algorithm.

Syntax: **set [\$alg] page size to \$size**

\$size String with standard page size: value can be one of the following:
"US Letter", "US Letter - landscape",
"US Legal", "US Legal - landscape",
"A0", "A0 - landscape",
"A1", "A1 - landscape",
"A2", "A2 - landscape",
"A3", "A3 - landscape",
"A4", "A4 - landscape",
"A5", "A5 - landscape",
"A6", "A6 - landscape",
"B3", "B3 - landscape",
"B4", "B4 - landscape",
"B5" or "B5 - landscape"

Examples

```
$page_size = "US Letter"
set $alg page size to $page_size

set page size to "US Letter"
```

set page topleft|bottomright (extents)

Sets the page extents coordinates to that specified in the variable \$value.

Syntax: **set [\$alg] page topleft|bottomright
northings|latitude|meters_y [to] \$value**

**set [\$alg] page topleft|bottomright
eastings|longitude| meters_x [to] \$value**

\$value Number, representing extent value

Example

```
set $alg page topleft eastings to $tl_e_extent
```

set page top|bottom|left|right border

Sets the page borders.

Syntax: **set [\$alg] page top|bottom|left|right border to \$number mm|inches**

\$number Number, representing size in inches or millimeters

Examples

```
$page_top = 1.0
set $alg page top border to $page_top inches

set page top border to 1.0 inches
```

set (algorithm) page view mode

Sets the page view mode of the current or specified algorithm to normal or layout.

Syntax: **set [\$alg] page view mode to \$pvmode|normal|layout**

\$pvmode Page view mode variable: value can be normal or layout

Example

```
set $alg page view mode to layout
```

set page width|height

Sets the specified page size parameter of the current or specified algorithm.

Syntax: **set [\$alg] page width|height [to] \$number inches|mm**

\$number Number, representing size in inches or millimeters

Examples

```
$page_width = 5
set $alg page width to $page_width inches

set $alg page width to 5 inches
```

set pointer mode

Sets the mouse pointer to the specified mode: zoom, zoombox, roam(hand) or pointer.

Syntax: **set pointer mode to zoom|zoombox|roam|pointer**

Examples

```
set pointer mode to zoom
```

```
set pointer mode to zoombox
```

```
set pointer mode to roam
```

set preference

Adds or updates the preference ‘name’ entry in the user’s preference file as a string or number.

Syntax: **set preference "*name*" [to]
\$string\$number[boolean|integer|double]**

“*name*” String with name of entry in preference file. The recommended format is “Class:Name:Variable” .

\$string String with value for entry

\$number Number with value for entry

Examples

```
set preference "Wizard:Image:MainChoice" $page_main_choice
```

```
set preference "MACHINECONFIG_HAS-RUN" TRUE boolean
```

set preference (color)

Sets the color preference entry in the user's preference file.

Syntax:	set preference "<i>colorname</i>" to \$red \$green \$blue set preference "<i>colorname</i>" to colorval \$color \$red,\$green,\$blue
"<i>colorname</i>"	Color preference entry name
\$red \$green \$blue	RGB numeric values for the required color.
\$color	Color specification in the form of a string (e.g. "red"), an RGB triple (e.g. 255,0,0 for red), or a variable which has been set to a color specification.

Examples

```
set preference "my_color_pref" to 155 203 200
set preference "my_color_pref" to 155,203,200
set preference "my_color_pref" to colorval $new_color
set preference "my_color_pref" to colorval "green"
set preference "my_color_pref" to 155,203,200
```

set surface description

Changes the current or specified surface description to the given text.

Syntax:	set surface \$srf description to \$text
\$text	Surface description text string

Examples

```
$text = "This surface"
set $srf1 description to $text
set surface description to "Main surface"
```

set surface lut (color table)

Sets the Color Table for the current surface.

Syntax: **set surface lut [to] \$lut**

\$lut String representing color lookup table filename. It should be within the %ERMAPPER%\lut directory and specified within quotes but without the file extension.

Examples

```
set surface lut to $new_lut
set surface lut to "pseudocolor"
```

set surface mode (color)

Sets the color mode for the current surface.

Syntax: **set surface mode [to] \$surface_mode|rgb|hsi|pseudo**

\$surface_mode Mode type variable: value can be rgb, hsi or pseudocolor (or pseudo)

Examples

```
$new_mode = rgb
set surface mode to $new_mode
set surface mode to pseudo
```

set surface name

Sets the current surface name to the given name.

Syntax: **set surface name [to] \$string**

\$string String with surface name.

Examples

```
$surface_name = "Top surface"
set surface name to $surface_name
set surface name to "Bottom surface"
```

set surface zscale|zoffset|transparency

Sets the Z Scale, Z Offset or transparency of the current surface to \$value.

Syntax: **set surface zscale|zoffset|transparency [to] \$value**
\$value Number representing required value.

Example

```
set surface transparency to 20
```

set transform clip

Applies a clip of \$pct percent to the current or specified transform. If \$pct is not specified 99.0 is used.

Syntax: **set transform|\$tra clip [to \$pct]**
\$pct Number, representing percentage clip

Example

```
set transform clip to 90
```

set transform input|output min|max (limits)

Sets the current or specified transform input/output limits.

Syntax: **set transform|\$tra input|output min|max to \$number**
\$number Number, representing input/output limits

Examples

```
$inmax = 361  
set transform input max to $inmax  
set transform input max to 7
```

set transform limits (actual or percentage)

Sets the current or specified transform limits to \$percent or actual.

Syntax: **set transform|\$tra limits to actual|\$percent**
\$percent Number, representing percentage

Example

```
set transform limits to actual
set transform limits to 50
```

set transform to gaussian equalize

Applies a gaussian equalize operation to the current or specified transform. If the gaussian envelope halfwidth is not specified a halfwidth of 3 standard deviations is used.

Syntax: **set transform|\$tra to gaussian equalize**
[\$halfwidth]
\$halfwidth Number, representing envelope halfwidth in standard deviations

Example

```
set transform to gaussian equalize
```

set transform to histogram equalize

Applies a histogram equalize operation to the current or specified transform.

Syntax: **set transform|\$tra to histogram equalize**

Example

```
set transform to histogram equalize
```

set transform type

Sets the current or specified transform type.

Syntax: **set transform|\$tra type to \$ttype|linear|exp|log|hist**
\$ttype Transform type variable: value can be linear, exp, log or hist

Example

```
set transform type to linear
```

set view mode

Sets the view mode of the current or specified algorithm to 2D, 3D perspective or 3D flythru.

Syntax: **set [\$alg] view mode to \$vmode|2d|perspective|flythru**
\$vmode ViewMode variable: value can be 2d, perspective or flythru

Example

```
set $alg view mode to perspective
```

set window geolink mode

Various geolink functionality.

Syntax: **set window|\$win geolink mode to none|window|screen| overview**

Example

```
set window geolink mode to window
```

setenv (environment variable)

Set the given environment.

Note: This changes the current environment within ER Mapper.

Syntax: **setenv \$envname \$value**

\$envname String with name of environment variable

\$value String with value to set the environment variable to

Example

```
setenv "ERM_MACHINE_TYPE" "win32"
```

show image (in container)

Displays the specified image in the container.

Syntax: **show image \$filename**

\$filename String with path and name of image file. path name can be absolute or relative to the %ERMAPPER%\icons directory

Examples

```
$image = "standard_icons\image_main"
show image $image

show image "standard_icons\image_main"
```

split string

Split the given string into an array of substrings at the given token. The array is terminated by an empty string. Specifying empty double quotes as the split character splits a string into an array of single letter elements.

```
$string = "first second third"
$array = split $string at " "
$array[1] -> "first"
$array[2] -> "second"
$array[3] -> "third"
$array[4] -> "" - terminator
$array[5] -> invalid
```

Note: The first array index will always be a 1.

Syntax:	split \$string at \$token
\$string	String to be split
\$token	String with character where string must be split

Example

```
$filenames = split $string at "\n"
```

surface active

Checks whether the current surface is active, and returns TRUE (1) for active, and FALSE (0) for inactive.

Syntax:	surface active
----------------	-----------------------

Examples

```
$surface_is_on = surface active
```

system command

Executes a system command from within the script.

Syntax: **system \$command [status ["Title"]]**

\$command System command to be executed

Note: "&" at the end of the command will make it run in the background.

status Optional status dialog box

"Title" String with status box title (optional)

Examples

```
$cmd = "ermbalance calcclip" + $balance_file_spec + "&"
system $cmd

system "ermbalance calcclip" + $balance_file_spec + "&"
```

transform2 = transform1

Assign specified transform (e.g. \$tra1) to another variable (e.g. \$tra2).

Note: \$tra2 and \$tra1 refer to the same object, so deleting \$tra1 will invalidate \$tra2. Use the copy or duplicate transform command to create a separate object.

Syntax: **\$tra2 = \$tra1**

Example

```
$tra2 = $tra1
```

turn layer on|off

Turns the current or specified layer on or off.

Syntax: **turn layer|\$lay on|off**

Example

```
turn $layer1 on
```

turn surface on|off

Enables/Disables the current surface.

Syntax: **turn surface on|off**

Example

```
turn surface on
```

window2 = window1

Copies the window reference from win1 to win2, that is, both \$win1 and \$win2 point to the same object.

Syntax: **\$win2 = \$win1**

Wizard close

The "Wizard Close" command is needed because you often *can't* automatically close a wizard when the user presses the **Finish** button. You usually need to check the necessary parameters have been entered. In this case you would check everything is fine, and if it is, close the wizard using the "Wizard Close" command, and if it is not, go back to the appropriate page of the wizard. You can use Exit to exit a script and close down the wizard automatically but this is not recommended.

Syntax: **Wizard close [name]**

name If you don't specify a Name, all wizards are closed. If you specify a Name, then the named wizard, and all sibling wizards created after it, are closed.

WizardPage begin|end

This marks the beginning and end of a new wizard page. 'WizardName' is the name of the wizard within the script. If it is a different name from the previous Wizard Block a new wizard window is drawn; if it is the same as the name in the previous Wizard Block, the previous window is redrawn with the new contents. WizardName is optional for second and subsequent pages: if omitted it defaults to the name from the previous Wizard Block.

Syntax: **WizardPage begin "WizardName"**
 title "title_text"
 (*container blocks*)
 WizardPage end

title "title_text" Specifies the title which will appear on the Title Bar of the wizard page on screen. This should include the name of the Wizard (which indicates what it is used for), followed by a hyphen and the step number. For example,
 title "ClassificationWizard - Step 1 of 4"

container blocks If no container block is defined a default container block the size of the whole page will be included.

WizardPage end Defines the end of the wizard page.

Example:

```
WizardPage begin "WizardName"#title for first page
  title "Page1Name" #if a new wizard
  container begin "Image"
    container information          #usually image info
    ...                          #on first page
  container end
  container begin "DataEntry"
    container information
    ...
  container end
  container begin "PageControls"  #control buttons
    ask action "< Back"
    ask action "Next >" goto label1
    ask action "Cancel" close goto label2
  container end
  wizard close
WizardPage end
```

zoom

Zoom in or out using specified option.

Syntax: **previous zoom**
 zoom in|out
 zoom to all datasets|current datasets|all raster
 datasets|all vector datasets|contents extents|page
 extents|page contents

Examples

```
previous zoom
zoom in
set pointer mode to roam
zoom to all datasets
zoom to page contents
```

Index

Symbols

|ask colourchooser 251

A

add transform (type) 32

algorithm commands in scripts 23

B

batch scripting

actions 15

algorithm commands 23

batch script library 48

cell type 239

color keywords 45

controls 243

dialog boxes 4

ER Mapper objects 11

file name, extension and path keywords 45

filter commands 36

flow control 243

formula commands 34

image manipulation 12

input in scripts 3

layer commands 28

mathematical functions 236

operators 235

page setup commands 39

page size chooser 240

page view mode commands 41

preferences commands 42

print commands 46

status dialog 47

surface commands 26

transforms commands 32

using preferences to remember settings 10

variables 236

view mode commands 41

warning command 304

warning dialog 47

window commands 21

wizards 8

button action command in batch scripting 249

Index

C

Cell type in batch scripting 239

container blocks in wizards 335

D

deleting files and directories in scripts 44, 268

dialog boxes, creating with scripts 4

E

editing filenames in scripts 44, 271

environment variables

 getting a value in scripts 44, 291

 setting in scripts 44, 331

exit from batch scripts 272

exiting in scripts 243

F

file exists 44

File name, extension and path keywords 272

filters

 script commands 36

formula

 script commands 34

free space, finding out in scripts 44, 281

G

geolinking

 in scripts 22, 330

goto commands in scripts 243

I

if ... then ... else commands in scripts 243

image algebra. See formulas

including files in scripts 244

L

labels in scripts 243

layers script commands 28

list all image files 253

loops in scripts 244

M

machine type, determining in scripts 240

O

operators in scripting language 244

output in scripts 46

P

page setup

 script commands 39

page size chooser in batch scripting 240

page view mode 41

page view mode, determining in scripts 238

preferences

- script commands 42
- printing in scripts 46
- S
- scripting language
 - absolute|relative path 245
 - add (new) layer 28, 245
 - add (new) transform (to layer) 246
 - add filter 36
 - add filter (to layer) 246
 - add formula 34
 - add formula (to layer) 246
 - add layer 28
 - add layer (to surface) 247
 - add surface (to algorithm) 26, 247
 - add transform (to layer) 32, 247
 - add transform point 32, 247
 - add transform to layer input 32, 248
 - algorithm2 = algorithm1 23, 248
 - ask 5
 - ask action 7, 249
 - ask bandchooser 5, 250
 - ask bandmenu 6, 251
 - ask colorchooser 6, 251
 - ask datum 6, 251
 - ask directory 6, 252
 - ask file 6, 253
 - ask gridlayermenu 6, 254
 - ask hardcopy 6, 254
 - ask link 6, 255
 - ask listmenu 6, 256
 - ask listmenu_exclusive 6, 258
 - ask lutmenu 6, 258
 - ask projection 258
 - ask text|number 7, 259
 - ask yes/no 7, 259
 - build absolute filespec 43, 259
 - build file path 43, 260
 - build relative filespec 43, 260
 - color keywords 260
 - color_blue (keyword) 45, 260
 - color_green (keyword) 45, 260
 - color_red (keyword) 45, 260
 - concatenation 236
 - container above|below|left|right 9, 261
 - container begin|end 9, 262
 - container items 9, 262

Index

- container labels 9, 262
- container right|left justify 9
- container right|left|justify 261
- container width|height 9, 263
- convert file separator 263
- convert to english|native 43
- copy algorithm 23, 263
- copy algorithm from window 21, 264
- copy algorithm to window 21, 264
- copy filter 36, 264
- copy formula 34, 264
- copy layer 28, 265
- copy surface 26, 265
- copy transform 32, 265
- copy window 13, 21, 265
- current algorithm 12, 23, 266
- current filter 12, 36, 266
- current formula 12, 34, 266
- current input 12
- current layer 12, 28, 266
- current surface 12, 26, 267
- current transform 12, 32, 267
- current window 12, 21, 267
- delete algorithm 23, 267
- delete directory|file 44, 268
- delete filter 36, 268
- delete formula 34, 268
- delete layer 28, 268
- delete status (dialog) 48, 269
- delete surface 26, 269
- delete transform 32, 269
- delete window 21, 269
- dialog box
 - title 5
- dirname (keyword) 45, 272
- duplicate filter 36, 270
- duplicate formula 34, 270
- duplicate layer 28, 270
- duplicate surface 26, 270
- duplicate transform 32, 271
- edit file 271
- edit text file 44
- error codes 244
- execute system command 45
- exists 271
- exit 243, 272

exit batch process 44
 file exists 44
 fileext (keyword) 45, 272
 filename (keyword) 45, 272
 filter2 = filter1 273
 filterl2 = filter1 36
 first|last|nex|previous surface 275
 first|last|next|previous algorithm 23, 273
 first|last|next|previous filter 36, 273
 first|last|next|previous formula 34, 274
 first|last|next|previous layer 29, 274
 first|last|next|previous surface 26
 first|last|next|previous transform 33, 274
 first|last|next|previous window 22, 275
 fit page to hardcopy 39, 275
 format number of digits after decimal point 45
 format value precision 276
 get (algorithm) page view mode 41, 288
 get (algorithm) view mode 41, 290
 get (ER Mapper) version 291
 get (page) contents topleft|bottomright 39, 276
 get algorithm coordsys 23, 276
 get algorithm datum|projection 23, 277
 get algorithm description 23, 277
 get algorithm layer count 23, 277
 get algorithm lut 23
 get algorithm lut (color table) 277
 get algorithm mode 23, 278
 get algorithm mosaic type 24, 278
 get algorithm topleft|bottomright (extents) 24, 278
 get algorithm units|rotation 24, 278
 get english file separator 43, 279
 get environment variable 44
 get ER Mapper version number 45
 get file size 44, 279
 get filter description 36, 279
 get filter matrix 36
 get filter params 37, 279
 get filter postsampled (flag) 37, 280
 get filter rows|cols 37, 280
 get filter scale|threshold (value) 37, 280
 get filter type 37, 281
 get filter userfile (filename) 37, 280
 get filter userfunc 37, 281
 get free space (on disk) 44, 281
 get layer azimuth|elevation 29, 281

Index

- get layer band count 29, 282
- get layer band description 29, 282
- get layer cell size 29, 282
- get layer cell type 29, 282
- get layer color 29, 283
- get layer coordinates 29, 283
- get layer dataset (filename) 29, 283
- get layer description 29, 284
- get layer edit/init program (name) 29, 284
- get layer editable (flag) 29, 284
- get layer formula 29, 284
- get layer input count 29
- get layer input filter count 29, 285
- get layer input transform count 29
- get layer link extension 30, 285
- get layer output filter count 30
- get layer output transform count 30, 285
- get layer shading 30, 285
- get layer type 30, 286
- get layer|\$lay input count 286
- get native path|file separator 43, 286
- get page constraints 39, 286
- get page scale 39, 287
- get page size 39, 287
- get page top|bottom|left|right border 39, 287
- get page topleft|bottomright (extents) 39, 287
- get page width|height 39, 288
- get preference 42, 288
- get preference (color) 42, 289
- get surface description 26, 289
- get surface layer count 26, 290
- get transform (limits) 290
- get transform input/output min|max 33
- get transform type 33
- getenv (environment variable) 291
- go algorithm 24, 291
- go background window 22, 292
- go window 22, 292
- goto 243
- if ... then ... else 243
- include "filename" 244
- keywords 241
- labels 243
- layer active 30, 292
- layer2 = layer1 28, 292
- list contents of directory 44

- listdir 44, 293
- load algorithm 24, 293
- load filter (filename) 37, 294
- load formula (filename) 34, 294
- load layer formula (filename) 30, 294
- looping 244
- mathematical functions 236
 - abs(x) 236
 - asin(x), acos(x), atan(x) 236
 - ceil(x) 236
 - floor(x) 236
 - log(x), exp(x) 236
 - min(x,y), max(x,y) 236
 - pow(x,y) 236
 - sin(x), cos(x), tan(x) 236
 - sqrt(x) 236
- move layer up|down|top|bottom 30, 295
- move surface up|down|top|bottom 27, 295
- new algorithm 24, 296
- new filter 37, 296
- new formula 34, 296
- new layer 30, 297
- new surface 27, 297
- new transform 33, 297
- new window 22, 298
- next layer 30
- next layer (type) 298
- open status (dialog) 48, 299
- open window 22, 299
- operators 235
 - != > >= 235
 - # 235
 - () 235
 - = * / + - % 235
 - ~ 235
- previous zoom 22, 336
- print 47, 300
- println 47, 300
- save algorithm 24, 301
- save algorithm as dataset 24, 302
- save algorithm as virtual dataset 24, 303
- save filter (filename) 37, 303
- save formula (filename) 34, 303
- save layer formula (filename) 30, 304
- say status 48, 304
- say warning 47, 48, 304

Index

- select algorithm 24, 305
- select filter 37, 305
- select formula 34, 305
- select layer 30
- select surface 27, 306
- select transform 33, 306
- select window 22, 306
- set (formula) input to band 306
- set (page) contents bottomright from topleft 39, 307
- set (page) contents extents to algorithm extents 40, 307
- set (page) contents topleft|bottomright 39, 307
- set algorithm background (color) 24, 309
- set algorithm coordsys 24, 309
- set algorithm datum|projection 24, 310
- set algorithm description 24, 310
- set algorithm lut (color table) 24, 311
- set algorithm mode 25, 311
- set algorithm mosaic type 25, 312
- set algorithm supersample type 25, 312
- set algorithm topleft|bottomright (extents) 25, 313
- set algorithm units|rotation 25, 313
- set environment variable 44
- set filter description 37, 314
- set filter matrix (element) 37, 314
- set filter params 37, 314
- set filter postsampled (flag) 37, 315
- set filter rows|cols 38, 315
- set filter scale|threshold 38, 315
- set filter type 38, 316
- set filter userfile (filename) 38, 316
- set filter userfunc (funcname) 38, 316
- set formula 35, 317
- set input to band 35
- set layer azimuth|elevation 30, 317
- set layer color 30, 318
- set layer dataset (filename) 30, 318
- set layer description 30, 319
- set layer edit|init program (name) 31, 319
- set layer editable (flag) 31, 319
- set layer formula 31, 320
- set layer input to band 31, 320
- set layer link extension 31, 320
- set layer link type to monocolour|truecolour 31, 321
- set layer shading on|off 31, 321
- set layer type 31, 321
- set output to filename|window 47, 300

- set page autovary_value 40, 308
- set page center horizontal|vertical 40, 321
- set page constraints 40, 322
- set page extents from contents 40, 322
- set page scale 40, 322
- set page size 40, 323
- set page top|bottom|left|right border 40, 324
- set page topleft|bottomright (extents) 40, 323
- set page view mode 41, 324
- set page width|height 40, 324
- set pointer mode 22, 325
- set preference 42, 325
- set preference (color) 42, 326
- set surface description 27, 326
- set surface lut (color table) 27, 327
- set surface mode 27
- set surface mode (color) 327
- set surface name 27, 327
- set surface zscale|zoffset|transparency 27, 328
- set transform clip 33, 328
- set transform input|output min|max (limits) 33, 328
- set transform limits (actual or percentage) 33, 329
- set transform output limits to input limits 33
- set transform to gaussian equalize 33, 329
- set transform to histogram equalize 33, 329
- set transform type 33, 330
- set view mode 41, 330
- set window geolink mode 22, 330
- setenv (environment variable) 331
- show image 9
- show image (in container) 331
- split string into array 44, 332
- surface active 27, 332
- system command 333
- system command status dialog 45
- transform2 = transform1 32, 333
- turn layer on|off 31, 333
- turn surface on|off 27, 334
- variables 236
 - arrays 240
 - color 239
 - coordsys 238
 - filtertype 238
 - layertype 237
 - machinetype 240
 - mode 237

Index

- mosaictype 238
- number 237
- page size options 240
- pageviewmode 238
- references 239
- string 237
- supersampletype 238
- transformtype 238
- viewmode 238
- yesno 237
- window2 = window1 21, 334
- wizard close 8, 334
- wizardpage begin|end 8, 335
- zoom in|out 22, 336
- zoom to 22, 336
- set 312, 313
- splitting strings in scripts 44, 332
- status dialog 47
- strings, splitting in scripts 44, 332
- surface script commands 26
- system command, executing in scripts 45
- T
- scripting language
 - match transform 33, 295
- transforms
 - script commands 32
- V
- variables in scripting language 236
- verify existence of specified file 44, 271
- view mode
 - script commands 41
- view mode, determining in scripts 238
- W
- warning command in batch scripting 304
- warning dialog 47
- wizards
 - container block 335
- Z
- zoom 336
- zooming in scripts 336