MAPCOMP Map Specification Language (R20)

Top

Introduction

INTREPID uses the MAPCOMP language to produce image output to printers, plotters or files.

A MAPCOMP file is an ASCII file, normally with the extension **`.map`**.

INTREPID automatically generates and prints the map specified in MAPCOMP files. With a few minor adjustments, MAPCOMP files are compatible between *Windows* and Unix.

You can use any text editor to modify or create your own MAPCOMP file.

This chapter contains specifications for the MAPCOMP language and some examples of MAPCOMP files.

Map Composition syntax

A MAPCOMP file has a nested **`Begin – End`** block structure describing objects on the composition. It contains four classes of object:

- Group objects contain other objects and affect their position or appearance.
- Geographically located objects describe how you require data from a dataset to be set out
- Annotations for sets of geographically located objects, e.g., tick marks or grids.
- Annotations such as North arrows, titles, sheet indexes, text annotations. Some of these depend on geographic data and others are simple annotations.

Properties statements and blocks

Object **`Begin – End`** blocks contain a number of statement lines describing the properties of the object.

Sometimes these are single lines specifying a single property.

Example:

```
Width = 210
```

Sometimes a property has its own **`Begin – End`** block

Example

```
MapProjection Begin
     Projection  = TMAM650
     Datum       = AGD66
     Xscale      = 50000
     Yscale      = 50000
MapProjection End
```

See Details of MAPCOMP language structure for a list of the property blocks that belong to each object type.

MAPCOMP objects

Group objects

Some objects are intended to act as containers for other objects. These group objects affect the objects within in various ways, such as

- Providing a border or margin around the group,
- Controlling the arrangement of objects within the group,
- Geographically aligning the data belonging to each object in the group.
- Automatically scaling (enlarging or reducing) objects in the group.

Here is a list of the group objects.

Object	Description
Non-Aligning Groups	
Page	A **Page** can contain objects that you wish to treat as a group. It is identical to a **Box** except for its name. We have provided it so that you can knowingly define a clearly shown page size, outer border and margin. Normally you will define a page box according to the paper you are using.
Box	A **Box** can contain objects that you wish to treat as a group. The objects within a box normally define its size.
Border/Margin Groups	
Margin	A specification for a margin around one or more objects. If the margin contains a **Page** group, the margin can be interior, which effectively reduces the size of the group rather than increasing it.
Border	A specification for a border around one or more objects.
Centre Group	
Centre	A box which centre aligns in both directions the objects it contains. If you have more than one object, they will be superimposed.
Horizontally Aligning Groups	
HBox	A box that aligns the objects it contains so that they are adjacent to each other in a horizontal row.
Top	A box which aligns the objects it contains along its top edge but does not influence their horizontal position.
VCentre	A box which aligns the objects it contains along its horizontal centre line but does not influence their horizontal position on that line.
Bottom	A box which aligns the objects it contains along its bottom edge but does not influence their horizontal position.
Vertically Aligning Groups	
VBox	A box that aligns the objects it contains so that they are adjacent to each other in a vertical row.

Object	Description
`Left`	A box which aligns the objects it contains along its left edge but does not influence their vertical position.
`HCentre`	A box which aligns the objects it contains along its vertical centre line but does not influence their vertical position on that line.
`Right`	A box which aligns the objects it contains along its right edge but does not influence their vertical position.
Special Purpose Groups	
`Flexible`	A box which scales the object it contains to fit its size. These boxes are intended for scalable graphics such as DGN or TIFF.
`Include`	A box which specifies another MAPCOMP file for insertion at this point in the composition.
`Data`	A box which contains geographically located data (i.e., datasets).

Nested groups in MAPCOMP language

Map Composition represents the nested group structure in MAPCOMP files using nested **`Begin – End`** blocks. It represents a group with a **`Begin – End`** block. Each object within a group is within the group's **`Begin – End`** block.

Each object in turn has its own **`Begin – End`** block, and so to contain another object, has the entire object's **`Begin – End`** block within its own **`Begin – End`** block.

If you are directly editing a MAPCOMP file, you may only nest other objects within group objects (e.g., **`Box, HCentre, Border`**). If you attempt to nest another object in a non-group object (e.g., **`Text, ScaleBar`**), Map Composition displays an error message when it attempts to process the MAPCOMP file.

Examples of group objects

A4 page with margin and border

Note that the margin of 5 mm has decreased the A4 page size from 210 mm x 297 mm.

```
Margin Begin
   X =    0
   Y =    0
   Top = 5
   Bottom = 5
   Left = 5
   Right = 5
   Border Begin
      X =    0
      Y =    0
      Thickness = 1
      Colour = Black
      Style = Solid
      Page Begin
         X =    5
         Y =    5
         Width = 200
         Height = 287
      Page End
   Border End
Margin End
```

Two line centred title

```
VBOX Begin
   X = 55
   Y = 240
   Text Begin
      String = "Ebagoola"
      Colour = Black
      Size = 6
      Font = 5
      Justify = cb
   Text End
   VSpace Begin
      Space = 5
   VSpace End
   Text Begin
      String = "October 1996"
      Colour = Black
      Size = 4
      Font = 5
      Justify = cb
   Text End
VBOX End
```

Geographically located data objects

These objects represent data from INTREPID datasets and make up the data that you are illustrating with the composition. You must place them inside a **`Data`** group object. Map Composition will align them geographically for you.

Object	Description
`PseudoColour`	Pseudocolour display from a grid dataset.
`GreyScale`	Grey scale display from a grid dataset
`FixedColour`	Fixed colour display from a grid dataset
`SunAngle`	Sun angle display from a grid dataset
`FalseColour`	False colour display from a 3 band grid dataset
`Drape`	Pseudocolour display from a grid with intensity drape
`TernaryDrape`	A false colour display with intensity drape from one or more grid datasets
`Contour`	A set of contours representing a grid dataset
`ColourContour`	A set of contours representing a grid dataset. These contours vary in colour according to the values of a field in an grid dataset.
`PathPlot`	A traverse line dataset
`StackProfilePlot`	A traverse line dataset with stack profile
`PointPlot`	A point dataset
`PolygonPlot`	A polygon dataset

Annotations that attach to a Data object

These annotations have their own **`Begin – End`** blocks, give geographic information about a **`Data`** object, and are defined within it.

Object	Description
`Ticks`	Tick, grid or border marks
`Corner`	Corner annotations showing extents

Annotation objects that depend on a Data object

These annotation objects depend geographically on a **`Data`** object for their appearance, but are not defined within it. If there are two or more data objects, these objects refer to the first one that you define.

Object	Description
`ScaleBar`	The scale bar shows the geographic scale of the **`Data`** object.
`NorthAarrow`	A North arrow for the **`Data`** object
`SheetIndex`	This shows all of the sheets defined by this MAPCOMP file, indicating the sheet shown on the current page.

Annotation objects that describe grid or Z data

Object	Description
Legend	This annotation describes a band of a grid dataset or Z field of a vector dataset. It gives a key to value ranges for different colours, shapes, sizes or thicknesses.
MultiPlot	This annotations are profile graphs, typically of a Z field with fiducial or a geographic location field on the horizontal axis.

The **MultiPlot** object replaces the proposed **XYPlot** object in MAPCOMP[1]. A **MultiPlot** object can contain a number of **Panel** objects, each with its own XY graph.

The **MultiPlot** object has special provision for plotting multichannel radiometrics data.

Annotation objects independent of datasets

Object	Description
Greybar	Bar showing grey scale range
Marker	Point to be marked in the composition using a marker
Line	Line to be marked in the composition
Polygon	Polygon to be marked in the composition
Image	Logo or trademark image
DGNInclude	A MicroStation DGN file

[1] It is still called XY Plot in the interactive Map Composition tool.

Determining the positions of objects

Each object can have an **X=** and **Y=** specification which defines the position of its lower left corner. This position is always relative to the lower left corner of the object that contains it (i.e., in the lower left corner **X = 0** and **Y = 0**).

Aligning groups (**Centre, HBox, Top, VCentre, Bottom, VBOX, Left, HCentre, Right** objects) determine or override many of the **X=** and **Y=** specifications for the objects they contain.

If there is no **X=** and **Y=** specification then the object will be located in the lower left corner of the object that contains it.

Non-aligning groups (**Page** and **Box** objects) do not influence the relative position of the objects they contain. Their contained objects therefore normally require **X=** and **Y=** specifications.

Centre boxes, **VBOX**es and **HBox**es align their contained objects in both dimensions and will ignore **X=** and **Y=** specifications in their contained objects.

Left, **HCentre** and **Right** boxes align their contained objects horizontally but not vertically. The objects require **Y=** specifications for their contained objects but not **X=** specifications, which the boxes will override if it exists.

Top, **VCentre** and **Bottom** boxes require **X=** specifications for their contained objects but not **Y=** specifications, which the boxes will override if it exists.

HBoxes, VBoxes, HSpace and VSpace

VBOXes and **HBox**es align their objects in a vertical column or horizontal row respectively. The order of the objects starts from the bottom or left respectively. The objects are positioned in the order in which they occur in the definition of the **VBOX** or **HBox**. The contained objects do not overlap each other[19]. The sizes of the contained objects determine the size of a **VBOX** or **HBox**, so **VBOX**es and **HBox**es do not have **Width=** or **Height=** specifications.

HSpace and VSpace You can separate objects within a **VBOX** or **HBox** using inserted horizontal and vertical space. Between the included objects in the box, use **HSpace** and **VSpace** blocks, specifying the number of millimetres of separation between the objects. Here is an example of a **HSpace** block.

```
HSpace Begin
Space = 5
HSpace End
```

Alternatively, you can create space between object by placing a margin around each one.

See Examples of group objects for a full example of a **VBOX**.

1.[9] Unless you have given them an internal margin. We suggest you avoid this practice unless you fully understand what you are doing.

Object positioning summary

In the following table 'automatic' indicates that an aligning groups object determines the **X=** or the **Y=**. The notation 'required' indicates that you need to specify an **X=** or **Y=** specification. The table also contains the default positions within the box if you omit the 'required' entry.

Object	X=	Y=
Within non-aligning groups		
`Page/Box`	required (default is left edge)	required (default is bottom)
`Centre`	automatic	automatic
Within horizontally aligning groups		
`HBox`	automatic	automatic
`Top`	required (default is left edge)	automatic
`VCentre`	required (default is left edge)	automatic
`Bottom`	required (default is left edge)	automatic
Within vertically aligning groups		
`VBOX`	automatic	automatic
`Left`	automatic	required (default is bottom)
`HCentre`	automatic	required (default is bottom)
`Right`	automatic	required (default is bottom)

Sizes of objects

Each object can have a **Width=** and **Height=** specification which defines its size in millimetres.

In some cases you can or should omit these specifications.

Since the purpose of a **Page** object is to define the page on which the Map Composition will print the composition, it clearly must have size specifications

The purpose of a **Box** object is normally to group a set of objects. You can specify size for a **Box**, but this is not normal practice. If there is no size for a **Box**, the sizes and positions of the contained objects determine its size.

The sizes of the contained objects determine the size of a **VBOX** or **HBox**, so **VBOX**es and **HBox**es do not need **Width=** or **Height=** specifications.

Left, HCentre, Right, Top, VCentre, Bottom align their objects in one dimension only. The length of the row or column of objects determines one dimension. The largest object normally determines the other dimension.

Since the purpose of a **Flexible** box is to determine the size of a scalable graphic, it clearly must have size specifications.

Data boxes should have size specifications. The **Data** box determines the size of its geographically located objects, so they should not have size specifications.

All annotation objects can have size specifications. In some cases (e.g., **Line, Polygon, Text** objects), Map Composition will calculate a default size based on the nature of the object.

The following table summarises how Map Composition determines the width and height of all objects.

Object	Width	Height
Non-Aligning Groups		
Page	**Width=** required	**Height=** required
Box	**Width=** optional (Default is determined by size and **X=** of contained objects)	**Height=** optional (Default is determined by size and **Y=** of contained objects)
Centre	**Width=** optional (Default is width of largest contained object)	**Height=** optional (Default is height of largest contained object)

Object	Width	Height
Horizontally Aligning Groups		
HBox	Automatic Determined by size of contained objects	
Top	Automatic Determined by size and **X=** of contained objects	**Height=** optional (default is height of largest contained object)
VCentre		
Bottom		
Vertically Aligning Groups		
VBOX	Automatic Determined by size of contained objects	
Left	**Width=** optional (default is width of largest contained object)	Automatic Determined by size and **Y=** of contained objects
HCentre		
Right		
Special Purpose Groups		
Flexible	**Width=** required	**Height=** required
Data	**Width=** required	**Height=** required
Border/Margin Groups		
Margin	**Width=** optional (default is sufficient to enclose all contained objects)	**Height=** optional (default is sufficient to enclose all contained objects)
Border		
Geographically Located Objects		
Automatic. Determined by size of containing **Data** object		
Annotation objects not part of a Data object		
Required. Map Composition calculates a default size		

MAPCOMP language features

Comments

If a line begins with '#', INTREPID ignores the contents of the line

(i.e., lines beginning with '#' are comment lines). Map Composition places some comment lines into your MAPCOMP files automatically.

e.g.,

```
#### Audit Stamp Intrepid V3.1 - 12/ 7/1995 12:23:48
```

Arrays

Some MAPCOMP statements consist of lists of items. These are called **arrays**. These may not fit conveniently on one line of a MAPCOMP file. Arrays are often more readable if split onto several lines. For example, you may have a long sequence of `X Y` coordinate pairs describing a line object. You can divide the statement onto several lines of text and place the whole statement within { and } characters. Map Composition will then interpret the contents of the { } as a single statement.

In this example the numbers are X, Y coordinate pairs. They are more readable if you place each pair on a separate line.

```
XY = {
      0 0
      10 10
      20 15
      30 17.5
}
```

Data types in the MAPCOMP syntax

The description of the MAPCOMP language appearing in the following section describes the assigning of values to keywords (See Details of MAPCOMP language structure). These values can be of the following data types.

Data type	Description
`<number>`	(number)
`<a..b>`	(number in the range **a..b**)
`<string>`	(characters enclosed in `"  "`)
`<char>`	(single character)
`<deg:min:sec>`	(latitude/longitude notation)
`<filename>`	(directory and extension are assumed)
`<path>`	(filename with directory specification)
`<YES|NO>`	
`<...|...|...>`	(listed options)
`<symbol>`	(marker symbol shape)*
`<ticktype>`	(tick marks type)*
`<filltype>`	(polygon fill type)*

Data type	Description
`<thickness>`	(line thickness)*
`<style>`	(line style)*
`<typeface>`	(typeface number)*
`<justification>`	(text justification type)*
`<colour>`	(colour)*

Examples

```
X             = 450(<number>)
Saturation    = 0.7(<0..1>)
String        = "Airborne Survey"(<string>)
HighSymbol    = H(<char>)
Corner        = 143:46:12(<deg:min:sec>)
Projection    = TMAMG50(<filename>)
Dataset       = d:/surveys/ebagoola(<path>)
Internal      = no(<YES|NO>)
Unit          = METRES(<METRES|DEGREES>)
Symbol        = cross(<symbol>)
TickMarks     = border(<ticktype>)
PolyFill      = hollow(<filltype>)
Thickness     = .75(<thickness>)
BorderStyle   = solid(<style>)
LabelFont     = 5(<typeface>)
Justify       = tc(<justification>)
SymbolColour  = blue (<colour>)
```

For possible values of the types marked * see Available map attribute values (R22).

Text features

Font size You can specify font size in millimetres. 1 millimetre equals approximately 3 points (more precisely 2.83 points).

Font weight You can specify the weight (line thickness) of text using the `LineThickness =` statement. The text weight is measured in mm. A value of 0 (the default) will result in the finest lines possible for your printer. The following statement will result in text of line thickness 0.5 mm

```
LineThickness = 0.5
```

Multiple line text You can specify a text object with several lines of text by placing a number of strings in the statement. INTREPID will place each string on a new line. Example:

```
String = {
      "Desmond Fitzgerald & Assoc."
      "Brighton, Australia"
      }
```

If you use multiple line text you **must** specify the line spacing in mm using the `VGap =` *`statement`* (otherwise the lines will be too close together).

Example: `VGap = 5`

Embedded codes in strings You can align text in a string with positions in the text object using embedded tab codes (e.g.,) `[htab25]`, `[vtab10]`. The number in the code specifies the distance in mm from the left or top edge of the text object. Example:

```
String = {
      "Anomaly [htab30] Depth"
      "A342 [htab30] 2540"
      "A54 [htab30] 560"
      "F25A [htab30] 740"
      }
VGap   = 5
```

The above example will produce a two column table with the second column 30 mm from the left edge of the text object.

Embedded parameters in strings You can include the names of scalar declared, array declared and command line replaceable parameters in a string. See Declared parameters within a MAPCOMP file and Command line replaceable parameters for information abut these features. Example:

```
DFA = "Desmond Fitzgerald and Associates"
Page Begin
      ...
      String = "Prepared by $DFA"
      ...
```

The TrueTypeFont statement

To render a text object in a TrueType font, include a **TrueTypeFont** statement, assigning the attribute to the font name or the name of the font file as shown in the examples below.

```
Text Begin
      X = 10.0
      Y = 160.0
      String = "Some text"
      Colour = Black
      Size = 8.000000
      TrueTypeFont = "Courier New"  # using the font name
      Angle = 0.0000000000
      Justify = lb
      Gap = 0.0000000000
      VGap = 0.0000000000
      TextThickness = 0.0000000000
Text End
 Text Begin
      X = 10.0
      Y = 130.0
      String = "Some more text"
      Colour = Black
      Size = 8.000000
      TrueTypeFont = "cour.ttf"  # using the font file name
      Angle = 0.0000000000
      Justify = lb
      Gap = 0.0000000000
      VGap = 0.0000000000
      TextThickness = 0.0000000000
 Text End
```

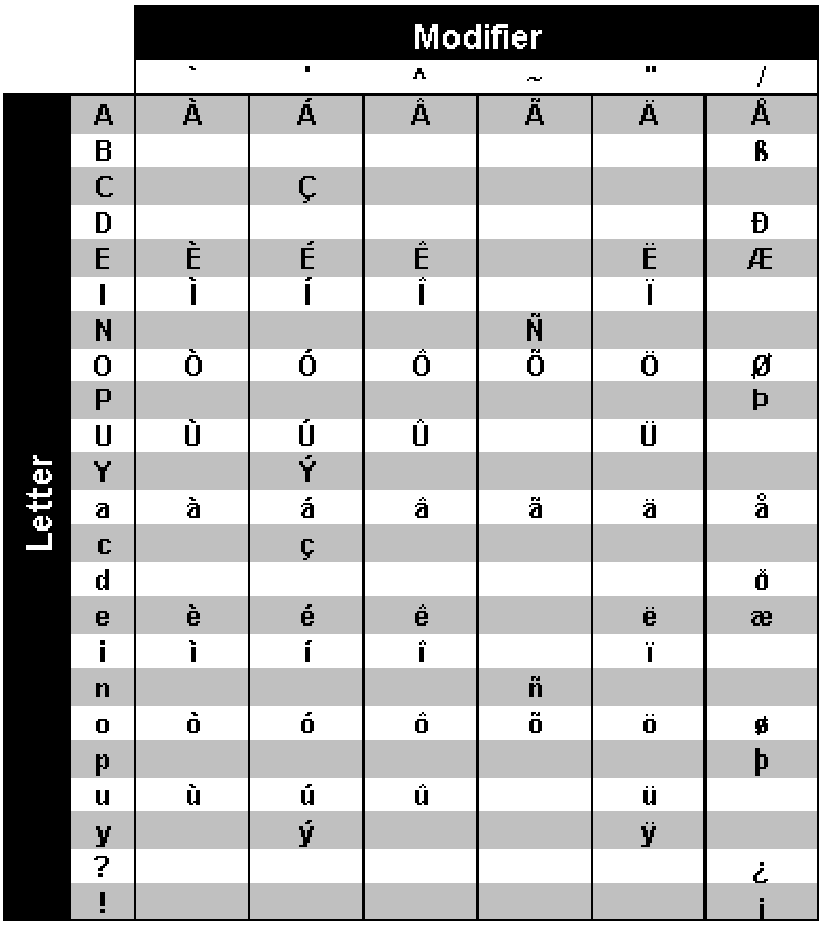
Font names should be enclosed in double quotes.

Extended Latin characters

You can embed extended Latin characters in TrueType text objects using special sequences consisting of a modifier character followed by the letter to modify.

The table below shows the available characters and the sequences that yield them. For example **~o** will yield **õ**. To disable this translation, precede the sequence by a backslash (\). For example, **\~o** will yield **~o**.

Note: not all TrueType fonts support the extended Latin character set.



		Modifier					
		`	'	^	~	"	/
Letter	A	À	Á	Â	Ã	Ä	Å
	B						ß
	C		Ç				
	D						Ð
	E	È	É	Ê		Ë	Æ
	I	Ì	Í	Î		Ï	
	N				Ñ		
	O	Ò	Ó	Ô	Õ	Ö	Ø
	P						Þ
	U	Ù	Ú	Û		Ü	
	Y		Ý				
	a	à	á	â	ã	ä	å
	c		ç				
	d						ð
	e	è	é	ê		ë	æ
	i	ì	í	î		ï	
	n				ñ		
	o	ò	ó	ô	õ	ö	ø
	p						þ
	u	ù	ú	û		ü	
	y		ý			ÿ	
	?						¿
	!						¡

Unicode characters

You can embed Unicode characters in TrueType text objects using a sequence consisting of **\u** followed by a 4-digit hexadecimal Unicode. For example, **\u00a9** displays the © symbol.

Symbol size

You can specify symbol sizes in millimetres.

Case sensitivity

MAPCOMP is case insensitive (except within " "). MAPCOMP files can have any mixture of upper or lower case.

Declared parameters within a MAPCOMP file

You can declare parameters and assign values to them within a MAPCOMP file. Normally these parameters would appear at the beginning of the file. You can then use the parameters to supply values in the MAPCOMP file. You need to place a **$** before the name of a parameter when you use it to supply a value.

Declared parameters can be scalar or in the form of an array.

Scalar declared parameters

A scalar parameter declaration is a simple *keyword = value* statement.

To use a scalar declared parameter, use its name preceded by a **$** symbol.

Example showing the use of a scalar declared parameter:

```
LTitle="Flight Path"
Page Begin
   ...
       Text Begin
          String = $LTitle
          Colour = Black
          ...
       Text End
   ...
Page End
```

Array declared parameters

An array parameter specification contains a list of values separated by spaces (See Arrays for format details).

To use one of the values of an array declared parameter, use its name followed by **!!** followed by a number indicating the value to be used (e.g., **4** for the fourth value.

In this **example** showing the use of an array declared parameter, the sheet name selected is the third one, **Ebagoola C**.

```
Sheetname = {"Ebagoola A" "Ebagoola B" "Ebagoola C"}
   ...
      Sheet = $Sheetname!!3
   ...
```

Internal MAPCOMP Macros

You can define MAPCOMP text as a macro (normally at the start of a MAPCOMP file), then refer to this text with a **Call=** statement whenever you wish to use the text in the MAPCOMP file.

For example, the macro shown at the right draws a rising curve. Place **Call=** statement for the macro the places in the file where you require a curve.

Internal macro example

```
Macro Begin
     Name = upcurve
     Line Begin
          XY = {
               0 0
               10 10
               20 15
               30 17.5
          }
     Line End
Macro End
Page Begin
      ...
      Call = upcurve
      ...
Page End
```

External macros

You can specify a call to an external program which will generate MAPCOMP statements at the position occupied by the external program call. The external program must be compiled and executable. We regard it as an **external macro** because it operates as a macro, providing MAPCOMP statements.

The format of an external macro call is as follows

```
macroname Begin
      argument1 = value1
      argument2 = value2
      ...
macroname End
```

Where

macroname is the name of the executable program, which must reside in the directory ***install_path*/bin/csh**.

argument1, argument2, ... are the names of input arguments for the program.

value1, value2, ... are values assigned to the input arguments for use in this execution of the program.

External macro example

```
Border Begin
      Data Begin
            MapExtent Begin
                  Xmin = 50000
                  Xmax = 150000
                  Ymin = 150000
                  Ymax = 250000
            MapExtent End
            MapProjection Begin
                  XScale = 1000000
                  YScale = 1000000
            MapProjection End
            emmacro begin
                  File=emfile
                  Scale = 1000000
                  Size = 5
                  extSize = 1
                  DipSize = 5
            emmacro end
      Data End
Border End
```

This MAPCOMP file (listed at the right) contains a call to an external macro called `emmacro` (an executable program residing in *`install_path`*`/bin/csh`). It generates MAPCOMP statements that will place a set of electromagnetic anomaly symbols into the composition. The input arguments for the program are `File`, `Scale`, ..., `DipSize`.

In our example the argument `File` specifies a text file `emfile` (resident in the working directory), which contains the data for the anomalies to be plotted. The other arguments specify parameters for the composition process.

The file `emfile` contains data like this:

```
10010 25134 2.23 45 0 100000 180000 10 type 12 200 300 10 A 5
10012 25144 2.26 20 0 110000 230000 10 type 8 200 300 110 D 5
```

It describes the anomaly data to be plotted. This file could have been automatically output from a dataset. You could also access an INTREPID dataset directly from the program.

The program `emmacro` is written in C.

It inputs the values of the parameters:

```
file = getenv("File");
chscale = getenv("Scale");
chsize = getenv("Size");
chtextsize = getenv("TextSize");
chdipsize = getenv("DipSize");
```

It opens the file and obtains the anomaly data:

```
fd = fopen(file,"r");
...
fscanf(fd,"%d %d %lf ... %lf %d %s %lf ...",
      &line, &fid, &lag, &head, &nomhead, &x, &y, ...)
```

It then outputs the MAPCOMP statements:

```
...
fprintf(out,"Polygon End\n");
fprintf(out,"Text Begin\n");
fprintf(out, "\tXY={%lf%lf}\n", x+X(-xaoffset,yaoffset),
y+Y(-xaoffset,yaoffset) );
fprintf(out,"\tString = %d\n",(int)ctp);
fprintf(out,"\tSize = %d\n",(int)textsize);
fprintf(out,"\tJustify = 14\n");
fprintf(out,"\tAngle = %lf\n",head);
fprintf(out,"Text End\n");
...
```

You can compile and install the **emmacro.c** file with the following commands.

```
cc emmacro.c -o emmacro -lm
cp emmacro $(INTREPID)/bin/csh
```

——

Contact our technical support service for a full listing of **emmacro.c** if required.

Command line replaceable parameters

INTREPID will substitute values of replaceable parameters from the **mapprint.exe** command line. Use the notation **$3, $4, $5, ...** to refer to the parameters in the MAPCOMP file.

Example

Under UNIX the first two command line parameters are 'reserved':

$1 is the MAPCOMP file name and **$2** is the output file name.

If the MAPCOMP file contains the statement

```
Z=$3
```

and the Print Map command line is

```
mapprint.exe morgan1.map morgan.prn thorium..LINE
```

then Print Map will use **thorium..line** as the value for **Z**.

Under Windows if you are using commands and outputting to a file, you need to use the **-file** switch after the command. This means that there are three 'reserved' parameters after the command and the first available for use as a command line parameter is **$4**.

In the case described above the *Windows* command line would be

```
mapprint.exe -file morgan1.map morgan.prn thorium..LINE
```

and the MAPCOMP file would contain the statement

```
Z=$4
```

Environment variable references

INTREPID can insert values of environment variables in MAPCOMP statements. Use the notation **$environment_var** to specify an environment variable value (where *environment_var* is the name of an environment variable).

Example: If you put

```
Dir=$INTREPID
```

Map Composition will look up the value of the environment variable INTREPID and use it for value of **Dir**.

References to parameter values in auxiliary files

INTREPID will look up values assigned to keywords in auxiliary files for use in MAPCOMP. Use the following notation in your MAPCOMP file:

$[auxfile]*blockname.keyword* or

$[auxfile]*blockname.sub_blockname.keyword*

Where

auxfile is the INTREPID auxiliary file;

blockname is the name of a **begin – end** block;

sub_blockname is the name of a **begin – end** block within the *blockname* block;

keyword is the name of a keyword within the **Begin – End** block.

Example

The file **ebagoola352.ers** includes the following lines

(... means lines omitted from this example):

```
DatasetHeader Begin
      Version    = "3.0"
      LastUpdated= Tue Jan 23 02:16:39 GMT 1994
      ...
      CoordinateSpace Begin
            Datum = "RAW"
            Projection= "RAW"
            CoordinateType= RAW
            Rotation= 0:0:0.0
      CoordinateSpace End
      ...
DatasetHeader End
```

If you put

```
Projection=$[ebagoola352.ers]DatasetHeader.CoordinateSpace.Projection
```

Map Composition will look up the auxiliary file and insert the value **RAW** for the attribute **Projection**.

Compatibility of MAPCOMP between *Windows* and Unix

MAPCOMP is almost totally compatible between platforms. The following notes describe the only compatibility considerations.

- If you are creating output files by command you will need to specify a device configuration macro appropriate to your system.
- INTREPID automatically adjusts the `\` and `/` separators in paths according to the platform you are using.
- Full paths of datasets or files in your MAPCOMP file will not normally be compatible because of *Windows* drive letters and Unix drive names. You can make full paths compatible using an environment variable. For example if you use an environment variable called DIR to contain the drive letter or drive name, you can indicate the drive letter or name in the MAPCOMP file with the notation **`$DIR`**.

For example a MAPCOMP file may specify a path **`$DIR/surv/t567`**. When you are using the file on a *Windows* computer, you will set DIR to the drive letter where the data resides (say, **`f:`**) from the *Windows* computer's viewpoint, whereas when you are using a Unix computer to process the same MAPCOMP file, you will set DIR to the drive name corresponding to the data location from the Unix computer's viewpoint (say, **`/jupiter`**)

For further discussion, see:

- "Using Print Map with commands" in Map printing (T46) and "Printing with commands under Unix" in Map printing (T46) for further information about this topic.
- "Accessing INTREPID data from both UNIX and Windows" in Configuring and using INTREPID (R04)
- INTREPID system parameters and `install.cfg` (R07)
- Environment variable references.

Details of MAPCOMP language structure

Units abbreviations

The syntax table below uses the following abbreviations for units

Abbreviation	Details
mm	millimetres
°	degrees
mm or °	millimetres or degrees depending on whether the corresponding data is projected or geodetic
Z	units of the corresponding Z field data
chr	character positions
lin/cm^2	lines per square centimetre

Notation for alternatives

If there are possible alternative sets of statements in a block, all statement definitions will start with '**.**' (Or sometimes '**-**'). The alternative sets of statements will be separated by the word OR. The following example shows that you can use a single **`Colour =`** statement or a four line **`Colour Begin ... End`** statement block.

`.Colour=<colour>`

`OR`
`Colour Begin`

`.Z=<path>`
`.Legend=<path>`
`.Colour End`

In some cases a number of objects so similar that we have used the same definition for them. In this case we list the alternatives at the beginning of the definition and note any individual differences as they occur. For example, the **`PathPlot`** and **`StackProfilePlot`** specifications are very similar. The additional statements possible for a **`StackProfilePlot`** are shown with the notation '(Stack Profile only)'.

Syntax table

Statement	Description	Unit	Default
Group Objects			
	Non-aligning and Centre groups definition. This applies to Page Box Centre		
`Box(/Page/Centre) Begin`	Object definition		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`Width=<number>`	Width of object Defaults: Page: 210 Other: depends on contained objects	mm	
`Height=<number>`	Height of object Defaults: Page: 297 Other: depends on contained objects	mm	
`... Begin`	Contained object definition(s)		
`...`			
`... End`			
`...`	(more contained objects if required) (Centre usually has only one object)		
`Box(/Page/Centre) End`			

Statement	Description	Unit	Default
Margin Begin	**Margin definition**		
X=<number>	Position of object—X coordinate	mm	0
Y=<number>	Position of object—Y coordinate	mm	0
Top=<number>	Width of top edge margin	mm	2
Bottom=<number>	Width of bottom edge margin	mm	2
Left=<number>	Width of left edge margin	mm	2
Right=<number>	Width of right edge margin	mm	2
Internal=<YES|NO>	Border added to (NO) or subtracted from (YES) size	mm	NO
... Begin	Contained object definition		
...			
... End			
...	(more contained objects if required)		
Margin End			
Border Begin	**Border definition**		
X=<number>	Position of object—X coordinate	mm	0
Y=<number>	Position of object—Y coordinate	mm	0
Thickness=<thickness>	Thickness of border line		1
Colour=<colour>	Colour of border line		black
Style=<linestyle>	Style of border line		solid
... Begin	Contained object definition		
...			
... End			
...	(more contained objects if required)		
Border End			
HBox Begin	**HBox object definition**		
X=<number>	Position of object—X coordinate	mm	0
Y=<number>	Position of object—Y coordinate	mm	0
... Begin	Contained object definition		
...			
... End			
VSpace Begin			
Space=<number>	Vertical space between contained objects	mm	0
VSpace End			
... Begin	Contained object definition		
...			
... End			
...	(more contained objects and VSpace if required)		
HBox End			
	Horizontal row object definitions This applies to Top VCentre Bottom		
Top(/VCentre/Bottom) Begin	Object definition		
X=<number>	Position of object—X coordinate	mm	0
Y=<number>	Position of object—Y coordinate	mm	0
... Begin	Contained object definition		
X=<number>	Position of contained object—X coordinate	mm	0
...			
... End			
...	(more contained objects if required)		
Top(/VCentre/Bottom) End			

Statement	Description	Unit	Default
`VBox Begin`	**VBox object definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`Width=<number>`	Width of object Default: width of largest object	mm	
`Height=<number>`	Height of object Default: combined height of all objects and `HSpace`	mm	<–
`... Begin`	Contained object definition		
`...`			
`... End`			
`HSpace Begin`			
`Space=<number>`	Horizontal space between contained objects	mm	0
`HSpace End`			
`... Begin`	Contained object definition		
`...`			
`... End`			
`...`	(more contained objects and HSpace if required)		
`VBox End`			
	Vertical column object definition. This applies to Left HCentre Right		
`Left Begin`	Object definition		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`... Begin`	Contained object definition		
`Y=<number>`	Position of contained object—Y coordinate	mm	0
`...`			
`... End`			
`...`	(more contained objects if required)		
`Left End`			
`Flexible Begin`	**Flexible object definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`Width=<number>`	Width of object	mm	
`Height=<number>`	Height of object	mm	
`Isotropic=<YES|NO)`	Yes: Preserve aspect ratio of contained object No: Stretch contained object to fit the box		YES
`... Begin`	Contained object definition		
`...`			
`... End`			
`Flexible End`			
`Include Begin`	**Include MAPCOMP file object definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`File=<path>`	MAPCOMP file to be included		
`Include End`			

Statement	Description	Unit	Default
`Data Begin`	**Data object definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`Width=<number>`	Width of object	mm	100
`Height=<number>`	Height of object	mm	100
`MapProjection Begin`	Projection definition		
`Projection= <filename>`	Projection for Data object Default: Projection of first contained object		<–
`Datum=<filename>`	Datum for Data object Default: Datum of first contained object		<–
`Rotation= <CENTRE|number>`	Rotate the data in the **`Data`** object. **`CENTRE`**:Rotate data so that on the vertical centre line of the **`Data`** object, the data is oriented North–South number:Rotate the data so that the North direction in the data is at this angle (measured clockwise where 0 is the vertical centre line)	°	no rotation
`XScale=<number>`	Scale for data box in X direction		to fit data object
`YScale=<number>`	Scale for data box in Y direction		
`MapProjection End`			

Statement	Description	Unit	Default
`MapExtent Begin`	Extent definition		
`.XMin=<number>`	Geographic X coordinate in lower left corner Default: Geo-graphic extent of all datasets in the Data object	m	
`.XMax=<number>`	Geographic X coordinate of top right corner of data shown in composition	m	
`.YMin=<number>`	Geographic Y coordinate in lower left corner	m	
`.YMax=<number>`	Geographic Y coordinate of top right corner of data shown in composition	m	
`.XsheetOverlap= <number>`	Size of sheet overlap in the X direction	m	0
`.YsheetOverlap= <number>`	Size of sheet overlap in the Y direction	m	0
`.XsheetSize= <number>`	X direction distance represented in one sheet	m	automatically calculated
`.YsheetSize= <number>`	Y direction distance represented in one sheet	m	
`. OR`			
`.LongMin= <deg:min:sec>`	Geographic Long coordinate in lower left corner Default: Geographic extent of all datasets included in the Data object	°	<–
`.LongMax= <deg:min:sec>`	Geographic Long coordinate of top right corner of data shown in composition	°	
`.LatMin=<deg:min:sec>`	Geographic Lat coordinate in lower left corner	°	
`.LatMax=<deg:min:sec>`	Geographic Lat coordinate of top right corner of data shown in composition	°	
`.LongSheetOverlap= <deg:min:sec>`	Size of sheet overlap in the Long direction	°	0
`.LatSheetOverlap= <deg:min:sec>`	Size of sheet overlap in the Lat direction	°	0
`.LongSheetSize= <deg:min:sec>`	Long direction distance represented in one sheet	°	automatically calculated
`.LatSheetSize= <deg:min:sec>`	Lat direction distance represented in one sheet	°	
`NXSheets=<number>`	No of sheets across		1
`NYSheets=<number>`	No of sheets down		1
`XSheet=<number>`	Column of current sheet to display in Map Composition tool window		1 (left)
`YSheet=<number>`	Row of current sheet to display in Map Composition tool window		1 (top)

`Sheets={`
`<string>`
`<YES|NO>`
`<number> <number>`
`<number> <number>`
`<number> <number>`
`...}`
	Title of sheet
	Will this sheet be printed?
	X Origin Y Origin
	X Extent Y Extent
	X Overlap Y Overlap
	next sheet(s) as above
		m or °	

`MapExtent End`			
`... Begin`	Contained geographically located object definition inside Data object		
`...`			
`... End`			
`...`	(more geographic objects if required)		
`Data End`	End statement for Data object		
Geographically Located Objects (contained by Data object)			
`GreyScale Begin`	**GreyScale Image definition**		
`Image=<path>`	Image Dataset		
`Legend=<path>`	Legend file		automatic
`Lut=<filename>`	Lookup table for legend		greyscale.lut
`GreyScale End`			

Statement	Description	Unit	Default
`PseudoColour Begin`	**PseudoColour Image definition**		
`Image=<path>`	Image Dataset		
`Legend=<path>`	Legend File (optional)		automatic
`Lut=<filename>`	Lookup table for legend		pseudocolour.lut
`PseudoColour End`			
`FixedColour Begin`	**Fixed Colour Image definition**		
`Image=<path>`	Image Dataset		
`Legend=<path>`	Legend File (optional)		automatic
`FixedColour End`			
`SunAngle Begin`	**Sun Angle Image definition**		
`Image=<path>`	Image Dataset		
`Legend=<path>`	Legend File		automatic
`Lut=<filename>`	Lookup table for legend		greyscale.lut
`Declination=<number>`	Sun Declination	°	45
`Inclination=<number>`	Sun Inclination	°	45
`VerticalEx=<number>`	Vertical Exaggeration		100
`SunAngle End`			
`FalseColour Begin`	**False Colour Image definition**		
`ImageRed=<path>`	Image Dataset for red colour		
`LegendRed=<path>`	Legend File for red colour		automatic
`ImageGreen=<path>`	Image Dataset for green colour		
`LegendBlue=<path>`	Legend File for green colour		automatic
`ImageBlue=<path>`	Image Dataset for blue colour		
`LegendBlue=<path>`	Legend File for blue colour		automatic
`FalseColour End`			
`Drape Begin`	**Drape Images definition**		
`ImagePseudoColour =<path>`	Image Dataset for pseudocolour		
`LegendPseudoColour =<path>`	Legend File for pseudocolour		automatic
`Lut=<filename>`	Lookup table for legend		pseudocolour.lut
`SaturationFactor= <0..1>`	Colour saturation for pseudocolour		1
`Transparency= <0..1>`	Cutoff level for high intensity values. Intensity data not shown if its Intensity value exceeds the Transparency value.		1
`.ImageIntensity=<path>`	Image Dataset for Intensity		
`.LegendIntensity=<path>`	Legend File for Intensity		automatic
`OR SunAngleOp Begin`	Sun AngleOperation definition		
`.ImageIntensity =<path>`	Image Dataset for Intensity		
`.Declination =<-180..180>`	Sun Declination for Intensity	°	45
`.Inclination=<0..90>`	Sun Inclination for Intensity	°	45
`.VerticalEx=<number>`	Vertical Exaggeration for Intensity		100
`.SunAngleOp End`			
`Drape End`			

Statement	Description	Unit	Default
`TernaryDrape Begin`	**Ternary Drape Images definition**		
`ImageRed=<path>`	Image Dataset for red colour		
`LegendRed=<path>`	Legend File for red colour		automatic
`ImageGreen=<path>`	Image Dataset for green colour		
`LegendBlue=<path>`	Legend File for green colour		automatic
`ImageBlue=<path>`	Image Dataset for blue colour		
`LegendBlue=<path>`	Legend File for blue colour		automatic
`.ImageIntensity=<path>`	Image Dataset for Intensity		
`.LegendIntensity=<path>`	Legend File for Intensity		automatic
`OR SunAngleOp Begin`	Sun AngleOperation definition		
`.ImageIntensity =<path>`	Image Dataset for Intensity		
`.Declination =<-180..180>`	Sun Declination for Intensity	°	45
`.Inclination=<0..90>`	Sun Inclination for Intensity	°	45
`.VerticalEx=<number>`	Vertical Exaggeration for Intensity		100
`.SunAngleOp End`			
`TernaryDrape End`			
`Contour Begin OR ColourContour Begin`	**Contour/Colour Contour definition**		
`Detail= <Draft|Outline|Full>`	Detail level for display in Map Composition tool		Draft
`Grid=<path>`	Dataset for contours		
`Legend=<path>`	Legend file *(ColourContour only)*		automatic
`LowClip=<number>`	Lowest contour value to be shown	Z	Min Z val
`HighClip=<number>`	Highest contour value to be shown	Z	Max Z val
`GapBetweenLabels =<number>`	Distance between labels	mm	100
`DrawIncrement=<number>`	Minimum size of plotter movements	mm	0.5
`Tension=<0..50>`	Spline tension		1
`HighLow Begin`	High and low point settings definition		
`SearchRadius= <number>`	Radius of area to be scanned for other high/low values. Default (–1) = 8 cells	m	–1
`Tolerance=<number>`	Minimum difference from all neighbouring values (within search radius) to qualify as a high/low value	Z	0
`ShowLow=<YES|NO>`	Include low point annotations?		NO
`LowSymbol= <char|symbol>`	Symbol for low point annotations		L
`LowSymbolSize= <symbolsize>`	Size of low point annotations	mm	2
`LowSymbolColour= <colour>`	Colour of low point annotations		black
`ShowHigh=<YES|NO>`	Include high point annotations?		NO
`HighSymbol= <char|symbol>`	Symbol for high point annotations		H
`HighSymbolSize= <symbolsize>`	Size of high point annotations	mm	2
`HighSymbolColour= <colour>`	Colour of high point annotations		black
`ShowValue=<YES|NO>`	Include value annotations?		NO
`TextSize=<number>`	Size of value annotations text	mm	2
`TextColour=<colour>`	Colour of value annotations text		black
`TextThickness= <thickness>`	Value annotations text weight (thickness of lines making up characters)	mm	0
`Decimals=<number>`	Number of decimal places shown by value annotations		0
`Angle=<number>`	Angle of value annotations text	°	0
`HighLow End`			

Statement	Description	Unit	Default
Cut Begin	Contour cut definition		
Interval=<number>	Interval between contours	Z	none
Density=<number>	Maximum density of contours	lin/ cm^2	
LineColour=<colour>	Fixed colour of contours (not used in ColourContour)		black
Annotate=<YES|NO>	Include value annotations on contours?		NO
TextSize=<number>	Size of value annotations text	mm	4
TextColour=<colour>	Colour of value annotations text		black
Decimals=<number>	Number of decimal places shown by value annotations		0
Cut End			
Cut Begin	Further contour cuts if required		
...			
Cut End			
Contour End			
PointPlot Begin	**Point Plot definition**		
Dataset=<path>	Point Dataset		
Marker Begin	Marker definition		
.Colour=<colour>	Fixed marker colour		black
ORColour Begin	Marker colours from Z field definition		
.Z=<path>	Z field for marker colour		
.Legend=<path>	Legend File for marker colour		automatic
.Colour End			
-Size=<number>	Fixed marker size[5]	mm	2
ORSize Begin	Marker size from Z field definition		
-Z=<path>	Z field for marker size		
-Legend=<path>	Legend File for marker size		automatic
-Size End			
.Thickness =<thickness>	Fixed marker line thickness[a0]	mm	1
ORThickness Begin	Marker line thickness from Z field definition		
.Z=<path>	Z field for marker line thickness		
.Legend=<path>	Legend File for marker line thickness		automatic
.Thickness End			
-Symbol=<symbol>	Fixed marker symbol		square
ORSymbol Begin	Marker symbols from Z field definition		
-Z=<path>	Z field for marker symbols		
-Legend=<path>	Legend File for marker symbols		automatic
-Symbol End			
.Angle=<angle>	Fixed marker angle (rotation about centre of marker measured anticlockwise)	°	0
ORAngle Begin	Marker angles from Z field definition		
.Z=<path>	Z field for marker angles	°	
.Angle End			
Marker End			

Statement	Description	Unit	Default
Text Begin	Text definition for point marker labels		
.String=<string>	Marker labels text		
ORString Begin	Marker labels from Z field definition		
.Z=<path>	Z field for marker labels		
.String End			
Colour=<colour>	Marker labels text colour		black
Size=<number>	Marker labels text size	mm	2
Font=<typeface>	Marker labels typeface		0
TextThickness= <thickness>	Marker label text weight (thickness of lines making up characters)	mm	0
Angle=<number>	Marker labels text angle (rotated anticlockwise from horizontal)	°	0
Justify = <justification>	Marker label text justification		lb
Gap=<number>	Leading space between left of text object and first character	chr	0
VGap=<number>	Additional inter-line spacing	mm	0
Text End			
PointPlot End			
PathPlot Begin OR StackProfilePlot Begin	**Path Plot and Stack Profile Plot definition**		
Detail= <Draft|Outline|Full>	Detail level for display in Map Composition tool		Draft
Dataset=<path>	Line dataset		
Z=<path>	Z field for profile *(Stack Profile only)*		
VScale=<number>	Vertical scale for profile *(Stack Profile only)*	Z/ cm	100
Base=<number>	Base value for stack profile *(Stack Profile only)*	Z	mean Z value
LeastSquaresPath= <YES|NO>	**YES**: Calculate line of best fit over each profile base line segment **NO**: Use first and last points of profile base line segments for line *(Stack Profile only)*		YES
TraverseLine Begin OR ProfileLine Begin	Traverse line definition within path plot block		
.Colour=<colour>	Fixed line colour		black
ORColour Begin	Line colour from Z field definition		
.Z=<path>	Z field for line colour		
.Legend=<path>	Legend file for line colour		automatic
.Colour End			
Thickness= <thickness>	Thickness of line		1
Style=<style>	Style of line (*bipole* for Path Plot only)		solid
Zdata Begin	Bipole style Z field definition *(Path Plot bipole style only)*		
Z=<path>	Z field for bipole style		
Legend=<path>	Legend file for bipole style. Recommended: ***install_path*/lut/bipole.leg**		
Zdata End			
ShowDirection= <YES|NO>	Show arrows on the ends of lines?		NO
ShowLineToBase= <YES|NO>	Show a line from plotted traverse line end to original flight path position for that point? *(Stack Profile only)*		NO
ShowLineToFiducial= <YES|NO>	Show lines from recovery points or fiducial marker locations on plotted lines to original flight path positions for those points? *(Stack Profile only)*		NO
ViewDirection= <number>	Perspective direction at which profile is drawn *(Stack Profile only)*	°	–45

Statement	Description	Unit	Default
`LineNumber Begin`	Line number field identification		
`Z=<path>`	Line number field		
`LineNumber End`			
`LineNumberText Begin`	Line number text definition (usually for line number)		
`Colour=<colour>`	Line number text colour		black
`Size=<number>`	Line number text size	mm	2
`Font=<typeface>`	Line number text typeface		0
`Angle=<number>`	Line number text angle from line direction at end: anticlockwise at East end, clockwise at West end of line (must be different from **`LineNumber2`**)	°	0
`TextThickness= <thickness>`	Line number text weight (thickness of lines making up characters)	mm	0
`Justify = <justification>`	Line number text justification		lb
`Gap=<number>`	Leading space between left of text object and first character	chr	0
`VGap=<number>`	Additional inter-line spacing	mm	0
`LineNumberText End`			
`LineNumber2 Begin`	Second line number field identification		
`Z=<path>`	Second line number field		
`LineNumber2 End`			
`LineNumber2Text Begin`	Second line number text definition (usually for flight number or date)		
`Colour=<colour>`	Second line number text colour		black
`Size=<number>`	Second line number text size	mm	2
`Font=<typeface>`	Second line number text typeface		0
`Angle=<number>`	Second line number text angle from line direction at end: anticlockwise at East end, clockwise at West end of line (must be different from **`LineNumber`**)	°	0
`TextThickness= <thickness>`	Second line number text weight (thickness of lines making up characters)	mm	0
`Justify = <justification>`	Second line number text justification		lb
`Gap=<number>`	Leading space between left of text object and first character	chr	0
`VGap=<number>`	Additional inter-line spacing	mm	0
`LineNumber2Text End`			
`Fiducial Begin`	Fiducial field identification		
`Z=<path>`	Fiducial field		
`Fiducial End`			
`ShowFiducials= <YES|NO>`	Show fiducial markers?		NO
`RecoveryType= <FiducialIncrement |RecoveryField>`	Plot Fiducials or Recovery points		FiducialIncrement
`.RecoveryField=<path>`	Recovery field name		
`ORFiducialInterval= <number>`	Number of fiducial values from one fiducial marker to the next		100
`.FiducialLabelRate= <number>`	Number of fiducial markers from one label to the next		10
`FiducialText Begin`	Fiducial numbers text definition		
`Colour=<colour>`	Fiducial numbers text colour		
`Size=<number>`	Fiducial numbers text size	mm	
`Font=<typeface>`	Fiducial numbers text typeface		
`Angle=<number>`	Fiducial numbers text angle	°	
`TextThickness= <thickness>`	Fiducial numbers text weight (thickness of lines making up characters)	mm	
`Justify = <justification>`	Fiducial numbers text justification		
`Gap=<number>`	Leading space between left of text object and first character	chr	
`VGap=<number>`	Additional inter-line spacing	mm	
`FiducialText End`			Z

Statement	Description	Unit	Default
`FiducialMarker Begin`	Fiducial marker definition		
`Colour=<colour>`	Colour of fiducial markers		black
`Size=<symbolsize>`	Size of fiducial markers	mm	2
`.Symbol=<symbol>`	Fixed symbol for fiducial markers		cross
`ORSymbol Begin`	Marker symbols from Z field definition		
`.Z=<path>`	Z field for fiducial markers		
`.Legend=<path>`	Legend file for fiducial markers		automatic
`.Symbol End`			
`FiducialMarker End`			
`TraverseLine End OR ProfileLine End`			
`PathPlot End OR StackProfilePlot End`			
`PolygonPlot Begin`	**Polygon Plot definition**		
`Poly Begin`	Polygon definition		
`Dataset=<path>`	Polygon dataset		
`Colour=<colour>`	Polygon colour		black
`Fill=<filltype>`	Polygon fill type		hollow
`Thickness= <thickness>`	Line thickness	mm	0
`Poly End`			
`ShowLabel=<YES|NO>`	Display a label for the polygon?		NO
`Text Begin`	**Text definition**		
`.String=<string>`	Polygon label text		
`ORString Begin`	Polygon label from Z field definition		
`.Z=<path>`	Z field for polygon label		
`.String End`			
`Colour=<colour>`	Polygon label colour		black
`Size=<number>`	Polygon label text size	mm	2
`Font=<typeface>`	Polygon label typeface		0
`Angle=<angle>`	Polygon label angle	°	0
`Justify=<justify>`	Polygon label justification		lb
`Text End`			
`PolygonPlot End`			

Statement	Description	Unit	Default
Geographically Located Attributes (contained by Data object)			
Ticks Begin	**Metre/Latitude Longitude Ticks definition**		
MetreGrid=<YES|NO>	Show metre ticks (YES) or latitude longitude ticks (NO)		YES
.EastInterval=<number>	Interval between Easting ticks *(for metre ticks only)*	°	
.NorthInterval=<number>	Interval between northing ticks *(for metre ticks only)*	°	
OR LongInterval= <deg:min:sec>	Interval between longitude ticks *(for lat/long ticks only)*	°	
.LatInterval= <deg:min:sec>	Interval between latitude ticks *(for lat/long ticks only)*	°	
LabelAtLeft=<YES|NO>	Place latitude/northing labels at the left of the data object?		NO
LabelAtRight=<YES|NO>	Place latitude/northing labels at the right of the data object?		NO
LabelAtTop=<YES|NO>	Place longitude/Easting labels at the top of the data object?		NO
LabelAtBottom=<YES|NO>	Place longitude/Easting labels at the bottom of the data object?		NO
Style= <TICK|GRID|BORDER>	Place + marks (Tick) or a grid (Grid) inside the data object or tick marks on the inside edge of the data object (Border)		TICK
Internal=<YES|NO>	Display tick labels inside (YES) or outside (NO) the data object.		NO
TickSize=<number>	Size of the ticks	mm	3
TickThickness= <thickness>	Thickness of the tick lines	mm	0
TextSize=<number>	Tick labels text size	mm	3
TextFont=<typeface>	Tick label text typeface		0
TextThickness= <thickness>	Tick label text weight (thickness of lines making up characters)	mm	0
LabelOffset=<number>	Offset of label away from border	mm	0
Ticks End			
CornerBegin	**Corner Extent Labels definition**		
Horizontal=<YES|NO>	Text orientation		YES
HGap=<number>	Horizontal distance from data object corner	mm	1
VGap=<number>	Vertical distance from data object corner	mm	1.5
Format= <D|DM|DMS>	Degrees OR Degrees:Minutes OR Degrees:Minutes:Seconds		DMS
CornerText Begin	Corner label text definition		
TextThickness= <thickness>	Corner labels text weight (thickness of lines making up characters)	mm	0
CornerText End			
Corner End			
Annotation objects that depend on a Data object			
ScaleBar Begin	**Scale Bar definition**		
X=<number>	Position of object—X coordinate	mm	
Y=<number>	Position of object—Y coordinate	mm	
Length=<number>	Length of scale bar (height is automatic)	mm	100
Interval=<number>	Metres/degrees between ticks	m or °	10
Unit=<METRES|LATLONG>	Unit shown in scale bar		METRES
ShowScale=<YES|NO>	Show composition scale bar		NO
ScaleBar End			

Statement	Description	Unit	Default
`NorthArrow Begin`	**North Arrow definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`Length=<number>`	Length of North arrow	mm	40
`GridNorth=<number>`	*For projected dataset only*: Forced rotation by you of North arrow from vertical (currently required for rotated dataset)	°	0
`TrueNorth=<number>`	*For projected dataset*: Displacement from vertical of secondary True North arrow *For geodetic dataset*: Forced rotation by you of North arrow from vertical (currently required for rotated dataset)	°	0
`MagneticNorth=<number>`	Displacement from vertical of secondary Magnetic North arrow	°	0
`ShowProjection=<YES|NO>`	Include projection information with North arrow		NO
`TextSize=<number>`	North arrow text size	mm	2
`TextFont=<typeface>`	North arrow text typeface		0
`TextThickness= <thickness>`	North arrow text weight (thickness of lines making up characters)	mm	0
`NorthArrow End`			
`SheetIndex Begin`	**Sheet Index definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`Width=<number>`	Width of object excluding margin	mm	User must define
`Height=<number>`	Height of object excluding margin	mm	
`BoxThickness= <thickness>`	Sheet index box line thickness	mm	0
`TextSize=<number>`	Sheet Index font size	mm	2
`TextFont=<typeface>`	Sheet Index text typeface		3
`TextThickness= <thickness>`	Sheet Index text weight (thickness of lines making up characters)	mm	0
`XSheets=<number>`	No of sheets across (columns)		
`YSheets=<number>`	No of sheets across (rows)		
`Names={ <string> <string> ...}`	List of names		
`Sheet=<string>`	Current sheet name (one of above list)		
`SheetIndex End`			
Annotation objects that describe grid or Z data			
`Legend Begin`	**Legend definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`Width=<number>`	Width of object excluding margin	mm	
`Height=<number>`	Height of object excluding margin	mm	100
`Name = <path>`	Legend file		
`ShowHighClip=<YES|NO>`	Show colour for high clip of data		YES
`ShowLowClip=<YES|NO>`	Show colour for low clip of data		YES
`ShowOutOfRange=<YES|NO>`	Show colour for out of range data		NO
`Decimals=<number>`	Decimal places for cutoff values		
`Legend End`			

Statement	Description	Unit	Default
MultiPlot Begin	**MultiPlot definition (replaces XYPlot)**		
Panel Begin	Panel definition		
XYData Begin	Specification of data to plot in panel		
X = <path>	Field for X axis (omit for multiband radiometrics data)*		
Y = <path>	Field for Y axis*		
Group = <number>	Dataset group (for example, line) to be plotted		1
StartBand = <number>	(Multiband plotting only) Lower and upper limits of band number range to plot (For truncating the spectra channel numer range)		
EndBand = <number>			
StartRow = <number>	Lower and upper limits of data point numbers to plot. (For multiband spectra, you normally only plot one point, so StartRow = EndRow)		
EndRow = <number>			
XYData End			
XAxis Begin	X Axis definition		
Scale = <number>	Scale for X axis		1000
XRangeStyle = <automatic | manual>	automatic : X axis range = range of datamanual : X axis range defined by XMin and XMax		automatic
XMax = <number>	Upper value limit for X axis range		
XMin = <number>	Lower value limit for X axis range		
XAxis End			
YAxis Begin	Y Axis definition		
Scale = <number>	Scale for Y axis		1000
YRangeStyle = <automatic | manual>	automatic : Y axis range = range of datamanual : Y axis range defined by YMin and YMax		automatic
YMax = <number>	Upper value limit for Y axis range		
YMin = <number>	Lower value limit for Y axis range		
YAxis End			
Line Begin	Line definition		
Colour = <colour>	Line colour		black
Thickness = <thickness>	Line thickness		
Style = <style>	Line style		
Line End			
Panel End			
MultiPlot End			
* If you specify X and Y, INTREPID will create a normal XY plot.If you specify a multiband spectra field as Y INTREPID will automatically plot channel number on the X axis.			
Annotation objects that do not depend on data from a dataset			
Greybar Begin	**Grey Bar definition**		
X=<number>	Position of object—X coordinate	mm	
Y=<number>	Position of object—Y coordinate	mm	
Width=<number>	Width of object excluding margin	mm	20
Height=<number>	Height of object excluding margin	mm	100
Vertical=<YES|NO>	Grey bar orientation: vertical (**YES**) or horizontal (**NO**)		YES
Greybar End			
Marker Begin	**Marker definition**		
X=<number>	Position of object—X coordinate	mm	0
Y=<number>	Position of object—Y coordinate	mm	0
Width=<number>	Width of object excluding margin	mm	
Height=<number>	Height of object excluding margin	mm	
XY={ <number> <number> ...}	Coordinates of markers within this object	mm	
Colour=<colour>	Marker colour		black
Size=<size>	Marker size		1
Symbol=<symbol>	Marker symbol		square
Marker End			

Statement	Description	Unit	Default

`Line Begin OR`
`Arrow Begin`
	Line or Arrow definition		

`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`XY={ <number> <number> ...}`	Coordinates of line vertices within this object	mm	
`Colour=<colour>`	Line colour		black
`Thickness=<thickness>`	Line thickness	mm	0
`Style=<style>`	Line style		solid

`Line End OR`
`Arrow End`			

`Polygon Begin`	**Polygon definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`XY={ <number> <number> ...}`	Coordinates of polygon vertices within this object	mm	
`Colour=<colour>`	Polygon colour		black
`Fill=<fill>`	Polygon fill type		hollow
`Polygon End`			
`Text Begin`	**Text definition**		
`X=<number>`	Position of object—X coordinate	mm	
`Y=<number>`	Position of object—Y coordinate	mm	
`XY=<number> <number>`	Displacement of text within object	mm	
`String={ <string> <string> ...}`	Text content (string). Each **`<string>`** occupies a separate line.[b1]		
`Colour=<colour>`	Text colour		black
`Size=<number>`	Text size	mm	2
`Font=<typeface>`	Text typeface		0
`Angle=<number>`	Text angle anticlockwise from horizontal	°	0
`TextThickness=<thickness>`	Text weight (thickness of lines making up characters)	mm	0
`Justify=<justification>`	Text position within object		lb
`Gap=<number>`	Leading space between left of text object and first character	chr	0
`VGap=<number>`	Additional inter-line spacing	mm	0
`Text End`			
`DGNInclude Begin`	**Include DGN file definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`File=<path>`	DGN file to be included		
`DGNInclude End`			
`Image Begin`	**Image definition**		
`X=<number>`	Position of object—X coordinate	mm	0
`Y=<number>`	Position of object—Y coordinate	mm	0
`.Columns=<number>`	No of columns in custom bitmap		
`.Rows=<number>`	No of rows in custom bitmap		
`.Pixels={<1..256> <1..256> ...}`	Pixel list for custom bitmap, from left to right across each row, colours from currently loaded lookup table (**`.lut`**) file		
`OR Pixels Begin`			
`.Format=<TIF|DGN|Intrepid>`	Format of image		
`.File=<path>`	Image for inclusion		
`.Pixels End`			
`Image End`			

a.[0] Special purpose symbol shapes use this attribute for other purposes (See "Special purpose symbol shapes" in Available map attribute values (R22) for details)

b.[1] See Text features for information about codes you can embed in these strings.

Sample MAPCOMP files

Here is a sample MAPCOMP file. This file will produce the solution to the Map Composition exercise in INTREPID *Guided Tours*.

```
Margin Begin
   Width = 210.000000
   Height = 297.000000
   Internal = No
   Top = 2.000000
   Bottom = 2.000000
   Left = 2.000000
   Right = 2.000000
   Border Begin
      X = 2.000000
      Y = 2.000000
      Width = 206.000000
      Height = 293.000000
      Thickness = 1.000000
      Colour = Black
      Style = Solid
      Page Begin
         Width = 206.000000
         Height = 293.000000
         Data Begin
            X = 42.177073
            Y = 101.383464
            Width = 119.700000
            Height = 119.700000
            MapProjection Begin
               Projection = TMAMG54
               Datum = AGD66
               XScale = 100000.000000
               YScale = 100000.000000
            MapProjection End
            MapExtent Begin
               Xmin = 740001.150000
               Xmax = 751971.150000
               Ymin = 8408029.770000
               Ymax = 8419999.770000
               XSheet = 0
               YSheet = 0
               NXSheets = 1
               NYSheets = 1
               Sheets = {
                  "SHEET 1" Yes
                  740001.150000
                  8408029.770000
                  11970.000000
                  11970.000000
                  0.000000
                  0.000000
               }
            MapExtent End
            PseudoColour Begin
               Width = 100.000000
               Height = 100.000000
              Image ={
                   D:/intrepid/tutorials/data/mlevel_grid
                     }
              Legend ={
                   D:/intrepid/tutorials/data/mlevel_grid
                       }
            PseudoColour End
            Ticks Begin
               MetreGrid = No
               LongInterval = 0:2:0
               LatInterval = 0:2:0
               LabelAtBottom = Yes
               LabelAtLeft = Yes
               Style = Tick
               Internal = No
               TextSize = 3.000000
               TickSize = 3.000000
            Ticks End
         Data End
         NorthArrow Begin
            X = 17.554884
            Y = 12.918093
            Width = 46.000000
            Height = 58.600000
            Length = 40.000000
            GridNorth =    0.0000000000
            TrueNorth =    0.0000000000
            MagneticNorth =  0.0000000
            ShowProjection = Yes
         NorthArrow End
         ScaleBar Begin
            X = 80.586713
            Y = 55.405865
            Width = 100.000000
            Height = 15.800000
            Length = 100.000000
            Interval = 20.000000
            Unit = Metres
            ShowScale = Yes
         ScaleBar End
         Legend Begin
            X = 171.119158
            Y = 121.988915
            Width = 26.000000
            Height = 100.000000
            Name = {
                   D:/intrepid/tutorials/data/mlevel_grid
                   }
            ShowHighClip = No
            ShowLowClip = No
            ShowOutOfRange = No
```

```
            Decimals = 0
         Legend End
         Text Begin
            X = -11.355379
            Y = -20.276701
            Width = 129.142857
            Height = 8.000000
            XY = {              44 268
            }
           String = {
                "Ebagoola Magnetics"
                   }
            Colour = Black
            Size = 8.000000
            Font = 5
            Angle =    0
            Justify = lb
         Text End
      Page End
   Border End
Margin End
```