Chapter 19 - COMMAND LANGUAGE REFERENCE

This section of the *User's Guide* contains an alphabetical sequenced list of LTPlus intrinsic functions, including a brief discription of each command and its usage, a summary of the required and optional command parameters (arguments), examples of command use, and reference(s) to other related, intrinsic functions.

The commands and their associated arguments appear in boldface and are always entered as lowercase characters.

Each command description includes a Chapter reference where the user may find additional, useful information.

Before any LTPlus map-based operations can be performed, a current, active map must be identified. The **acquire** command identifies a map which was previously created and/or processed.

acquire

Opens named data file for use.

acquire name

This function will open for access the data files associated with the specified name directory. If no map exists the user is asked if it should be created.

Options/arguments:

name Map name

Example(s):

To open the files comprising the **cub_creek** map, enter:

acquire cub_creek

See also:

create

One of the most important functions in LTPlus is **assemble**. It is responsible for transforming nonsensical raster information into topologically structured data. This command is generally executed after the raw data has been edited.

assemble

Vectorizes lines and constructs topology.

assemble [-n]

Vectorize lines (thin) and build topology for entire raster. Raw, vectorized data will be automatically reduced unless the -n option is specified.

Options/arguments:

-n Suppresses reduction of raw data

Example(s):

To assemble the current, active map without automatic reductions, enter:

assemble -n

See also:

The **atrbt_concat** command is useful for combining multiple attributes assigned to an object into a single string. Some GIS applications cannot handle multiple attributes; this command will make the attribute entries palatable for such applications. The concatenated attribute string will be stored according to the action code parameter.

atrbt_concat

Concatenates attributes for objects in specified set.

atrbt_concat s action

Concatenates all atributes for all objects in specified set s.

Options/arguments:

s Set number, where:
0 - 31, inclusive, are valid

action Action code, where:
0 = Replace all attributes w/single new one.
1 = Replace/set 1st attribute, leave others.
2 = Replace/set 2nd attribute, leave others.
3 = Insert new 1st attribute, push others up by one.
4 = Prefix 1st existing attribute.
5 = Suffix 1st existing attribute.
6 = Add a new (additional) last attribute.
7 = Remove all attributes for one object.

Example(s):

To concatenate individual attributes for each object in set number **8** for the current map and replace all attributes with the new one, enter:

atrbt_concat 8 0

More specifically, if an object in that set were to have three attributes (AW99, RW78 and PY76), application of the **atrbt_concat** function would result in the same object having an attribute of **AW99RW78PY76** as one string.

See also:

All **crit_** and **set_** functions

The **atrbt_contur** command establishes a mode for associating elevational attributes with contour line objects. It is an extremely efficient method for attaching the attributes by identifying increment or decrement profiles.

atrbt_contur

Sets mode for contour digitizing.

atrbt_contur action [elev]

Sets the graphics input mode to digitize contour attributes. The **start** function is executed to begin a linear profile. The profile is defined by subsequent execution of **increment** and **decrement** functions. If **elev** is specified, that value will be assigned to the first line crossed by the profile. Otherwise, the beginning elevation attribute will be acquired from the first contour line crossed by the profile; it must already have a valid, numeric attribute. Subsequent contour lines will be assigned elevation attributes based upon an **increment/ decrement** profile and the contour interval value from the map's **header** file. The action code will establish whether or not multiple attributes are allowed (for checking). If a mistake is made while in **atrbt_contur** mode (e.g., user presses **increment** when they meant to press **start** and many contours are mislabeled as a result), then executing the **undo** function will remove all attributes from the mislabeled contours. While this doesn't restore the correct labels (attributes), it does remove the incorrect ones and identifies the contours which need to be relabeled.

Options/arguments:

action	Attribute action code, where: 0 = Replace all attributes w/single new one. 1 = Replace/set 1st attribute, leave others. 2 = Replace/set 2nd attribute, leave others. 3 = Insert 1st new attribute, increment others by one 4 = Prefix 1st existing attribute. 5 = Suffix 1st existing attribute. 6 = Add a new (additional) last attribute. 7 = Remove ALL attributes for one object. 8 = Prefix 2nd existing attribute. 9 = Suffix 2nd existing attribute.
elev	Elevation (attribute) value; numeric

Example(s):

To begin a linear elevation profile for the attributing of lines (1 attribute only) with a starting elevation of **4600** feet, enter:

atrbt_contur 0 4600

If a valid elevational attribute has already been assigned to the line which will be crossed first along the linear elevation profile, and if multiple attributes are to be allowed, enter:

atrbt_contur 3

See also:

atrbt_dig, decrement, header, increment, start

The **atrbt_copy** command sets the graphic input mode to copy attributes from one object to another using the current, active graphics input device (**GIN**).

atrbt_copy

Sets graphic input mode for interactive copy of attribute from one object to another using input from the current GIN.

attrbt_copy otype action

Sets the graphics input mode to allow copying of attributes to objects using the current active graphics input device (**GIN**). The attribute string is specified by using the **start** command while the cursor is located on an object already has the desired attribute. This will locate the polygon, find its first attribute and display it. The attribute is assigned by using the **continue** command while the cursor is located on the object to which the attribute is to be assigned. Mulitple uses of the **continue** command allow assigning the same attribute to multiple objects. At any time a new attribute string may be specified by using the **start** command again. The attribute is applied to the **otype** object(s) as defined by the action code. The **increment** and **decrement** commands will behave like the **continue** command.

Options/arguments:

otype	Object type code, where: **0** = node 1 = segment **2** = polygon (def=same as map_type value in header)
action	Action code, where: **0** = Replace all attributes w/single new one. **1** = Replace/set 1st attribute, leave others. **2** = Replace/set 2nd attribute, leave others. **3** = Insert new 1st attribute, push others up by one. 4 = Prefix 1st existing attribute. **5** = Suffix 1st existing attribute. **6** = Add a new (additional) last attribute. **7** = Remove ALL attributes for one object. **8** = Prefix 2nd existing attribute. **9** = Suffix 2nd existing attribute.

Note that NO special variable sequences are allowed.

Example(s):

To copy the 1st attribute from the polygon at location **458, 122** to the 1st attribute for the polygons at **199, 200** and **53,466** enter:

atrbt_copy 2 1
start 458 122
continue 199 200
continue 53 466

To copy the 1st attribute from the polygon at location 458, 122 to the 2nd attributes for the polygons at 199, 200 and 53,466 enter:

atrbt_copy 2 2
start 458 122
continue 199 200
continue 53 466

To copy the 1st attribute from the node at 458,122 to replace all attributes for the nodes at 199,200 and 53,466 enter:

atrbt_copy 0 0
start 458 122
continue 199 200
continue 53 466

See also:

atrbt_rec, atrbt_serial, atrbt_set, atrbt_list, atrbt_dig

The **atrbt_dig** command sets the graphic input mode to assign attributes from the current, active graphics input device (**GIN**).

atrbt_dig

Sets graphic input mode for attribute assignment from the current **GIN**.

attrbt_dig otype action attr

Sets the graphics input mode to allow assignment of attributes to objects using the current active graphics input device (**GIN**). The attribute string specified by **attr** is applied to the **otype** object(s) as defined by the action code. This mode is actively used by a **continue** command. The action applying the **attr** string to the **otype** object will occur at every X,Y position. The **start**, **increment** and **decrement** commands will behave like the **continue** command.

If **$++** is imbedded within the attribute string, it will be replaced with a 5 character string (**xxxxx**) which is a whole number >= 0. Each time an attribute containing the **$++** string is assigned, the value is incremented. The initial value can be reset any time by using the **plus_plus** command.

Options/arguments:

otype	Object type code, where: **0** = node **1** = segment **2** = polygon (def=same as **map_type** value in **header**)
action	Action code, where: **0** = Replace all attributes w/single new one. **1** = Replace/set 1st attribute, leave others. **2** = Replace/set 2nd attribute, leave others. **3** = Insert new 1st attribute, push others up by one. **4** = Prefix 1st existing attribute. **5** = Suffix 1st existing attribute. **6** = Add a new (additional) last attribute. **7** = Remove ALL attributes for one object. **8** = Prefix 2nd existing attribute. **9** = Suffix 2nd existing attribute.
attr	Attribute string (maximum 30 characters long)

Note that the following special variable sequences are allowed:

$record

This special attribute variable is converted to the object record number when the attribute is assigned. For example: if the attribute string entered by the user is **'abc$record'** and is assigned using **atrbt_dig**, **atrbt_rec**, **atrbt_serial**, or **atrbt_set** to object **123**, then the attribute assigned is **'abc123'**.

$serial

This special attribute variable is converted to the object serial number when the attribute is assigned. For example: if the attribute string entered by the user is **'ab$serialzz'** and is assigned using **atrbt_dig**, **atrbt_rec**, **atrbt_serial**, or **atrbt_set** to an object with a serial number of **999**, then the attribute assigned is **'ab999zz'**.

$++

This special attribute variable is replaced by a 5 digit integer which can be initialized using the **plus_plus** command. Each time an attribute is stored using the **$++** variable substring, the internal value is incremented by 1.

Example(s):

To assign the attribute **HW678** to the polygon containing the location defined by any subsequent **continue** command, enter:

atrbt_dig 2 3 HW678

To replace the attribute assigned to any segment identified by a subsequent **continue** command with an attribute of **JK782**, enter:

atrbt_dig 1 0 JK782

See also:

atrbt_rec, atrbt_serial, atrbt_set, plus_plus

The **atrbt_edit** command invokes an *ascii*editor for viewing and modifying an existing attribute file.

atrbt_edit

Invokes and *ascii* editor, identified in the **config** file, for viewing/editing of the attributes for the current, active map.

atrbt_edit s

Invokes an *ascii* editor for viewing/editing the attributes assigned to the objects specified in set **s**.

Options/arguments:

s Set number, where:
0 - 31, inclusive, are valid set numbers

Example(s):

To activate an *ascii* keyboard editor to modify the attributes for all objects on the current map, enter:

atrbt_edit 0

To activate an *ascii* keyboard editor to modify the attributes for all objects in set **14** of the current, active map, enter:

atrbt_edit 14

See also:

atrbt_print

The **atrbt_line** command sets the graphics input mode to digitize line attributes between any two nodes.

atrbt_line

Sets mode for digitizing all line attributes between any two nodes.

atrbt_line action [mp=xxx] attr

Sets the graphics input mode to digitize line attributes between any two nodes. When in **atrbt_line** mode, the cursor is located near a node at one end of a series of segments and **start** is executed. The starting node will be shown in a cursor box. The cursor cross-hair is then moved to the other end (or an intermediate point) of the series of segments and **continue** is executed. The program then finds the shortest path between the two nodes and attributes all segments along the path with the input attribute and attribute action. Subsequent presses of **continue** will continue the process from the previous endpoint.

If **$mp** is imbedded within the attribute string (**attr**), it will be replaced with a 7 character string (**xxxxx.xx**) which is the number of miles from the beginning node. The beginning node can be given a non-zero value by using the **mp=xxx** optional parameter. The **$mp** string will always be in map miles. Each **$mp** string will be at least 0.01 greater than the preceding one.

If a **$++** is imbedded within the attribute string, it will be replaced with a 5 character string (**xxxxx**) which is a whole number >=0. Each time an attribute containing the **$++** string is assigned, the value is incremented. The initial value can be reset any time by using the **plus_plus** command.

Options/arguments:

action	Attribute action code **0** = Replace all attributes w/single new one. **1** = Replace/set 1st attribute, leave others. **2** = Replace/set 2nd attribute, leave others. **3** = Insert 1st new attribute, push others up by one. **4** = Prefix 1st existing attribute. **5** = Suffix 1st existing attribute. **6** = Add a new (additional) last attribute. **7** = Remove All attributes for one object. **8** = Prefix 2nd existing attribute. **9** = Suffix 2nd existing attribute.
mp=xxx	initial node distance in miles (to the nearest 0.01 miles).
attr	any string of up to 30 characters. If **$mp** is imbedded in the string, the mile post of the beginning of each segment attributed will replace the string.

NOTE: Special variable sequences allowed:

$record

This special attribute variable is converted to the object record number when the attribute is assigned. For example: if the attribute string entered by the user is **'abc$record'**; and is assigned using the **atrbt_dig**, **atrbt_rec**, **atrbt_serial** or **atrbt_set** to object **123**, then the attribute assigned is **'abc123**'.

$serial

This special attribute variable is converted to the object serial number when the attribute is assigned. For example: if the attribute string entered by the user is **'ab$serialzz'** and is assigned using **atrbt_dig**, **atrbt_rec**, **atrbt_serial**, or **atrbt_set** to an object with a serial number of **999**, then the attribute assigned is **'ab999zz'**.

$++

This special attribute variable is replaced by a 5 digit integer which can be initialized using the **plus_plus** command. Each time an attribute is stored using the **$++** variable substring, the internal value is incremented by 1.

Example(s):

To attribute all segments between the node at **45 120** and the node at **366 711** with the attribute **'cold creek**' as the second attribute, enter:

atrbt_line 2 cold_creek
start 45 120 (enter with the mouse or digitizer)
continue 366 711 (enter with the mouse or digitizer)

To attribute all segments (on a transportation layer) between the node at **532 1029** and the node at **875 262** with the road number **2642011**- followed by the mile point of the segment beginning, where the first segment begins at mile point **12.62**, (as the first attribute), enter:

atrbt_line 1 mp=12.62 2642011-$mp
start 532 1029
continue 875 262

See also:

atrbt_dig, plus_plus

The **atrbt_list** command sets the graphic input mode to assign attributes from the current, active graphics input device (**GIN**), and gets attribute strings from the specified file.

atrbt_list

Sets graphic input mode for attribute assignment from the current **GIN**, with attribute strings to come from the specified file.

attrbt_list otype action fname

Sets the graphics input mode to allow assignment of attributes to objects using the current active graphics input device (**GIN**). The attribute strings come from the specified file. The file may be a dictionary file or other *ascii* file. If a full path is not specified, then the dictionaries directory will be assumed. From the file, all lines with a non-blank initial character will be truncated at 30 characters and used as attributes. All other lines will be ignored. To begin with, the first line (attribute) is displayed. It may be assigned to any number of objects by using the **continue** command. To use subsequent lines as attributes, use the **start** command. When **start** is used, no attribute will be assigned, but the next line of the file will be read and displayed on the command line area and used as the attribute for subsequent **continue** commands. When the end of the list is reached, a message is printed and the list repeated. Each attriubte is applied to the **otype** object(s) as defined by the action code. The **increment** and **decrement** commands will behave like the **continue** command.

If the **$++** attribute variable is imbedded within the attribute string, it will be replaced with a numeric string as specified by the **plus_plus** command. Each time an attribute containing the **$++** string is assigned, the value is incremented. The initial value can be reset any time by using the **plus_plus** command.

Options/arguments:

otype	Object type code, where: **0** = node **1** = segment **2** = polygon (def=same as map_type value in header)
action	Action code, where: **0** = Replace all attributes w/single new one. **1** = Replace/set 1st attribute, leave others. **2** = Replace/set 2nd attribute, leave others. **3** = Insert new 1st attribute, push others up by one. **4** = Prefix 1st existing attribute. **5** = Suffix 1st existing attribute. **6** = Add a new (additional) last attribute. **7** = Remove ALL attributes for one object. **8** = Prefix 2nd existing attribute. **9** = Suffix 2nd existing attribute.
fname	File Name of list of attributes (may be a dictionary)

Note that the following special variable sequences are allowed:

$record

This special attribute variable is converted to the object record number when the attribute is assigned. For example: if the attribute string entered by the user is **'abc$record'** and is assigned using **atrbt_list**, **atrbt_rec**,

atrbt_serial, or **atrbt_set** to object **123**, then the attribute assigned is **'abc123'**.

$serial

This special attribute variable is converted to the object serial number when the attribute is assigned. For example: if the attribute string entered by the user is **'ab$serialzz'** and is assigned using **atrbt_list**, **atrbt_rec**, **atrbt_serial,** or **atrbt_set** to an object with a serial number of **999**, then the attribute assigned is **'ab999zz'**.

$++

This special attribute variable is replace by a 5 digit integer which can be initialized using the **plus_plus** command. Each time an attribute is stored using the **$++** variable substring, the internal value is incremented by 1.

Example(s):

To assign the first line of the dictionary **dict1** as the **1st** attribute to the polygon at location **255, 672**, enter:

atrbt_list 2 1 dict1
continue 255 672

To assign the first line of the dictionary tmbr_codes as the **2nd** attribute of the polygon at location **255, 672** and the second line of the file to the polygons at **100,300** and **789,1450** enter:

atrbt_list 2 2 tmbr_codes
continue 255, 672
start
continue 100 300
continue 789 1450

See also:

atrbt_rec, atrbt_serial, atrbt_set, atrbt_copy, atrbt_dig

The attributes for a specified set of the current, active map may be displayed on the screen, routed to the primary system printer, or written to a designated *ascii* file with the **atrbt_prin**t command. The printer device and editor type parameters are identified in the **config** file.

atrbt_print

Prints named attribute set **s** to specified device **dev** or file **fname** when **dev = 2**.

atrbt_print s dev [fname]

Prints attribute set specified by set **s** on the device type **dev** or to file name **fname** when **dev = 2**.

Options/arguments:

s	Set to be printed/displayed, where: **0 - 31**, inclusive = valid set number
dev	Device type, where: **0** = screen **1** = printer **2** = write to ascii file (requires **fname**)
fname	*ascii* output file (required when **dev = 2**)

Example(s):

To display the attributes of set **13** of the current, active map on the screen, enter:

atrbt_print 13 0

To print the attributes of all objects of the current, active map to a printer device, enter:

atrbt_print 0 1

See also:

atrbt_edit

The user may invoke the **atrbt_rec** command to cause a specified action to occur to a designated attribute of an object on the current, active map, contained in the specified record number.

atrbt_rec

Performs action on attribute by record number.

atrbt_rec rec action attr

Causes the specified action with attribute string **attr** to occur to the object in record number **rec**.

If the **$++** attribute variable is imbedded within the attribute string, it will be replaced with a 5 character string (**xxxxx**) which is a whole number >= 0. Each time an attribute containing the **$++** sting is assigned, the value is incremented. The initial value can be reset any time by using the **plus_plus** command.

Options/arguments:

rec	Record number: a sequentially-assigned, dynamic value given to an object during the assembly process.
action	Action code, where: **0** = Replace all attributes w/single new one. **1** = Replace/set 1st attribute, leave others. **2** = Replace/set 2nd attribute, leave others. **3** = Insert new 1st attribute, increment others by one. **4** = Prefix 1st existing attribute. **5** = Suffix 1st existing attribute. **6** = Add a new (additional) last attribute. **7** = Remove ALL attributes for one object. **8** = Prefix 2nd existing attribute. **9** = Suffix 2nd existing attribute.
attr	Attribute string (maximum 30 characters long)

Note that the following special variable sequences are allowed:

$record

This special attribute variable is converted to the object record number when the attribute is assigned. For example: if the attribute string entered by the user is **'abc$record'** and is assigned using **atrbt_dig, atrbt_rec, atrbt_serial,** or **atrbt_set** to object **123**, then the attribute assigned is **'abc123'**.

$serial

This special attribute variable is converted to the object serial number when the attribute is assigned. For example: if the attribute string entered by the user is **'ab$serialzz'** and is assigned using **atrbt_dig, atrbt_rec, atrbt_serial,** or **atrbt_set** to an object with a serial number of **999**, then the attribute assigned is **'ab999zz'**.

$++

This special attribute variable is replaced by a 5 digit integer which can be initialized using the **plus_plus** command. Each time an attribute is stored using the **$++** variable substring, the internal value is incremented by 1.

Example(s):

To assign the attribute **XYZZY** to a polygon at record number **87** on the current, active map, enter:

atrbt_rec 87 3 XYZZY

To suffix the existing attribute of the segment at record number **31** on the current, active map with the characters **GY7**, enter:

atrbt_rec 31 5 GY7

See also:

atrbt_dig, atrbt_serial, atrbt_set, plus_plus

The user may invoke the **atrbt_serial** command to cause a specified action to occur to a designated attribute of an object on the current, active map identified by the specified serial number.

atrbt_serial

Performs action on attribute by serial number.

atrbt_serial ser action attr

Causes the specified action with attribute string **attr** to occur to the object identified by serial number **ser.**

If the **$++** attribute variable is imbedded within the attribute string, it will be replaced with a 5 character string (**xxxxx**) which is a whole number >= 0. Each time an attribute containing the **$++** string is assigned, the value is incremented. The initial value can be reset any time by using the **plus_plus** command.

Options/arguments:

ser	Serial number: a sequentially assigned, unique value given to an object during the assembly process. An object's serial number is *never changed* and is *not re-used* if the object is deleted.
action	Action code, where: **0** = Replace all attributes w/single new one. **1** = Replace/set 1st attribute, leave others. **2** = Replace/set 2nd attribute, leave others. **3** = Insert new 1st attribute, push others up by one. **4** = Prefix 1st existing attribute. **5** = Suffix 1st existing attribute. **6** = Add a new (additional) last attribute. **7** = Remove ALL attributes for one object. **8** = Prefix 2nd existing attribute. **9** = Suffix 2nd existing attribute.
attr	Attribute string (maximum 30 characters long)

Note that the following special variable sequences are allowed:

$record

This special attribute variable is converted to the object record number when the attribute is assigned. For example: if the attribute string entered by the user is **'abc$record'** and is assigned using **atrbt_dig, atrbt_rec, atrbt_serial,** or **atrbt_set** to object **123**, then the attribute assigned is **'abc123'**.

$serial

This special attribute variable is converted to the object serial number when the attribute is assigned. For example: if the attribute string entered by the user is **'ab$serialzz'** and is assigned using **atrbt_dig, atrbt_rec, atrbt_serial,** or **atrbt_set** to an object with a serial number of **999**, then the attribute assigned is **'ab999zz'**.

$++

This special attribute variable is replaced by a 5 digit integer which can be initialized using the **plus_plus** command. Each time an attribute is stored using the **$++** variable substring, the internal value is incremented by 1.

Example(s):

To assign the attribute **XYZZY** to a polygon with serial number **87** on the current, active map, enter:

atrbt_serial 87 3 XYZZY

To suffix the existing attribute of the segment with serial number **31** on the current, active map with the characters **GY7**, enter:

atrbt_serial 31 5 GY7

See also:

atrbt_dig, atrbt_record, atrbt_set, plus_plus

The user may invoke the **atrbt_set** command to cause a specified action to occur with a designated string for all of the attributes of objects in the specified set on the current, active map.

atrbt_set

Performs action on all attributes in named set.

atrbt_set s action attr

Causes the specified action with attribute string attr to occur to all objects in set **s**.

If **$++** is imbedded within the attribute string, it will be replaced with a 5 character string (**xxxxx**) which is a whole number >= 0. Each time an attribute containing the **$++** string is assigned, the value is incremented. The initial value can be reset any time by using the **plus_plus** command.

Options/arguments:

s	Set number, where: **0 - 31**, inclusive, are valid set numbers
action	Action code, where: **0** = Replace all attributes w/single new one. **1** = Replace/set 1st attribute, leave others. **2** = Replace/set 2nd attribute, leave others. **3** = Insert new 1st attribute, increment others by one. **4** = Prefix 1st existing attribute. **5** = Suffix 1st existing attribute. **6** = Add a new (additional) last attribute. **7** = Remove ALL attributes for one object. **8** = Prefix 2nd existing attribute. **9** = Suffix 2nd existing attribute.
attr	Attribute string (maximum 30 characters long)

Note that the following special variable sequences are allowed:

$record

This special attribute variable is converted to the object record number when the attribute is assigned. For example: if the attribute string entered by the user is **'abc$record'** and is assigned using **atrbt_dig, atrbt_rec, atrbt_serial,** or **atrbt_set** to object **123**, then the attribute assigned is **'abc123'**.

$serial

This special attribute variable is converted to the object serial number when the attribute is assigned. For example: if the attribute string entered by the user is **'ab$serialzz'** and is assigned using **atrbt_dig, atrbt_rec, atrbt_serial,** or **atrbt_set** to an object with a serial number of **999**, then the attribute assigned is **'ab999zz'**.

$++

This special attribute variable is replaced by a 5 digit integer which can be initialized using the **plus_plus** command. Each time an attribute is stored using the **$++** variable substring, the internal value is incremented by 1.

Example(s):

To assign the attribute **XYZZY** to the objects found in set **12** of the current, active map, enter:

atrbt_set 12 3 XYZZY

To suffix the existing attributes of the segments found in set **7** of the current, active map with the characters **GY7**, enter:

atrbt_set 7 5 GY7

See also:

atrbt_dig, atrbt_record, atrbt_serial, plus_plus

The **atrbt_thin** command initiates a reduction type of action on the attributes of all of the objects included in a specified set of the current, active map.

atrbt_thin

Performs thinning on objects in named set.

atrbt_thin action s

Initiates specified action type against attributes defined by set number **s** for current, active map.

Options/arguments:

action	Action performed against objects, where: **-2** = remove all redundant attributes **-1** = remove all attributes **0** = save last attribute **1** = save first attribute . . . . . . **n** = save nth attribute
s	Set number, where: **0 - 31**, inclusive, are valid set numbers

---

Example(s):

To remove all of the attributes for set **6** of the current, active map, enter:

atrbt_thin -1 6

To save the **14**th attribute for all objects of the current, active map, enter:

atrbt_thin 14 0

See also:

Chapter 10 Map Manipulation

It is often necessary to clean raw data prior to importing it into the LTPlus environment. The **b_edit** command is capable of performing user-designated preprocessing operations on a specified *ascii* formatted data file.

b_edit

Pre-processes named file prior to importing.

b_edit ifl [ofl] [-c] [-e] [a=n] [b=n] [c=n] [d=n] [f=n] [g=n] [i=char] [j=n] [l=n] [s=n] [t=n] [xx=n]

Activates a preprocessor against the named input file **ifl** to prepare it for importing into the LTPlus environment. The preprocessor filtering actions are defined by the item=val/str argument(s) passed to this function. The output from this command will be stored in the named input file (**ifl**) unless an output file (**ofl**) is designated. The **-e** and **-c** options delete extension and control characters, respectively. The item=arg parameters are additional filtering actions, described below.

Options/arguments:

ifl	Input file, *ascii* format
ofl	Name of *ascii* output file to which filtered results will be written. If ofl is not specified, the results will *overwrite* the input file (ifl)
-c	delete all control characters other than linefeeds
-e	delete all extension characters

item=arg:

a =	add a linefeed after specified number of characters.
b =	number of lines to be deleted from the beginning of the file
c =	number of characters to be deleted from beginning of each line.
d =	delete specified column from each line.
f =	fill all lines to specified number of characters with blanks.
g =	combine specified number of input lines into single output lines
i =	decimal equivalent of *ascii* character to be deleted from file (all occurrences)
j =	insert a new (blank) column at specified col. number.
l =	minimum length of line to be retained
s =	fill blanks with character which precedes it by specified number of characters
t =	truncate all lines at specified number of characters if they are longer than that.
xx =	Replace ASCII (decimal) value xx with ASCII (decial) value n. e.g, to replace all semi-colons (*ascii* decimal value 59) with linefeeds (*ascii* decimal value 10) use a specification: 59=10

NOTE: Only one of each parameter may be specified for all parameters except: **d,i,j,xx**

NOTE: ANY combination of parameters is legal.

NOTE: Line length does NOT include the linefeed character.

NOTE: Columns are numbered with the beginning column being number 1.

NOTE: The order in which these parameter take effect can be important. Here is the process in which the file is read, edited, and written:

1. One line is read from the input file. Control (**-c**) & extention (**-e**) characters are deleted if specified. If a linefeed is not encountered first and there is a specified value (**a=**), a linefeed is inserted at the specified number of characters (not counting the characters deleted by **-c** and **-e**). **b_edit** will not do line length processing of lines longer than 2000 characters, although other processing may be done.

2. If there is a specified number of lines to delete from the top of the file (**b=**), then the line number of the read line is checked and the line is omitted from output if it is within the specified count. If so, return to step 1 and read next line.

3. If a minimum line length is specified (**l=**), then the length of the line is checked. If it is less than the specified length, then it is disregarded. If so, return to step1and read next line.

4. If there is a specification to delete a number of characters from the beginning of each line (**c=**), then that number of characters is deleted from the line. If there are fewer characters than specified under **c=**, then all characters of the line are deleted, but processing continues.

5. The line is scanned for occurrences of specified *ascii* characters to be deleted **(i=**), if specified. They are deleted if present.

6. If column delete(s) are specified (**d=**) then those columns are deleted (if they exist) beginning with the column furthest to the RIGHT.

7. If column insertion(s) are specified (**j=**) then those columns are added beginning with the column furthest to the RIGHT. If the string does not have enough columns, then it is filled with blanks up to the maximum column specified under **j=** , THEN expanded beginning from the RIGHT.

8. If fill to a given length is specified (**f=**) then the line is filled to that length if it is shorter than specified. If longer, then it is left alone.

9. If blanks are to be replaced by a preceding character (**s=**) then they are replaced by the specified preceding character. If the specified preceding character is on the previous line, then it is used only if the previous line is to be joined (**g=**) with this line. If there is no preceding line or if there are insufficient characters on the preceding line to go back **s** characters, then the blank(s) are left.

10. If line truncation is specified (**t=**) then the line is truncated at that length if it is longer. Shorter lines are left alone.

11. The grouping (**g**) parameter is checked and a linefeed character is added to the end of the line if no grouping is specified, or if the specified number of lines has been written according to the grouping parameter.

12. All character replacement (**xx=**) is done. Any number of characters (control, displayable, or extention) may be replaced for any other *ascii* character in the range of 0 to 255.

13. The line is written to the output file.

14. Go to step 1.

Example(s):

To remove all extension and control characters, **34** lines from the beginning of the file, **4** characters from the beginning of each line and all occurrences of the character **G** from the *ascii* file **cub_mtn** and store the results in an *ascii* file called **cub_peak**, enter:

b_edit cub_mtn cub_peak -c -e b=34 c=4 i=71

To remove all control characters and retain only lines with a minimum line length of **40** from a file called **cub_creek** and store the results in that same file, enter:

b_edit cub_creek -c l=40

To replace all occurrences of the semi-colon (ASCII 59) with linefeeds (ASCII 10) enter:

b_edit cub_creek 59=10

Because every system has a finite amount of hard disk space, it is important to be able to archive map data when it is no longer needed on the system. The **backup** command provides a means for moving all of the files associated with a map name to a user-designated offline storage medium. This command looks for the storage location identified in the **config** file. It is important that both the source (named or current map) and destination **(bkup_path**) path names be correct (see **config**). The actual **backup** command is created from three fields in the **config** file: **bkup_path**, **bkup#1** and **bkup#2**, where **#** is either the number **1** or **2**. These fields are wrapped around the current map name. Thus, you may backup using any of the UNIX/XENIX system commands that you want to use including **backup**, **tar**, or **cpi**o. The **bkup_path** field is used when no parameters are used. It will precede the name of the current map when making the system call. Usage of the **backup** command is as follows:

backup

Initiates **backup** of specified map data.

backup [num] [name]

Copies the files for the named map name (default is the current map) using the command prefix defined by the value of **num**. It may have a value of **1** or **2**, and refers to the **bkup#1** and **bkup#2** parameters in the **config** file (note that **num** is substituted for the **#** sign). By default, the current, active map is archived to the device identified by **bkup_path** using a **tar** format. The **bkup_path** element in the **config** file names the device to which the backup is directed. Common device names are:

	TAPE	FLOPPY	
For UNIX:	**/dev/rmt/c0s0**	**/dev/rdsk/f0**	(or **f1**)
For XENIX:	**/dev/rct0**	**/dev/rfd0**	(or **rfd1**)

Options/arguments:

num	Number of backup prefix and suffix to use for creating backup command. Valid values are 1 or 2.
name	Map name. Default is the current map.

Example(s):

To make a copy of the data files associated with the map called **cub** using **bkup11** and **bkup12** from the **config** file, (**bkup11 = ls** and **bkup12** is **| cpio -ocBv -O /dev/rdsk/f0q15d**), enter:

backup 1 cub

The result is a system command:

ls /u[sr]/ltp/maps/cub | cpio -ocBv -O /dev/rdsk/f0q15d

To make a backup copy of the data files associated with the current, active map (which, by default, is at the **maps_path** directory) to the **bkup_path** which is set to **/dev/rdsk/f0**, enter:

backup

See also:

catalog, config, retrieve, save

This is a mode-setting command intended for use as a raster editing tool. It's primary purpose is to allow the user to close the gap between two spur endpoints with a straight line of pixels.

bridge_gap

Sets the mode to **bridge_gap** to close line breaks.

bridge_gap

Sets the graphic input mode to **bridge_gap**. When in this mode, a **start** or **continue** command will cause the program to search a square area (up to the **header** file **search_radius** value in pixels) for the closest two spur endpoints. If two endpoints are found, then they are connected with a straight line. This command is used for closing line breaks on thinned raster data. Assembly is not required. Note that the current mode/action is displayed in the upper-left corner of the screen. NOTE: the **grow_spurs** command uses the same logic, but automatically applies it to all raster spurs on the map.

Options/arguments:

None.

Example(s):

To connect a line break near pixel coordinates **x=125** and **y=482**, enter:

bridge_gap start 125 482
(or **continue 125 482**)

See also:

continue, header, start, grow_spurs

The **brstr_create** command copies the temp directory raster file to a file called **rasterb**. This can be used to save a second copy of the raster for later merging or merging selectively with the displayed raster. See the **brstr_delete** command. The **rasterb** file is carried with the map until the map is purged or the **brstr_delete** command is executed.

NOTE: all normal editing and viewing done with a map is with the raster file; the **rasterb** file is set aside until later. It can only be accessed by using the **displrstr_get** or recall commands.

brstr_create

Stores the temp raster over the temp **rasterb** file.

brstr_create

Stores the temp raster over the temp **rasterb** file.

Options/arguments:

None.

Example(s):

To create/overwrite the temp **rasterb** with a copy of the temp raster file, enter:

brstr_create

See also:

brstr_delete, displrstr_ers, displrstr_get, recall

This command deletes the temp directory **rasterb** file, if it exists. This file is created with **brstr_create** command.

NOTE: all normal editing and viewing done with a map is with the raster file; the **rasterb** file is set aside until later. It can only be accessed by using the **disprstr_get** or **recall** commands.

brstr_delete

Deletes the **rasterb** file in the **temp** map directory.

brstr_delete

Options/arguments:

None.

Example(s):

To delete the temp **rasterb** file, enter:

brstr_delete

See also:

brstr_create, recall

This command has been deleted in version 2.12 & later versions.

Use the **disprstr_ge**t or **recall** commands to access the **braster**.

See also:

brstr_create, brstr_delete, disprstr_ers, disprstr_mrg, disprstr_get, recall

---

This command displays the named **cursor_keys** functions.

ckeys_disp

Displays the named cursor key functions.

ckeys_disp [-p] cname

Displays the cursor key functions stored in the named **cname** file which is assumed to be in the **/u[sr]/ltp/cursor_keys** directory unless a full path name is specified. A **-p** option will print the cursor functions to the hardcopy printer device associated with the system in the **config** file.

Options/arguments:

-p	Route output to the system printer
cname	Cursor function file name

Example(s):

To display the cursor functions stored in the file called **cub_keys**, enter:

ckeys_disp cub_keys

See also:

ckeys_get, ckeys_make, ckeys_save, stickup

This command loads the named **cursor_keys** function file.

ckeys_get

Loads the named cursor key function file.

ckeys_get cname

Loads the cursor key functions stored in the named **cname** file which is assumed to be in the **/u[sr]/ltp/cursor_keys** directory unless a full path name is specified. These are the functions which are activated when the associated cursor keys are pressed.

Options/arguments:

cname Cursor function file name

Example(s):

To load the cursor functions stored in the file called **cub_function**, enter:

ckeys_get cub_function

See also:

ckeys_disp, ckeys_make, ckeys_save, stickup

The sole purpose of the **clear_mod**e command is to deactivate any current, active mode.

clear_mode

Clears any existing graphics input mode.

clear_mode

Clears any existing input mode; equivalent to start-up state.

Options/arguments:

None.

Example(s):

To exit from the current input mode without activating a new input mode, enter:

clear_mode

To exit from the current input mode and activate a different mode, simply enter the new mode command.

See also:

All Input Mode commands (*Chapter 9*)

The **config** file contains many of the LTPlus operating parameters. It may be modified by anyone via the **config** command (or by editing the file with the **edit** command or with any system *ascii* editor). A set of standard default parameters in a **config** file will be packaged with the LTPlus software. The system manager will have to review and modify any of the **config** file parameters which could prevent start up of LTPlus. Once the software can be implemented properly, any user may customize the LTPlus processing environment by executing a **config** command, described as follows:

config

Modifies default configuration parameters.

config item=string

Modifies default configuration settings. Only one equivalency (**item=string**) may accompany a **config** command entry.

Options/arguments:

item =	description and available options
prtr_adr	(not used)
prtr_type	(not used)
crt_adr	(not used)
crt_type	(not used)
bkup11	Initial portion of system backup command which will precede the file name. Suggested UNIX string: **tar cvf /dev/rdsk/f0** Suggested XENIX string: **tar cvf /dev/rfd0**
bkup12	Final portion of system backup. Suggested UNIX string: Suggested XENIX string:
bkup21	Initial portion of system **backup** command which will precede the file name. Suggested UNIX string: **ls** Suggested XENIX string: **ls**
bkup22	Final portion of system **backup** Suggested UNIX string: **| cpio -ocBv -O /dev/rdsk/f0** Suggested XENIX string: **| cpio -ocBv -O /dev/rfd0**
print1	Initial portion of the system **print** command to be used which precedes the file name. Suggested UNIX string: **pr -f** Suggested XENIX string: **lp**
print2	Final portion of the UNIX **print** command to be used following the file name. Suggested UNIX string: **| lp** Suggested XENIX string:
edit1	**vi**
edit2	
cursor_file	Cursor keys file (**def=cursor_keys**) file must be found in: **/u[sr]/ltp/cursor_keys**
mouse_adr	Port number (**/dev/...**)
mouse_type	Mouse driver file name (i.e., **al40f82**, **cc9000f2**). Must be found in: **/u[sr]/ltp/digitizers**
dig1_adr	Port number (**/dev/...**)
dig1_type	Digitizer driver file name (i.e., **al40f82**, **cc9000f2**). Must be found in: **/u[sr]/ltp/digitizers**

dig2_adr	Port number (**/dev/**...)
dig2_type	Digitizer driver file name (i.e., **al40f82**, **cc9000f2**). Must be found in: **/u[sr]/ltp/digitizers**
plt1_adr	Port number (**/dev/**...)
plt1_type	Valid values are **'center'** or **'lowerleft'** Center denotes plotters whose origin is in the center of the paper. Any other term implies a plotter whose origin is the lower left. Center is the default if lower left is not specified (i.e., any entry, including null, that does not start with the character L will default to center).
plt2_adr	Port number (**/dev/**...)
plt2_type	Valid values are **'center'** or **'lowerleft'**
_____	
maps_path	Location where maps are stored. Default is **/u[sr]/ltp/maps**
bkup_path	Device for archiving action. Default is **/dev/rdsk/f0** (UNIX) or **/dev/rfd0** (XENIX)
data3_path	(not used)
data4_path	(not used)
export_path	Default directory to hold exported *ascii* map files. Default is: **/u[sr]/ltp/export**
import_path	Default directory to hold importable *ascii* map files. Default is: /u**[sr]/ltp/import**
malloc	Memory allocation. Recommended value is system memory less 3000000 bytes.
_____	
menuset	Startup menuset name. Valid name must located in **/u[sr]/ltp/menus** directory.
ginput	Startup graphics input device name. Valid options are: **mouse, dig1, dig2**
rasterh	Raster height in pixels (**def=1024**)
rasterw	Raster width in pixels (**def=1024**) _____
pen0	Map area bkgd (**def = bright_black**)
pen1	Cursor (**def = fishbelly_white**)
pen2	Raster (**def = polar_blue**)
pen3	Vector (**def = vampire_red**) [aka vector, in graphics command]
pen4	Text (**def = glacier_cyan**)
pen5	Rstr/txt hi-lt (**def = dark_white**)
pen6	Vctr/txt hi-lt (**def = lides_green**) [aka highlight]
pen7	Rstr/vctr hi-lt (**def = lumpy_brown**) [aka normal over raster]
pen8	Rstr/vctr/txt hlt (**def=cadaver_blue**) [aka highlight over raster]

_____

Color choices are as follows:

sickly_magenta	**ripe_magenta**
fishbelly_white	**dark_white**
cadaver_blue	**polar_blue**
glacier_cyan	**caribbean_cyan**
lumpy_brown	**infectious_yellow**
lides_green	**canpea_green**
bright_black	**seattle_grey**
vampire_red	**redtape_red** (a color you may never change)

A color may also be set with a decimal value (0-63) representing a mixture of red, green, and blue, as described in *Chapter 3* (**config**).

Example(s):

To change the primary digitizer device file to be **al40f82**, enter:

config dig1_type=al40f82.dgt (assumes file is in **/u[sr]/ltp/digitizers** directory)

See also:

header

This command invokes an input mode which allows the user to connect map positions by clearing a path through unwanted raster data and drawing a line.

connect

Activates mode for connecting points.

connect [-s]

Activates the **connect** mode. This mode allows the user to connect positions by punching through unwanted raster garbage to draw a line. It is different from **draw line** in that it not only draws a line between points, but also erases on both sides of the line. This allows the user to draw through garbage on the raster without first erasing it. The garbage can often be eliminated more efficiently through a *macro* or other non-interactive command. While in **connect** mode, sequential x,y locations are entered using the **continue** command. To start a new set of connected points while in the **connect** mode, execute the **start** command, followed by sequential **continue** command(s). The **-s** option will cause each digitized coordinate position to be modified (snapped) to the closest existing point (on pixel), within a predefined tolerance (**search_radius** in pixels; **header** file element). If no existing point can be located within that radius, the position will be recorded as entered. Note that both the **increment** and **decrement** command will behave like the **continue** command.

Options/arguments:

-s Snap mode activated

Example(s):

To activate the **connect** mode, with the snapping function on, enter:

connect -s

To record subsequent positions as connected, enter:

continue x y

To start a new set of connected points, enter:

start x y

To exit this mode, enter:

clear_mode

or enter a new mode command.

See also:

clear_mode, continue, header, start

The **continue** command is generally executed with a corresponding x,y position while an input mode is active.

continue

Continues the current mode/action.

continue x y

Continue with the current mode/action at coordinates **x,y**. If no mode is active when this function is executed, no action will occur. Note that the current mode/action is displayed in the upper-left screen tile.

Options/arguments:

x	X coordinate value (easting)
y	Y coordinate value (northing)

Note: normally the **start**, **continue**, **increment**, and **decrement** commands are entered as button presses on the mouse or digitizer cursor. In these cases, the input coordinates will be automatically supplied. If the command is entered from the keyboard without coordinate arguments, the user will be prompted to enter the x,y positions.

Example(s):

If the **draw** mode has been activated, enter subsequent points with:

continue x y

See also:

connect, decrement, draw, erase, erase_seg, increment, query, separate, start, vertex_move, wink

---

LTPlus, by design, conducts its basic operations on any given map in pixel coordinates. However, it provides the user with the opportunity to perceive movement within that same map in as many as 17 different coordinate unit types, as listed below. The current coordinate unit type will be identified in screen tile 2 (upper-left window). Not that the user may define special coordinate types. User-defined coordinate systems may be established and stored in the **config** file (see **config**).

coord_type

Defines the cuurent units of display and input.

coord_type c

Defines the current, active units of display and input.

Options/arguments:

c Units of measurement, where:
0 = Raster pixels
1 = W. Longitude, N. Latitude (ddd/mm/ss.ss)
2 = UTM meters
3 = State Plane feet
4 = Digitizer1, absolute coordinates
5 = Digitizer2, absolute coordinates
6 = Map inches
10-19 = User-defined units

Example(s):

To establish display and input units as ground feet in the State Plane Coordinate System, enter:

coord_type 3

To set raster pixels as the units for display and input, enter:

coord_type 0

See also:

register, config

Every LTPlus map-based operation is dependent upon the existence of a current, active map. A new map is identified for use by first being created, using, as would seem proper, the **create** command. When a new map is created, its **header** file data is initially set to the contents of the default **header** file in the **/u[sr]/ltp/config** directory. This allows the user to save time by setting up a default **header** file which is used as a basis for one or more maps within a project.

create

Creates and opens the named file for use.

create mname

Creates and opens the specified map name (**mname**). The name is assumed to be new. If it already exists in the **maps_path** directory (in the **config** file), the user wll be notified about the duplicate name condition and prompted for a different name.

Options/arguments:

mname Map name

Example(s):

To create a new map called **cub_falls**, enter:

create cub_falls

See also:

acquire, catalog

The **crit_adj** command establishes as a set selection criteria the adjacency between all polygon objects in the current map and those in an identified set of objects of the same current map.

crit_adj

Criteria: adjacency to polygons in named set.

crit_adj s

Establishes that during the next set selection, only those polygons which are adjacent to the polygons in the named set **s** will be selected. The named set s must contain polygons for this function to have any meaning. Assembly is required. Up to 10 **crit_adj** criteria commands may be entered. When more than one is entered in this category, only one of them must be true for an object to be included in the target set.

Options/arguments:

s Set number, where:
0 - 31, inclusive, are valid entries

Example(s);

To establish that objects will only be selected (at the set selection terminus of **crit_exec**) if they are polygons which are adjacent to polygons in set **12**, enter:

crit_adj 12

See also:

crit_clear, crit_exec, crit_report, crit_type

The **crit_arange** command establishes lexical range criteria for evaluating all attributes of all objects.

crit_arange

Criteria: lexical range of attributes.

crit_arange arg1 arg2

Establishes selection criteria based upon a lexical range of attributes for all objects in the map file. That is, objects with attributes which fall within the alphabetic (*ascii*) range defined by **arg1** and **arg2** will be selected when this criteria function is activated with a **crit_exec** command. The special, wildcard characters described in **crit_astrg** apply to the operation of this command as well. Up to 10 occurrences of this command are allowed. When multiple **crit_arange** entries exist, only one must be true for an object to be included in the target set.

Options/arguments:

arg1 Lower bound attribute
arg2 Upper bound attribute

Example(s):

To select objects with attributes which begin with the characters **A** through **D** only, enter:

crit_arange A D*

To select objects with attributes which end with digits **12** through **153** only, enter:

crit_arange *12 *153

To select objects with attributes which contain the codes **0601** through **0620** in columns **3** through **6** only, enter:

crit_arange ??0601 ??0620zz

NOTE: the meta characters **?** and ***** are allowed in these parameters to represent as single character **(?)** and any number (>=0) of characters (*). The * meta character may be used only once per parameter string. To test for the real (non-meta) **?,** * and characters, use **\?,** *, and **** in the parameter string. For example, to test for strings whose 3rd character is in the range * through **?,** enter:

crit_arange ?? ??\?***

See also:

crit_astrg, crit_clear, crit_exec, crit_slen

The **crit_area** command sets criteria for evaluating the areas of polygon objects.

crit_area

Criteria: area range for polygon objects.

crit_area area b1 b2

Establishes criteria for set selection based upon the area of the objects (polygons) in the map file. Up to 10 occurrences of this command are allowed. When multiple **crit_area** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

area	Area units code, where: **0** = pixels **1** = square inches (map) **2** = acres **3** = square miles **4** = square cm (map) **5** = square meters **6** = square kilometers
b1	Lower bound of inclusive range
b2	Upper bound of inclusive range

Example(s):

To set three ranges of area, including <=1 map inch, between 50 and 100 square meters and >=400 acres, enter:

crit_area 1 0 1
crit_area 5 50 100
crit_area 2 400 99999999

See also:

crit_clear, crit_exec, crit_length, crit_slen

Yet another command for evaluating object attributes, the **crit_astrg** command sets criteria with regard to matching attribute characters or character sequences with user-specified data (template).

crit_astrg

Criteria: match of input template string to attribute.

crit_astrg string

Establishes criteria for set selection by attribute string matching. The input string must match the attribute string exactly except where special, wildcard (meta) characters are included in the string. Up to 10 occurrences of this command are allowed. When multiple **crit_astrg** entries exist, only one must be true for an object to be included in the target set.

Options/arguments:

string String of characters, where:
? = will match any 1 character in attribute
* = will match any group of characters in attribute
\\ = will match \ in attribute
\? = will match ? in attribute
* = will match * in attribute

NOTE: only one * meta-character may be used in a string

Example(s):

An entry of:

crit_astrg a*2

will match the strings **abcdhegar%#&q&2** and **a2** but will not match **2a** .
An entry of:

crit_astrg a?2

will match the strings **a?2**, **ab2** and **a22** , but will not match the string **a2**.
To match an attribute string of \clear, enter:

crit_astrg \\clear

See also:
crit_clear, crit_exec, crit_arange, crit_slen

Criteria development is accumulative in nature. Each criteria set is added to the previous sets until all criteria are cleared with a **crit_clear** command.

crit_clear

Clears all existing set selection criteria.

crit_clear

Clears all existing set selection criteria

Options/arguments:

None.

Example(s):

To clear all previously established set selection criteria, enter:

crit_clear

See also:

All **crit_** functions

The **crit_exe**c command applies all of the current criteria to the current, active map and stores the results in a named set.

crit_exec

Starts object selection based on pre-established criteria statements and identifies selected objects in a target set.

crit_exec s [descr]

Applies pre-established criteria to active map and stores selected objects in set number **s**. The set may be described with an optional **descr** string entry. If set number **s** already exists, its contents will be overwritten.

The set created will be highlighted automatically, and the number of objects, total length/perimeter in miles, and total area in acres will be displayed on the command line area.

Options/arguments:

s	Set number, where: **1 - 24** = desired set number
descr	Set descriptor (optional)

Example(s):

To activate a sequence of previously entered **crit_ selection** commands and store in set number 10, enter:

crit_exec 10 all streams which exceed 3 mi in len

To activate a sequence of previously entered **crit_ selection** commands and store in set number 8, with no descriptor, enter:

crit_exec 8

See also:

All **crit_** and **set_** functions

The **crit_length** command sets criteria for evaluating the lengths of all lines and polygon perimeters.

crit_length

Criteria: length range for lines and polygons.

crit_length length b1 b2

Establishes criteria for set selection based upon the length (lines) or perimeter (polygons) of the objects (lines or polygons). Up to 10 occurrences of this command are allowed. When multiple **crit_length** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

u	Length units code, where: **0** = pixels **1** = map inches **2** = ground feet **3** = ground miles **4** = map centimeters **5** = ground meters **6** = ground kilometers
b1	Lower bound of inclusive range
b2	Upper bound of inclusive range

Example(s):

To set three ranges of length, including <=1 map inch, between 50 and 100 meters and >=400 feet, enter:

crit_length 1 0 1
crit_length 5 50 100
crit_length 2 400 99999999

See also:

crit_area, crit_clear, crit_exec, crit_slen

The **crit_newatr** command sets criteria for determining whether or not the attributes assigned to all objects on the map have corresponding definitions in the data dictionary identified in the map's **header** file.

crit_newatr

Criteria: object attributes not in data dictionary.

crit_newatr

Establishes criteria to determine which objects of the current map have attributes that are not defined in the attribute dictionary. The attribute dictionary for each map is defined in its **header** file and, thus, may be changed.

Options/arguments:

None.

Example(s):

To establish as a set selection criteria the lack of a corresponding dictionary entry for each attribute of the current, active map file, enter:

crit_newatr

See also:

crit_exec, crit_clear, dict_edit, dict_print, header

The **crit_psides** command establishes criteria for evaluating the number of sides of polygon objects. A polygon is an area defined by a sequence of vectors, or line segments, which form a perimeter. The straight-line segments which comprise the perimeter of a polygon are considered to be sides of that polygon object.

crit_psides

Criteria: range of number of sides of polygons.

crit_psides s1 s2

Establishes criteria for selection of objects (polygons) with **s1** to **s2** number of sides (inclusive). Up to 10 occurrences of this command are allowed. When multiple **crit_psides** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

s1	Minimum number of sides (<0)
s2	Maximum number of sides (>=s1)

Example(s):

To establish set selection criteria for all polygons with **1** to **8** sides, enter:

crit_psides 1 8

To establish set selection criteria for all polygons with more than **10** sides, enter:

crit_psides 11 99999999

See also:

crit_clear, crit_exec, crit_type

The crit_record command establishes set selection criteria for a range of object record numbers.

crit_record

Criteria: range of object record numbers.

crit_record b1 b2

Establishes criteria for set selection by a range of record numbers. Up to 10 occurrences of this command are allowed. When multiple **crit_record** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

b1 Lower bound of record number range
b2 Upper bound of record number range

Example(s):

To set as selection criteria a range of record numbers from **500** to **800**, inclusive, enter:

crit_record 500 800

See also:

crit_exec, crit_clear, crit_serial

The **crit_report** command generates a report showing all pending set selection criteria created by crit_ commands. The report can be written to the screen, a printer or to a file.

crit_report

Generate a report of current set selection criteria.

crit_report p [fname]

Reports on all established criteria. Report will be output to the device specified by **p**. If **p** is 2**,** indicating that output is to be to an *ascii* file, the filename must be supplied as well (**fname**). If a full pathname is not supplied the file will be located in the **export_path** directory (see **config**).

Options/arguments:

p	output device, where: **0** = screen **1** = printer **2** = file
fname	Name of file to be written (only when **p=2**)

Example(s):

To display all current set selection criteria on the screen, enter:

crit_report 0

To output the report to a file called **kettle**, enter:

crit_report 2 kettle

See also:

All **crit_** functions

The **crit_serial** command establishes set selection criteria for a range of object serial numbers.

crit_serial

Criteria: range of object serial numbers.

crit_serial b1 b2

Establishes criteria for set selection by a range of serial numbers. Up to 10 occurrences of this command are allowed. When multiple **crit_serial** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

b1 Lower bound of serial number range

b2 Upper bound of serial number range

Example(s):

To set as selection criteria a range of serial numbers from **500** to **800**, inclusive, enter:

crit_serial 500 800

See also:

crit_exec, crit_clear, crit_record

The **crit_slen** command establishes set selection criteria for the length of an object's attribute string.

crit_slen

Criteria: range of sttribute string length.

crit_slen n1 n2

Defines the string length range criteria for set selection. Note: by specifying zero (0) for n1 and n2, objects without strings will be selected. When multiple **crit_slen** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

n1 Lower bound of string length range
n2 Upper bound of string length range

Example(s):

To set as selection criteria a string length range of **1** to **18** characters, inclusive, enter:

crit_slen 1 18

See also:

crit_astrg, crit_arange, crit_exec, crit_clear

The **crit_subel** command establishes topologic constraints on objects to be selected. To be selected, an object must be a topologic element of a member of the specified set.

crit_subel

Criteria: topologic members of the specified set.

crit_subel s

This command establishes criteria requiring that an object must be an element of an object in the set specified (**s**). A *segment* is an element of a polygon if it makes up part of the boundary of that polygon. A node is an element of a segment if it is an endpoint of that segment. A *node* is an element of a polygon if it is an element of a segment which is an element of that polygon. Every object is a *subelement* of itself. An exterior (island perimeter) is a subelement of its parent polygon. When multiple **crit_subel** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

s Set number, where:
0 - 31 are valid entries

Example(s):

To establish that objects will only be selected if they are subelements of the objects in set **14**, enter:

crit_subel 14

See also:

crit_exec, crit_clear, crit_superel, crit_type

The command establishes topologic constraints on objects to be selected. To be selected, an object must have as an element a member of the specified set.

crit_superel

Criteria: has an element which is a member of the specified set.

crit_superel s

This command establishes criteria requiring that an object must have as an element an object in the set specified (**s**). A *segment* is an element of a polygon if it makes up part of the boundary of that polygon. A *node* is an element of a segment if it is an endpoint of that segment. A node is an element of a polygon if it is an element of a segment which is an element of that polygon. Every object is a *superelement* of itself. A normal (interior) polygon (or the map border) is a superelement of its parent islands. When multiple **crit_superel** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

s Set number, where:
0 - 31 are valid entries

Example(s):

To establish that objects will only be selected if they have an element which is a member of set **16**, enter:

crit_superel 16

See also:

crit_exec, crit_clear, crit_subel, crit_type

The **crit_type** command establishes criteria based upon an object type.

crit_type

Criteria: object type.

crit_type c

This command establishes criteria for set selection by object type. When multiple **crit_type** entries exist, only one must be true for an object to be included in the target set. Assembly is required.

Options/arguments:

c Object type code, where:
0 = any node
1 = any segment
2 = any polygon
3 = node with 0 branches; an isolated point
4 = node with 1 branch; a spur endpoint
5 = node with 2 branches; a node on a simple island
6 = node with 3 branches; a junction node
7 = node with 4 branches; a junction node
8 = segment with 2 free (spur) ends; a free segment
9 = segment with 1 free (spur) end; a spur
10= segment with no free ends
11= single links to islands (segment)
12= segments between polygons with the same attribute
13= possible segment contour attribute sequence error
14= interior polygon
15= exterior polygon (negative area value)
16= map border exterior polygon (negative area value)

Example(s):

To establish set selection criteria for determining all objects which are single links to islands, enter:

crit_type 11

To establish set selection criteria for determining all spurs, enter:

crit_type 9

See also:

crit_exec, crit_clear, crit_subel, crit_superel, log

Add a point to the contour attribute profile.

decrement

decrement x y

This command adds a point to the contour attribute profile. This function is meaningful only when **atrbt_contur** graphics input mode is active. In all other modes, it behaves exactly like the **continue** command. All vectorized line crossed from the last point on the profile up to the **decrement** point will be assigned a numeric attribute in decrements of the contour interval (stored in the **header** file).

Options/arguments:

x	**X raster** coordinate of point
y	**Y raster** coordinate of point

Example(s):

To assign attributes to contour lines which are incrementally downhill from a previous **atrbt_contur, start, continue** or **increment** command, enter:

decrement x y

See also:

atrbt_contur, continue, header, increment, start

The default **header** file, stored as **/u[sr]/ltp/config/heade**r, contains information used for all newly created maps. Some of the parameters in the **header** file affect the processing environment, while others provide valuable data which can be integrated into a file **header** for some of the export formats. The **default_head** command allows the user to update and/or display the default **header** file.

default_head

Updates/displays the default **header** file.

decrement x y

This command displays and/or modifies the default **header** file. Only one **item = string** pair may accompany each **default_head** command. The **-p** option prints the default **header** file to the default primary printer.

Options/arguments:

-p Output to printer
item string argument, where:
= arg banner = text (maximum 72 characters)
comments = text
compress = yes/no
contour_interval = numeric value (feet)
cursor_file = name of cursor key definition file; if blank, use default set in **config** file
datum = year
dictionary = name of data dictionary (as found under: **/u[sr]/ltp/dictionaries** path)
edgematch = [N] [E] [S] [W]
edge_e = map name of map to the east
edge_n = map name of map to the north
edge_s = map name of map to the south
edge_w = map name of map to the west
head1 = header value (1st 72 char) used in dem output
head2 = header value (2nd 72 char) used in dem output
layer_name = text; layer name
map_ht = map height in minutes (def = 7.5)
map_wd = map width in minutes (def = 7.5)
maptype = type of map (poly, line, contour)
menu_set = name of menuset (as found under: **/u[sr]/ltp/menus**)
org = name of organization which digitized map
quad_unit = quad or unit name
revision_yr = numeric value; revision year
scale = numeric value; scale ratio (i.e. for 1:24000 the entry would be 24000)
search_rad = numeric value; pixel radius for search
source_yr = numeric value; source year
state = numeric value; 2-digit state code
stplzn = numeric value; FIPS state plane coordinate system zone code (different from the USGS state plane zone code for the same zone) (4 digits)
template1 = map name of first template map to snap to
template2 = map name of second template map to snap to
template3 = map name of third template map to snap to
udrn = numeric value; user-defined registration type

utmln = numeric value; UTM longitude W. decimal degrees)
utmlt = numeric value; UTM latitude N. decimal degrees)

Example(s):

To set the default State Plane coordinate system zone code to Oregon, South (**3601**), enter:

default_head stplzn=3601

See also:

config, header

The **del_points** command will delete all isolated points from the current raster. An isolated point is defined as an on pixel completely surrounded by (**8**) off pixels. This action can be executed against a raw (unassembled) raster. The changes can be saved with a **save** command. No modifications to the vector file will occur. NOTE: To recover raster pixels deleted with the **del_points** command, use the **recall** command.

del_points

Deletes isolated points from a raster objects.

del_points

Removes all isolated points from the raster display. An isolated point may be defined as a node with zero (0) branches. The raster display will be modified to reflect these deletions. To store the changes, execute the **save** command. The vector file will not be modified. Assembly is not required

---

Options/arguments:

None.

Example(s):

To delete all isolated points in the current, active map, enter:

del_points

See also:

del_objects, del_spurs, fill_holes, recall, save

The **del_spurs** command is used to remove all spurs which are less than or equal to the specified number of pixels. A spur is defined as a segment with at least one end free. The changes can be saved with a **save** command. No modifications to the vector file will occur. NOTE: To recover deleted spurs, use the **recall** command.

del_spurs n

Delete spurs from the raster.

del_spurs n

Removes all spurs <= n pixels in length from the raster display. A spur is defined as a segment with one free end (not connected to any other segment). To save these changes, execute the save command. Assembly is not required.

Options/arguments:

n Maximum length of spur to be deleted (in pixels)

Example(s):

To delete all spurs with a length of **2** or fewer pixels from the current, active map's raster, enter:

del_spurs 2

See also:

del_objects, del_points, recall, save

The **dem_view** function allows the user to view a digital elevation model (dem) on the screen as if the topography were side-lit. This allows a final check of the dem file for elevation errors, smoothing, etc.

dem_view

Allows viewing of a dem file.

dem_view fname d

Displays a side-lit dem on the screen. The file viewed is specified by **fname**, and the direction of the lighting by **d**.

Options/arguments:

fname	File name of the dem to be viewed
d	Direction of lighting code, where: **0** = lit from north/northwest **1** = lit from east/northeast **2** = lit from south/southeast **3** = lit from west/southwest

---

Example(s):

To view a dem file named **452212045.dem** as if lit from the **north/northwest**, enter:

dem_view 452212045.dem 0

See also:

export

The data dictionary may be displayed/printed with the **dict_print** command.

dict_print

Prints/displays the data dictionary file to a named device.

dict_print dev dname

Prints/displays the specified data dictionary dname to the named device **dev**. If **dev=2** (*ascii* file), **fname** must be included.

Options/arguments:

dev	Device type code, where: **0** = screen **1** = primary system printer **2** = ascii file (requires fname)
dname	Name of data dictionary file

Example(s):

To route a listing of the data dictionary file called **punk** to the system printer, enter:

dict_print 1 punk

To display the data dictionary file called **dogey** to the screen, enter:

dict_print 0 dogey

See also:

config

The contents of the raster displayed on the screen may be erased with the **disprstr_ers** command. This will not affect the contents of the current raster file unless the erased (blank) raster is saved (see **save** and **disprstr_put**).

disprstr_ers

Erases the entire active raster.

disprstr_ers

Erases the entire active raster.

Options/arguments:

None.

Example(s):

To erase the active map raster display, enter:

disprstr_ers

See also:

disprstr_get, disprstr_mrg, disprstr_put, recall

The contents of the stored (temp) raster or **braster** may be overwritten on the displayed raster, replacing it. This is the purpose of the **disprstr_get** command. The raster is the current raster as edited. The **braster** is normally a copy of the original scan, but can also be a copy of the raster created with the **brstr_create** command.

disprstr_get

Gets the stored raster or **braster** to replace the displayed raster.

disprsts_get r

Replaces the displayed raster with the stored raster or **braster**.

Options/arguments:

r raster code:
a = edited raster
b = braster (original scan) FOR VIEWING ONLY
c = braster (original scan) TO REPLACE/OVERWRITE EDITED RASTER

NOTE: if **b** (original scan for viewing only) is chosen, the user will **NOT** be allowed to edit it and if the map is saved the edited raster will **NOT** be overwritten.

Example(s):

To replace the raster displayed with the temp raster, enter:

disprstr_get a

To display the original scan, enter:

disprstr_get b

See also:

disprstr_ers, disprstr_mrg, disprstr_put, recall

The contents of the stored (**temp0**) raster may be merged with the displayed raster.

disprstr_mrg

Merges the stored raster with the displayed raster.

disprstr_mrg

Merges the stored raster with the displayed raster.

Options/arguments:

None.

Example(s):

To merge the stored raster with the active map raster, enter:

disprstr_mrg

See also:

disprstr_ers, disprstr_get, disprstr_put, recall

The contents of the displayed raster are stored in the **temp0** raster, replacing it.

disprstr_put

Stores the displayed raster over the **temp0** raster.

disprstr_put

Replaces the stored raster with the displayed raster.

Options/arguments:

None.

Example(s):

To replace the **temp0** raster with the displayed raster, enter:

disprstr_put

See also:

disprstr_ers, disprstr_get, disprstr_mrg, recall

Activates the drawing mode.

draw

Activates the drawing input mode.

draw [-s] shape

Activates the graphics input **draw** mode and an optional **-s** automatic snap capability. The snap option causes all enterd **x,y** positions to be corrected to existing coordinate positions (on pixels) if such existing positions can be located within a given radius of the entered location. The search radius tolerance value is in the map **header** file (**search_rad =**). Use the **start** command to begin drawing an object and the **continue** command for all subsequent vertices for that object. Then, use the **start** command to begin a new object. Note that the current graphics input mode/action is displayed in the upper-left window of the LTPlus screen. The **draw** mode remains active until either another mode is invoked or the **clear_mode** command is executed.

Options/arguments:

-s	Invokes snap mode (optional)
shape	Shape type descriptor, including:
	point (single pixel)
	line (pixel unit width sequence)
	fat_line (3-pixel width sequence) rectangle (defined by a pair of x,y positions)
	rectangle (defined by two diagonal points)
	wedge (walking triangle)
	spur a 3 pixel spur attached to the nearest line.

Example(s):

To invoke the **fat_line** draw mode starting with a position defined as **x=456** and **y=222**, with the snap mode activated, enter:

draw -s fat_line (or **draw -s f**)
start 456 222

To continue the **draw -s fat_line** mode at a position defined as **x=488** and **y=225**, enter:

continue 488 225

To remain in the **draw -s fat_line** mode while moving, but not drawing, to a new position defined as **x=500** and **y=236**, enter:

start 500 236

To invoke the **draw rectangle** mode starting at the position defined by **x=790** and **y=345**, with snap mode suppressed, enter:

draw rectangle (or draw r)
start 790 345

To re-position the starting position of the rectangle to **x=690** and **y=345**, enter (prior to any continue commands):

start 690 345

To complete the rectangle drawing action to encompass the rectangular area bounded by **690, 345** and **700,390** (lower-left and upper-right corners....any two, diagonally opposed positions will suffice), enter:

continue 700 390

Note that the **draw rectangle** command produces a unit width perimeter of the defined rectangular area.

To exit from the draw mode, enter:

clear_mode

or enter a new graphics input mode command.

See also:

clear_mode, continue, erase, start

---

edit

Allows the user to edit ascii text files.

edit ftype [fname]

Invokes the system text editor (identified in the **config** file) for the file type and name specified.

Options/arguments:

ftype File type, where the valid options are:

config	**config** file (**fname** not req.)
cursor_key	a **cursor_key** file
default_head	**def. header** file (**fname** not req.)
diary	current map's **diary** file (**fname** not req.)
dictionary	an attribute **dictionary** file
digmenu	a **digmenu** file
digitizer	a digitizer device file
export	a file in the **export** directory
header	current map's **header** file (**fname** not req.)
import	a file in the **import** directory
macro	a **macro** file
map_list	a **maplist** file
menu_set	a **menuset** file
other	any complete file name
pen	a plotter penfile
tolerance	The **tolerance** file (**fname** not req.)

fname File name; not used for the following **ftype** files:

config, default_head, diary, header, tolerance

The **fname** parameter should not be a full path name except when ftype is other. The path name for all other file types is determined by the ftype argument (i.e., if **ftype** is a **macro** and the **macro** name is **cub_mac**, the full path is presumed to be:

/u[sr]/ltp/macros/cub_mac

Example(s):

To display vectors from adjacent maps onto the screen with the current, active map, enter:

edge_add

See also:

edge_maps, graphics, header, erase_margin, margin

Activates the **erase** mode in the specified eraser shape.

erase

Activates the **erase** mode in the specified eraser shape.

erase [-s] shape

Activates the graphics input **erase** mode and an optional **-s** automatic snap capability. The snap option causes all entered x,y positions to be corrected to existing coordinate positions (on pixels) if such existing positions can be located within a given radius of the entered location. The search radius tolerance value is in the map **header** file (**search_rad =**). Use the **start** command to begin erasing an object and the **continue** command for all subsequent vertices for that object. Then, use the **start** command to begin erasing a new object. Note that the current graphics input mode/action is displayed in the upper-left window of the LTPlus screen. The **draw** mode remains active until either another mode is invoked or the **clear_mode** command is executed.

Options/arguments:

-s	Invokes snap mode (optional)
shape	Shape type descriptor, including:
	point (single pixel)
	line (pixel unit width sequence)
	f**at_line** (3-pixel width sequence)
	rectangle (defined by a pair of x,y positions)
	wedge (walking triangle)

Example(s):

To invoke the **fat_line erase** mode starting with a position defined as **x=456** and **y=222**, with the snap mode activated, enter

erase -s fat_line (or erase -s f)
start 456 222

To continue the **erase -s fat_line mode** at a position defined as **x=488** and **y=225**, enter:

continue 488 225

To remain in the **erase -s fat_line** mode while moving, but not erasing, to a new position defined as **x=500** and **y=236,** enter:

start 500 236

To invoke the **erase rectangle** mode starting at the position defined by **x=790** and **y=345**, with snap mode suppressed, enter:

erase rectangle (or erase r)
start 790 345

To re-position the starting position of the rectangle to **x=690** and **y=345**, enter (prior to any continue commands):

start 690 345

To complete the rectangle erasing action to encompass the rectangular area bounded by **690,345** and **700,390** (lower-left and upper-right corners....any two, diagonally opposed positions will suffice), enter:

continue 700 390

Note that the **erase rectangle** command deletes all data (erases on pixels) within the defined rectangular area.

To exit from the **erase** mode, enter:

clear_mode

or enter a new graphics input mode command.

See also:

clear_mode, continue, draw, start

To erase from view any raster data within the margin area, execute the **erase_margin** command. This command affects the raster display only and makes no modification to the stored raster or vector data. The margin area is presumed to be the raster area exterior to the four pre-defined map corners, regardless of orientation (map shape).

erase_margin

Erases the currently defined map margin.

erase_margin

Turns off all pixels in the currently defined margin.

Options/arguments:

None.

Example(s):

To clear (turn off pixels) in the margin of the current, active map (raster view only), enter:

erase_margin

See also:

margin

---

erase_obj

Changes the input mode to erase the raster under user-defined vector objects.

erase_obj [otype]

Changes the input mode to erase the raster under vector objects. The vector objects remain until the map is re-assembled. Only the raster is changed. When using this mode, a 3 pixel-wide path is erased under the vector object chosen. The program searches within the specified search radius (**header**) when in this mode. NOTE: The undo command will retrieve only the last 32 segments removed with **erase_obj**. The recall command will retrieve any or all of the raster removed with **erase_obj**. The **disprstr_get** command will replace the displayed (and modified) raster with an untouched version (since the last save command).

THE DIFFERENCES BETWEEN **erase_obj** AND **erase_seg** ARE:

1) **erase_obj** requires assembly while **erase_seg** does not.
2) **erase_obj** will only erase under a vector object.
3) **erase_obj** will stop erasing a segment or polygon when the position of a vector node is reached, even when the node is no longer a junction or endpoint on the raster. **erase_seg** will erase any raster data back to the first raster junction in both directions, no matter what.
4) **erase_obj** checks for an endpoint and length match between the raster and vector objects before erasing. **erase_seg** does no checking.

Options/arguments:

otype Object type code, where:
0 = node
1 = segment
2 = polygon

The default is the **map_type = string** in **header**)

Example(s):

To erase the segment closest to the position defined by **x=450** and **y=678**, enter:

erase_obj 1
continue 450 678

See also:

del_objects, disprstr_get, erase_seg, header, recall, save, undo

---

erase_seg

Changes input mode to erase raster segment closest to input position.

erase_seg

Changes input mode to erase the raster segment closest to location specified by the **start** or **continue** commands. When in this mode, **start** or **continue** causes the file to be searched for the closest raster pixel to the input position (limited by the user-defined search radius in the header file). The raster is then erased from that pixel in both directions until a raster junciton or spur endpoint is reached. NOTE: To repair changes made with **erase_seg**, reference **disprstr_get, recall** and **undo** commands.

THE DIFFERENCES BETWEEN **erase_obj** AND **erase_seg** ARE:

1) **erase_obj** requires assembly while **erase_seg** does not.
2) **erase_obj** will only erase under a vector object.
3) **erase_obj** will stop erasing a segment or polygon when the position of a vector node is reached, even when the node is no longer a junction or endpoint on the raster. **erase_seg** will erase any raster data back to the first raster junction in both directions, no matter what.
4) **erase_obj** checks for an endpoint and length match between the raster and vector objects before erasing. **erase_seg** does no checking.

Options/arguments:

None.

Example(s):

To erase the segment closest to the position defined by **x=450** and **y=678**, enter:

erase_seg
continue 450 678

See also:

del_objects, disprstr_get, erase_obj, erase_seg, header, recall, save, undo

---

As its name implies, the **execute** function will pass a command to the host operating system. This allows the user to remain in LTPlus while invoking a UNIX/Xenix operating system command. The keyboard and display are returned to their default conditions while the executed command is underway. Upon completion of the executed system command, the user is returned to the LTPlus environment.

execute

Passes a command to the UNIX/Xenix operating system for invocation.

execute [-s] [command]

Passes a command to the host operating system and returns to the LTPlus environment when complete. If no command parameter is supplied, the shell command is invoked. If **-s** (silent) is specified, then the command is executed without changing the graphics mode or giving the user a prompt. When using the **-s** parameter, no user input is allowed and no command output is displayed. If **-s** is NOT specified, the user is put into a normal C-shell screen, is allowed input and shown normal screen output as the command is executed, and is prompted to press [RETURN] to continue befor going back to the LTPlus graphics mode.

Options/arguments:

-s	Silent flag
command	An operating system command with valid syntax.

Example(s):

To display all processes running on the operating system while in an LTPlus session, enter:

execute ps -ef

To copy the file **/u/ltp/config/config** to **/u/ltp/config/cbackup** without leaving ltplus graphics mode, enter:

execute -s cp /u/ltp/config/config /u/ltp/config/cbackup

See also:

shell

The **exit** command may be executed by any user, at any time during an LTPlus session. Its sole and obvious purpose is to permit a graceful exit from the LTPlus processing environment. Graceful means that program and data files will be protected from the sorts of violent and aggressive actions which may occur during an ungraceful or unplanned exit. The **exit** command is a kinder and gentler means of escape than what you would experience if some idiot were to trip over and unplug the cpu power cord (a thousand points of light...).

exit

Exit LTPlus.

exit [n]

Exit from LTPlus. Setting the **n** value to **0** will exit without saving the current map, while a value of **1** will cause a save to occur upon exiting.

Options/arguments:

n Map save/not save flag, where
0 = no save
1 = save

Example(s):

To leave LTPlus and save the current map, enter:

exit 1

See also:

EXPORT FORMATS

Data may be exported for input to a variety of other applications. Export files are typically placed in the **/u[sr]/ltp/export** directory. The *ascii* **export** file may also be written directly to a DOS partition or DOS-format floppy (as with the import command). The **export** function supports a number of popular data formats, including:

cell	This is a fairly generic cell format for handling grid-based information. It is commonly used for importing data from MTHMAP and R2MAP.
cff	Cartographic Feature File is the vector format associated with the base layer data supplied by the Geometronics Service Center (GSC).
dem	Digital Elevation Model, a term created by the U.S. Geological Survey to describe topographic data stored as horizontal and vertical coordinates.
dig	A vector format used to bring digitized data into the ERDAS software environment. See also the **import gis** format, used by ERDAS.
digit	A vector and attribute format for the GRASS GIS (SCS).
dlg3	This is a comprehensive vector format developed by the U.S. Geological Survey and compatible with a variety of external applications, including the GRASS and ARC/INFO GIS.
dxf	Defined by Autodesk (AutoCad). This Vector format is the CAD industry standard.
moss	This is a data-intensive vector format which supports the MOSS GIS application.
rdef2	Raster Data Exchange Format, developed for use with DWRIS, LIDES and LTPlus.
rstr	A generic raster format.
scsgef	This format was developed for use by the SCS as the digital soil survey component of SSURGO data base. It is a vector format.
snap	This includes formats for both polygon (timber unit) and line (roads) output files. Both are vector files intended for use with USDA Forest Service PC-Snap software.
xy	Any user specified (generic) xy vector format.

EXPORT COMMAND PARAMETERS

The **export** command writes data in the format specified by type to the named file. If the specified file name does not contain '/' character, then it is assumed that it is in the default path (**config** file). The data is read from the active map. A file may be exported directly to a DOS device (a:, b: or c:) by including the appropriate device identifier as part of the full pathname (**fname**) or by setting the **export_path** element of the **config** file to the desired device identifier (a:, b: or c:). Different format types require different optional parameters **[p1]...[pn]**, which should be included in the indicated sequence following the file name. The parameters associated with each command are (note that parameters in brackets are optional):

cell fname t c w h [str]
cff fname c s
dem fname c w h g e1 e2 st f
dig fname a c s
digit fname a c s i
dlg3 fname a at c s
dxf fname a c s [l]
moss fname a c otype s
rdef2 fname a rc
rstr fname b n m
scsgef fname [-f]
snap fname otype c s
xy fname c s cd dl dp ht i1 i2 a1 a2 v1 v2 x1 x2 y1 y2

The parameter definitions are:

a Attribute code:
1 = export objects only
2 = export attributes only
3 = export both objects and attributes
4 = use attribute for 3D elevation value (avail. with dxf output only)

a1 Beginning column for attribute information.

a2 Ending column for attribute information.

at Attribute type code:
1 = numeric code pairs
2 = text strings

b Bits-used code (1-255)
This value specifies the bits of each byte to be taken as raster data. To use the low four bits, set this value to 15; to use all eight bits, set this value to 255. In general, the bits used form an 8-bit binary integer, less the flag bits used.

c Coordinate type code
0 = raster pixels
1 = lon/lat, ddd/mm/ss.ss, (arc-seconds for DEM format only)
2 = utm meters
3 = state plane feet
4 = digitizer1 coordinates
5 = digitizer2 coordinates
6 = map inches
7-9 = reserved
10-19 = user-defined coordinates

cd Coordinate display type (not coordinate system code):
1. decimal x & y (or decimal degrees)
2. ddd/mm.mm (degrees, decimal minutes)
3. ddd/mm/ss.ss (degrees, minutes, decimal seconds)

dl ASCII decimal value of delimeter used in ddd/mm/ss.ss notation
(it would be 47 in the example above, for '/').
NOTE: 32 = ' ' (blank), 39 = "' (apostrophy), 44 = ',' (comma), 45 = '-', 47 = '/'.
NOTE: this character will immediately follow the degrees or minutes ending column specified and will be there only if the next smaller value is specified.

dp Number of decimal places for output of x and y coordinates.

e1 Map attribute units:
1 = feet (* normal)
2 = meters

e2 Output dem units:
1 = feet
2 = meters (* normal)

f Number of smoothing passes:
0 = none, no smoothing
1-3 = number of passes (normally 1)

-f Flag for creation of feature file (**scsgef** format only)

fname Export file name
The **fname** parameter may be a file name (with or without a complete path), or a directory name. If **fname** does not contain the '/' character, it is assumed to be a file name in the default export directory (listed in the **config** file). If it does contain the '/' character, it is assumed to be a path name. The path is checked to see if it is a file or a directory. If it is a file, the file is imported. If it is a directory, the program looks in that directory for a file which has the same name as the map. This is to allow easy batch exporting for a list of maps.

g Clipping coordinate type code
Note, clipping allows a small portion of overage [1/10,000 of the length of a side extra]. This allows the inclusion of dem points within 2-3 feet outside the quad proper.
0 = raster pixels
1 = lon/lat, ddd/mm/ss.ss
2 = utm meters
3 = state plane feet
4 = digitizer1 coordinates
5 = digitizer2 coordinates
6 = map inches
7-9 = reserved
10-19 = user-defined coordinates

h Cell height (pixels; normally 30)

ht Header type code:
1 = no header lines at all.
2 = header line for each object. Attribute on header line.
3 = header line for each object. Attribute on vertex line(s).
4 = ONLY header lines, with x,y,attribute (opt. item number) using object centroid locations.

i Flag for digit [**GRASS**] export format:
0 = Place attributes directly in digit attribute file (no cat file). Use this option **ONLY** if your attributes are **GRASS** catagory numbers. With this option, no **'cat'** file will be output.
1 = Create **GRASS** catagory numbers for each attribute string. Put new catagory numbers in attribute file, and create a **'cat'** file containing catagory number and attribute corrispondence.

NOTE: GRASS attribute file contains catagory numbers. Usually these begin with 1 etc. GRASS attributes CAN ONLY be assigned as catagory numbers (NO TEXT). GRASS uses another catagory file which allows a 1-line description for each catagory number. It is usually simpler to assign catagory numbers to objects in LTPlus, if GRASS output is desired.

i1 Beginning column number for item number (1st col = 1), **0** = none.

i2 Ending column number for item number (1st col = 1), **0** = none.

l Layer separation flag
If **l=0**, then the dxf file written will have LTPlus objects put into separate layers as follows:
dxf layer - LTPATR All LTPlus attributes will be written as DXF text on this layer.
dxf layer - LTPOBJ All LTPlus objects (points, lines, and polygons) will be written as dxf points or polylines on this layer.

If the layer separation flag = **1**, then all LTPlus attributes will be put on a DXF layer called LTPATR, and all LTPlus objects will be put on a DXF layer which has a layer name the same as their first LTPlus attribute, only capitalized. Objects without attributes will have a layer name of LTPOBJ. (When the layer separation is **1**) there will also be a header item for extents and acad version number. IF there is a DICTIONARY for the map, there will be a DXF LAYER TABLE made from that dictionary (same layer names [only CAPITALIZED] in the same order.

NOTE: The layer table will automatically include a **dxf 0** layer at the beginning. Do not put one in your dictionary. ALSO: The layer table will automatically include a last layer called LTPATR for all the atribute/text data IF attributes are included in the export statement.

m Maximum line length

n Mask: Value of 0-255, inclusive, specifies byte to be combined with the data.

otype Object type code:
0 = any node
1 = any segment
2 = any polygon
3 = node with 0 branches: an isolated point
4 = node with 1 branch: a spur endpoint
5 = node with 2 branches: a node on a simple island
6 = node with 3 branches: a junction node
7 = node with 4 branches: a junction node
8 = segment with 2 free (spur) ends: a free segment
9 = segment with 1 free (spur) end: a spur
10= segment with no free ends
11= single links to islands (segment)
12= segments between polys with same attribute
13= possible segment contour attr. sequence error
14= interior polygon
15= exterior polygon (negative area)
16= map border exterior polygon (negative area)

rc Registration code:
0 = omit registration
1 = include registration

s Set number of objects to be exported

sn State abbreviation and name of soil survey

st Smoothing type, where:
0 = none, no smoothing
1 = smooth all points
2-**2000** = variable smooting

All dem grid points which are on a (min) profile section longer than **st** will be smoothed. Except for grid in decimal degrees (min/sec), the **st** value is in grid units; e.g., if the dem is on a UTM grid, then **st** is in meters. For a standard dem from 40 ft. contours, a value of 200-400 will normally do a good job of leaving well-contoured areas untouched while smoothing areas of less than 5-2.5% slope. This removes streaking in flat areas typical of programs like this which use multiple profiles to calculate dem's. Note that ***lower numbers mean more smoothing***, while ***higher numbers mean less smoothing***. For DEMs gridded along Latitude and Longitude lines, the smoothing value is in approximate ground meters for north-south distances and a smaller value for east-west distances. This means that the value for **st** will be about the same as for **utm** gridded DEMs, but that smoothing will be heaver when contours run north-south than east-west. This effect is greater the further north the DEM.

str Characters used for polygons with no attributes.

t Characters per cell

v1 Beginning column (in header line) for number of verticies.

v2 Ending column (in header line) for number of verticies.

x1 Beginning column number for x coordinate.
If the coordinate type is geodetic, you may enter dd.mm.ss where dd = column numbers for the beginning of the degrees, minutes, and seconds (0 indicates leave out).

x2 Ending column number for x coordinate.
If the coordinate type is geodetic, you may enter dd.mm.ss where dd = column numbers for the ending of the degrees, minutes, and seconds (0 indicates leave out).

y1 Beginning column number for y coordinate.
If the coordinate type is geodetic, you may enter dd.mm.ss where dd = column numbers for the beginning of the degrees, minutes, and seconds (0 indicates leave out).

y2 Ending column number for y coordinate.
If the coordinate type is geodetic, you may enter dd.mm.ss where dd = column numbers for the ending of the degrees, minutes, and seconds (0 indicates leave out).

w Cell width

Example(s):

To format the current, active map file for input to MOSS and store the new format as a file called **cub_moss,** where objects only are passed, with coordinates in state plane feet and where the objects are polygon only and the set containing the selected objects to be formatted is number **3,** enter:

export moss cub_moss 1 3 2 3

See also:

crit_type, coord_type, import

The **fill_holes** command will evaluate the current raster and turn on any single *off* pixel which is surrounded on four sides by *on* pixels. This single *off* pixel is known as a hole. The raster need not be assembled. The inverse of **fill_holes** is **del_points**.

fill_holes

Fills all single-pixel holes in the raster.

fill_holes

Fills all single-pixel holes within the raster (inverse of **del_points**). Assembly is not required.

Options/arguments:

None.

Example(s):

To fill (turn pixels on) all single-pixel holes within the raster of the current, active map, enter:

fill_holes

See also:

del_points

The **frame_map** command connects or disconnects the four map corners, in sequence, on the raster display. To generate a visible map border, the **d** (**draw**) option should be included as an argument to this command. To erase an existing map border (frame), the command should include the **e** (**erase**) parameter. In addition, the user may designate that the frame be calculated and drawn in the raster coordinate system by entering **r** as a parameter to this command, or in a corrected geodetic projection by supplying a **g** with the **frame_map** command. The raster (**r**) option will simply connect the four corners as they are stored in pixel coordinates derived during the raster registration process. A geodetic (**g**) frame will connect the four corners as they are calculated from their proper geodetic positions.

frame_map

Draws/erases the map border.

frame_map action arg

Draws/erases lines from SW to NW to NE to SE to SW raster corners, thereby creating a frame, or border, around the active map.

Options/arguments:

action Framing action, where:
d = draw
e = erase

arg Type of fram eto be drawn/erased, where:
g = geodetic
r = raster

Example(s):

To draw a frame for the geodetic corners, enter:

frame_map d g

To erase an existing raster frame, enter:

frame_map e r

To erase a geodetic frame, enter:

frame_map e g

To erase a geodetic frame, enter:

frame_map e g

See also:

erase_margin, margin, level_dup

Only one graphics input device may be active at any given time, although up to three devices may be present. The standard three devices are the mouse, digitizer 1 and digitizer 2. Each device type and address should be identified in the **config** file. The primary, default startup device is also set in the **config** file as **ginput = (dev)**. The user may reset the graphics input device at any time.

Entering an optional **-a** argument sets the coordinate mode to absolute; the default (no argument) is relative mode. In relative mode, LTPlus is basically treating the incoming x,y positions as relative movement, with no relationship between the screen cursor position and the map location on the digitizer. This is perfectly sufficient for most LTPlus activities. However, absolute mode is desirable when digitizing or editing map data; it provides the correlation between movement on the map surface and the position of the screen cursor on the corresponding map screen display.

ginput

Select an active graphics input device (**GIN**).

ginput gdev [-a,-d]

Selects graphics input device based on argument **gdev**. Option **-a** specifies that the coordinates be received from **gdev** as absolute in nature; the default setting is for relative coordinates. The **-d** argument allows the user to perform debugging tasks.

Options/arguments:

gdev Graphics input device; valid entries are:
mouse (or m)
dig1 (or 1)
dig2 (or 2)

-a Specifies absolute coordinates

-d Specifies debug mode. When this option is chosen, the first 1600 characters or 10000 reads (whichever comes first) are printed on the screen. After that, the input is used as normal digitizer input. This allows debugging of a digitizer interface after the basic baud, address and output sequences are correct.

Example(s):

To select digitizer 1, in absolute mode, as the active graphics input device, enter:

ginput -a dig1

See also:

config

The screen cursor may be re-positioned to a specified location. Note that the current coordinate type units are in effect with regard to the **x,y** arguments for this command.

goto

Re-positions cursor to specified location.

goto type arg1 arg2

Re-positions cursor to new location based on the specified locator type and the current cursor x,y location. Note that the view window, but not the cursor, will be moved when using the digitizer for input in absolute mode.

Options/arguments:

type Locator type, including:
serial, which requires
arg1 = serial number
OR
record, which requires
arg1 = record number
OR
coordinate, which requires
arg1 = x value (decimal), in current units
arg2 = y value (decimal), in current units
OR
log, which requires
arg1 = record # in log file
OR
= + for next record
OR
= - for previous record

Example(s):

To re-position the cursor to the object with record number **47**, enter:

goto record 47

To re-position the cursor to the object defined as the next log file entry, enter:

goto log +

To re-position the cursor to the position with lat/lon equal to **43/24/55.99N 119/17/36.25W**, enter :

coord_type 1
goto coord 119.2934028 43.41555278

NOTE: **x** [longitude in this case] is always first, and decimal degrees must be used when current units are lon/lat. The **coord_type** command is used to change the currently displayed units.

See also:

log, coord_type

Chapter 8 Display and Level Operations

The **graphics** command provides the user with full control of the type and size of the map display. The user may designate that the entire screen width be used for the graphics display by entering the optional **-f** argument. The default condition is a normal, TILE 7 (square) map display. Vectors from any set may be displayed in any combination as normal or highlighted. NOTE: The normal color relates to the **pen3 = value** in the **config** file; the highlight color relates to the **pen6 = value** in the **config** file. However, if normal is displayed over the raster then the color displayed will be the **pen7 = value**. When highlight is displayed over the raster, the color will be the **pen8 = valu**e. When highlight and normal are displayed together, with or without raster, highlight will dominate (or overwrite) normal (i.e., h + n = h and h + n + r = h + r).

graphics

Controls the graphic display.

graphics [+-c] [+-f] [+-r] [+-nXX[-YY]] ... [+-hXX[-YY]] ...

Controls the graphic display. If NO PARAMETERS are used, then the graphics display is reset (show raster only, no vectors, and show menus). If PARAMETERS ARE used, then they are used as changes to the current display. For all parameters, the plus **(+)** sign means to add something to the current display. The minus **(-)** sign means to remove it. **c** is for showing polygon centroids. **f** is for full-width graphics display, **r** is for showing the raster, **n** is for displaying vectors in 'normal' color, **h** is for displaying vectors in 'highlight' color.

NOTE that the **c**, **n**, and **h** parameters concern vector data only and will have no effect unless the map has been assembled or you have imported vector data to display.

The n and h parameters may be used with a single level (**+n35**) or on a range of levels (**+h35-42**). If a vector level is displayed in both normal (**+n**) and highlight (**+h**), it will appear the same as if it was just highlighted.

NOTE that polygon centoids are only dispalyed when the vector data for the polygon is also displayed, and in the same color (highlight/normal), and that this is can only be done for sets **0-31** (**not 32-95**).

NOTE that the parameters may be listed in any order, in any combination.

Options/arguments:

+	Add the option specified to the display
-	Remove the option specified from the display
+-c	Display/undisplay centroid marks (only displays if polygons are displayed)
+-f	Full screen width graphics display (no menus)
+-r	Dispay/undisplay the raster
+-nXX	Add/remove the vectors in set/level **XX** in normal mode
+-nXX-YY	Add/remove the vectors in range **XX** through **YY** in normal mode (**XX < YY**). (see table below)
+-hXX	Add/remove the vectors in set/level **XX** in highlight mode
+-hXX-YY	Add/remove the vectors in range **XX** through **YY** in highlight mode (**XX < YY**). (see table below)

XX:	**0**	all map objects (pts/lns/polys)
	1-24	map objects in this set number
	25	map objects with 1+ attributes

26	map objects with no attributes
27	map objects with 2+ attributes
28	map objects with <2 attributes
29	map objects w/out 2 identical attrs
30	segs w/same poly on both sides
31	reserved
32	geodetic window (in vector form, following the projected curve of latitude and longitude)
33	geodetic planar (straight) map frame
34	geodetic corners
35-95	user imported vector files (dlg, moss, etc.)

Example(s):

To activate a full screen-width display, enter:

graphics +f

To display the vectors in set **12** highlighted, enter:

graphics +h12

To display (unhighlighted) the imported vector files on level **45**, enter:

graphics +n45

To reset **graphics** to powerup status (all vector display off, raster display only in the large square graphics area), enter:

graphics

To remove the raster from the display, remove set **16** and **import** level **42** from normal display, and add sets **4** and **7** highlighted and **import** vector level **35** (normal), and highlight **import** levels **88** through **91**, and change the display from full width to the square graphics area only, enter:

graphics -r -n16 -n42 +h4 +h7 +n35 +h88-91 -f

See also:

redisp_map

A *spur* is defined as a line segment which is either connected on only one end, or not connected at all. The **grow_spurs** command will find all spur end locations on the raster and attempt to close it with another spur end, or by extending it into another raster object. It uses the same logic as the **bridge_gap** input mode. NOTE: a gap between two spurs will be closed up to twice the distance specified in the command. A spur will only be extended up to the input distance into another raster object which is not a spur. NOTE: If a closure cannot be found within the specified distance, the spur will be left alone (no pixels will be added).

grow_spurs

Attempts to close all spurs, using the input number of pixels or less.

grow_spurs p

Attempts to close all spurs, using the input number of pixels or less.

Options/arguments:

p Maximum number of pixels to be added

Example(s):

To attempt to close all spurs with 15 pixel gaps or less enter:

grow_spurs 15

See also:

del_spurs, bridge_gap

The **header** file, stored as **/u[sr]/ltp/maps/map_name/header**, contains information relevant to the current, active map. Some of the parameters in the **header** affect the processing environment, while others provide valuable data which can be integrated into a file header for some of the export formats. The **header** command allows the user to update and/or display the **header** file.

header

Updates/displays the **header** file.

header [-p] [item=arg] ...

Display and/or modify the **header** file for the active map. Multiple **item=string** pairs are allowed per execution of **header**. The **-p** option prints the **header** file to the default primary printer.

Options/arguments:

-p Output to printer
item = arg may be any of the following:
banner = text (max 72 chars)
comments = text
compress = yes/no
contour_interval = n
cursor_file = name of cursor key definition file
datum = yr
dictionary = text : name of data dictionary
edgematch = [N] [E] [S] [W]
edge_e = text : name of map to the east
edge_n = text : name of map to the north
edge_s = text : name of map to the south
edge_w = text : name of map to the west
head1 = header value (1st 72 char) used in dem output
head2 = header value (2nd 72 char) used in dem output
layer_name = text : map layer
map_ht = map height in minutes (def=7.5)
map_wd = map width in minutes (def=7.5)
maptype = text : type of map
menu_set = text : name of menuset
quad_unit = text : quad or unit name
revision_yr = revision year
scale = scale ratio
search_rad = pixel radius for search
source_yr = source year
state = 2-digit state code
stplzn = state plane coord zone code
template1 = map name of first template map to snap to
template2 = map name of second template map to snap to
template3 = map name of third template map to snap to
udrn = user-defined registration type
utmln = UTM longitude W (decimal degrees)
utmlt = UTM latitude N (decimal degrees)

Example(s):

To set the state plane coordinate system zone code to Oregon, South (**3601**), enter:

header stplzn=3601

To identify an adjacent map name, **quill_creek**, to the west, enter:

header edge_w=quill_creek

See also:

config, default_head

---

help

Displays information for a specified topic.

help t topic

Displays helpful information for the specific **topic**, in the defined length format, **t**.

Options/arguments:

t	Quantity of help information displayed, where **s** = short **l** = long
topic	Name of function, menu or macro

Example(s):

To display useful information about the **import** funtions, enter:

help import

See also:

macro_hist, next_step

Chapter 15 Import and Export

Imported raster information, which includes scanned data, is imported directly into the current, active map's raster. Vector information is imported into a user-designated level of the current, active map and may be moved to the map's raster with the **level_dup** or **level_icopy** commands if desired. As an example, one of the more common routes involves importing information scanned by a Houston Instruments (HI) ScanCad system. The data is output by the HI system in a DOS raster format file with a .**ras** extension. To move this file into LTPlus, the user should execute the **import** command with the .**ras** format type included as a parameter. The LTPlus **import** formats include:

asn Raster format generated by USFS FASTGRAMS conversion software.

cell This is a fairly generic format for handling raster, grid-based information. It is commonly used for importing data from MTHMAP and R2MAP.

cff Cartographic Feature File is the vector format associated with the base layer data supplied by the Geometronics Service Center (GSC).

dem Digital Elevation Model is a list of elevations and header data as defined by the U.S. Gelogic Survey

digit.ascii An ascii vector format used by GRASS (CERL/SCS).

digit.att An ascii format holding attributes and their locations, used by GRASS (CERL/SCS).

dlg3 This is a comprehensive vector format developed by the U.S. Geological Survey and compatible with a variety of external applications, including the ARC/INFO GIS.

dxf Rapidly becoming the CAD industry standard, this vector format is most closely associated with the AutoCad software package.

gis A cell type format output by ERDAS its .gis file. When this format is imported, the entire raster is erased and resized to fit the cell data. All vectorized data, attributes and registration are lost. You should only import this data into a newly-created map. This format carries registration, polygon and attribute information. Reference the dig format (used with ERDAS).

map LTPlus internal map files format.

moss This is a data-intensive vector format which supports the MOSS GIS application.

optr9 A raster format developed to transport digital data generated by the Optronics scanner at GSC.

ras A specialized raster format generated by the Houston Instruments ScanCad system.

rdef2 Raster Data Exchange Format, developed for exchanging raster data among DWRIS, LIDES and LTPlus.

rstr A generic, bit-mapped raster format.

rlc A run-length encoded raster format used by several scanners including the Ideal Context scanner.

rle1 A one-byte, Run Length Encoded raster format.

scsgef An SCS format that may consist of three or four files that hold header, segment coordinate pairs, points, and an optional feature file containing isolated segments. The line file is is required for this category of **import**. The point and feature files may be useful as well. The header data is not currently used.

tan The LTPlus name for a pair of files output by the Tangent scanner. This raster format consists of two files which have the same name with different extentions. The .cmd file has header information, and the .img file is a bit-mapped raster file.

xy The LTPlus name for any user-specified ASCII xy vector format. This can be any file which contains x,y information with optional attribute information. Each vertex must be on a separate line. There may (or may not) be a single header line between vertex lists. All items must be separated by at least one blank (if not, they can be inserted using **b_edit**).

NOTE: When importing raster or cell files, a duplicate copy of the original scan or cell file is automatically stored as the braster. YOU DO NOT HAVE TO CREATE A BRASTER (**brstr_create**) MANUALLY.

import The **import** command reads data in various file formats and, depending on the format, places them on the raster (base level) or at the specified display/import level (35-95).

VECTOR FILES
Files in vector format imported by the user are put onto an import/display level specified by the user. There are 61 of these levels available, numbered 35 - 95. From there, they can be displayed (graphics), used to modify raster-based data (**level_snap**), or duplicated onto the raster itself (**level_dup** or **level_icopy**). All (and only) **import** formats which include the **l** (level) parameter are of this type. They include:

cff dem digit.ascii dlg3 dxf moss xy

NOTE: Whenever vector data is imported, it is also displayed on the screen. To remove it from the display or make other display changes, use the graphics command.

RASTER AND CELL-BASED FILES
Other file formats include several types of raster and cell-based data. These are imported directly onto a new raster and destroy all old data for the map to which they are imported. **Import** formats of this type do not have the l (level) parameter. These include:

cell gis optr9 ras rdef2 rlc rstr tan

These types do not have explicit vector or topology information, but do have relative position information. In LTPlus, this is handled most efficiently by sizing a raster exactly to the file so that nothing is lost or distorted. This destroys the old raster coordinate system which links all of the old data together (on that map). If you want to bring this type of data onto an existing map, first **import** it into a blank map, **assemble** the data, then **export** a vector format file (i.e., **dlg3** or **moss**) and **import** that vector file onto an import/display level of the intended target map.

NOTE: When you **import** raster data, it is placed in a raster file which can be edited AND another copy is also stored as a **b-raster** to save a copy of the original scan which can be used to check the final output or for other uses.

SPECIAL FILES
One type of format includes attribute information only (an attribute string, object type to which it belongs, and location information). These include:

digit.att

Another type of special format is the **dem** format. This USGS format consists of header information about the position, data density, and accuracy of the data, and the main data portion which consists of lists of elevations at evenly spaced positions on a coordinate grid. Data in this format are imported into LTPlus as vector points, with elevations as their attributes. The format is:

dem

Another type of special format is the **scsgef** format. The map data is distributed into three or four separate files. Only one **scsgef** file may be imported at a time! The coordinates are in lon/lat format, in decimal degrees. The files are:

header - contains name of digitized soil survey, map corner coordinates, map scale, etc. This file is not implemented in LTPlus yet.

line - contains coordinate pairs (lon/lat) of the segments. Both right and left polygons are listed along with their respective attributes.

point - contains location soil map symbols and/or soil point features.

feature - contains linear features that are not part of polygons. A digitized escarpment is the typical example of a linear feature. This is an optional file!

import type iname [p1] ... [pn]

Imports a file in format **type** at location **iname** into the currently acquired map. The imported data will be placed on the level specified by the user (for vector type files) or replace all map data (for cell and raster type files); reference the important notes, above.

NOTE: Under SCO Unix or Xenix, a file may be imported directly from the DOS partition (on the C drive) or from a DOS diskette in drives C or B. The full path name, included as a parameter, may include the DOS device location (A:, B:, or C:) or the import_path argument in the **config** file may be set to one of the DOS device identifiers (A:, B:, or C:).

IMPORT COMMAND PARAMETERS
Different format types require different optional arguments **[p1]....[pn]**, which should be included in the indicated sequence following the file name. The **import** command takes different parameters depending on the format to be imported. Here are all of the different forms (one per format) with Options/arguments:

import asn iname
import cell iname n a
import cff iname l c
import dem iname l
import digit.ascii iname l c
import digit.att iname c s ac
import dlg3 iname l
import dxf iname l c
import gis iname a
import map iname l
import moss iname l c otype
import optr9 iname

import ras iname r t
import rdef2 iname a
import rlc iname r t
import rle1 iname
import rstr iname b j
import scsgef -ftype iname l
import tan iname r t
import xy iname l c otype ht in v at x1 x2 x3 y1 y2 y3

The various parameters displayed in this list are defined as follows:

a Attribute code:
1 = import objects only
2 = import attributes only
3 = import both objects and attributes

ac Action code (for attributes)
0 = Replace all attributes with a single new one
1 = Replace/set 1st attribute, leave others
2 = Replace/set 2nd attribute, leave others
3 = Insert new 1st attrribute, push others up by 1
4 = Prefix 1st existing attribute
5 = Suffix 1st existing attribute
6 = Add a new (additional) last attribute at item number on the ascii line of the attribute.
(the attribute, if specified, may not contain blanks, (0 = none).

b Bits-used code (1-255)
This value specifies the bits of each byte to be taken as raster data. To use the low four bits, set this value to 15; to use all eight bits, set this value to 255. In general, the bits used form an 8-bit binary integer, less the flag bits used.

c Coordinate type code
0 = raster pixels
1 = lon/lat, ddd/mm/ss.ss
2 = utm meters
3 = state plane feet
4 = digitizer1 coordinates
5 = digitizer2 coordinates
6 = map inches
7-9 = reserved
10-19 = user-defined coordinates

-ftype type of **scsgef** file where the type might be:
g indicates file named by **iname** is to be imported as the **header** file
l indicates file named by iname is to be imported as the line file
p indicates file named by **iname** is to be imported as the point file
f indicates file named by **iname** is to be imported as the feature file

ht object header (type of header line for each vector object) type:
1 = none, no header lines for vector objects.
2 = ONLY header lines containing an xy location (centroid) and an attribute. NO vertex data.
3 = header line containing the attribute and item number (if any).
4 = header line containing the attribute (if any), but the item number is in the vertex line(s).
5 = header line for each object containing the item number (if any), but the attribute is in the vertex line(s).
6 = header line for each object, but attribute & item number (if any) are both in the vertex line(s).

in Item number in the file line of the object's item number. Confusing enough for you?

Ok, for example, in **moss import/export** format the header line for object number **25** might look like this:

25 ponderosa_pine 126

The item number of the object is 25 and is the first group of characters on the line. In this example the item number of the 'item number' (object number) is 1 (the First item on the line).

iname **Import** file name
The **iname** parameter may be a file name (with or without a complete path), or a directory name. If **iname** does not contain the / character, it is assumed to be a file name in the default **import** directory (listed in the **config** file). If it does contain the / character, it is assumed to be a path name. The path is checked to see if it is a file or a directory. If it is a file, the file is imported. If it is a directory, the program looks in that directory for a file which has the same name as the map. This is to allow easy batch importing for a list of maps.

j Ratio of **import** file ascii lines per one scan line.

l Level number of base category

Each **dlg3** file may contain one or more data categories, with each category imported to one, user-designated LTPlus level which lies within the inclusive range of 35 and 95. It is suggested that an organized pattern of level usage be adopted. For example, all adjacent map data imported (for edgematching) might be placed on levels 35-38, with the map to the west on level 35, etc. This allows much more efficient usage and makes it easier to track your data.

n Number of ascii characters per cell in a cell file; normally 1 or 2.

otype Object type code

MOSS i**mport/export** type files hold only one type of object per file:
0 = points
1 = lines
2 = polygons (forces closure when in xy import format)
xy (user defined) import allows another type, automatic:
3 = auto: (**1 vertex=node**, **2=seg**, **3+=seg**[unclosed] or **poly**[closed]

r Reduction factor for raster resolution of **.ras** files.

When raster resolution is reduced, several scan pixels are grouped into one raster pixel in LTPlus. A 3:1 linear reduction would convert a 3 x 3 block of scan pixels into a single LTPlus raster pixel. The allowable values are:

1 = no reduction
2 = 2:1 linear reduction
3 = 3:1 linear reduction
4 = 4:1 linear reduction
5 = 5:1 linear reduction
6 = 6:1 linear reduction

The advantages of reduction are speed in processing, and smaller file sizes. If the scan has much higher resolution than the original data, then reduction should be given serious consideration.

s Set number

Set number of map objects in the current, active map to which to assign the imported attributes. If an imported attribute location associates it with an object in set **s**, then the attribute will be assigned. Otherwise, the attribute will not be assigned.

t Fractional threshhold for use with **r > 1**, above.

When raster resolution is reduced, several scan pixels are grouped into one raster pixel in LTPlus. The **t** value(fractional threshhold) allows the user to specify the portion of the original scan which must be on to produce an on raster pixel. If **t=.25**, then 25% of the original scan pixels in a reduced group would have to be *on* to get an *on* LTPlus raster pixel. The allowable values are between 0.0 and 1.0.

v Item number on the vector object header line of the number of verticies. (0 = none)

x1 Item number on the ascii line of the x coordinate in units of type c. NOTE if coord type 1 is specified (lat/lon) then x1 specifies where degrees (or decimal degrees) are located. IF the degrees/min/sec ARE NOT SEPARATED BY BLANKS (e.g. 45/22/30.0) then this item will specify them all (SPECIFY 0 [=none] for x2 & x3).

x2 Item number on the ascii line of min (1/60 units) of x coordinate value in units of type c. USE THIS ITEM ONLY IF degrees min sec ARE SEPARATED BY BLANKS.

x3 item number on the ascii line of sec (1/3600 units) of x coordinate value in units of type c. USE THIS ITEM ONLY IF degrees min sec ARE SEPARATED BY BLANKS AND SECONDS ARE USED.

y1 item number on the ascii line of the y coordinate in units of type c. NOTE if coord type 1 is specified (lat/lon) then y1 specifies where degrees (or decimal degrees) are located. IF the degrees/min/sec ARE NOT SEPARATED BY BLANKS (e.g. 45/22/30.0) then this item will specify them all (SPECIFY 0 [=none] for y2 & y3).

y2 item number on the ascii line of min (1/60 units) of y coordinate value in units of type c. USE THIS ITEM ONLY IF degrees min sec ARE SEPARATED BY BLANKS.

y3 item number on the *ascii* line of sec (1/3600 units) of y coordinate value in units of type **c**. USE THIS ITEM ONLY IF degrees min sec ARE SEPARATED BY BLANKS AND SECONDS ARE USED.

Example(s):

To **import moss** formatted polygon data, in UTM meters, from a file called **quill_creek** onto level **45** which will be called **soils level**, enter:

import moss quill_creek 45 2 2 soils level

To **import a .ras** scan file with a path **/u/scans/foggy_dew** and reduce the resolution by a linear factor of **4** with a threshhold of **.25**, enter:

import ras /u/scans/foggy_dew 4 .25

See also:

b_edit, crit_type, coord_type, export, level_clear, level_dup, level_icopy, level_snap

---

increment

Adds a point to the contour attribute profile.

increment x y

Adds a point to a contour attribute profile. This function is meaningful only when **atrbt_contur graphics** input mode is active. In all other modes, it behaves exactly like **continue**. All vectorized lines crossed from the last point on the profile up to the **increment** point will be assigned a numeric attribute in increments of the contour interval value (stored in the **header** file).

Options/arguments:

x X raster coordinate of point
y Y raster coordinate of point

Example(s):

To assign attributes to contour lines which are incrementally uphill from a previous **atrbt_contur, start, continue, or decrement** command, enter:

increment x y

See also:

atrbt_contur, continue, decrement, header, start

---

level_clear

Clears all data from a range of import/display levels. This command is needed when old data on a level needs to be updated with a new **import** file. It clears all data from a range of import/display levels. Valid level numbers are in the range **0 - 95** (levels **1 - 24** are sets created by the user and can be deleted with this command, or with the **set_clear** command).

level_clear level1 [level2]

Clears all data from levels **level1** through **level2**, inclusive.

Options/arguments:

level1 Level number, beginning of range, where valid entries are **0 - 95**, inclusive
level2 Level number, end of range, where valid entries are **0 - 95**, inclusive

NOTE: If **level1** is specified and **level2** is not, then **level2 = level1**.

Example(s):

To **clear** the (**import**) data in level **35**, enter:

level_clear 35

See also:

import, level_dup

---

level_clip

Clips the vector data in an **import** level so that no coordinate falls outside the 4 registered corners. The map raster MUST be registered. Only IMPORTED levels (**35-95**) may be clipped, NOT map vectors. The clipping is always done to match the STRAIGHT-LINE (not geodetic) frame.

NOTE: This command is normally used to clip TEMPLATE data so that only the portion of the template which overlays the map will be used for snapping. If this is not done and the template crosses the map rectangle, then **level_snap** MAY cause points on the map frame to move off of it.

level_clip level1 [level2]

Clears all data from levels **level1** through **level2**, inclusive.

Options/arguments:

level1 Level number, beginning of range, where valid entries are **35 - 95**, inclusive
level2 [optional] Level number, end of range, where valid entries are **0 - 95**, inclusive

NOTE: If **level1** is specified and **level2** is not, then **level2 = level1**.

Example(s):

To **clip** the (**import**) data in level **35**, enter:

level_clip 35

See also:

import, level_dup, level_clear

---

level_dup

Duplicates all of the data from an import/display level onto the base raster/map. Objects and/or attributes may be transferred from any level between **32** and **95**.

level_dup ln itc

Duplicates all data from level **ln** onto the raster/map according to the information type code (**itc**).

Options/arguments:

ln	Level number where valid entries are **32 - 95**, inclusive (**32** is an automatic level containing a geodetic frame, while the rest are levels for imported vector data)
itc	Information type code, where: **1** = vectors only **2** = attributes only **3** = both vectors and attributes **4** = copy point attributes from level to map segments **5** = copy point attributes from level to map polygons

NOTE: attribute(s) for each object will be added as new last attribute.

Example(s):

To draw the (curved) geodetic window on the raster, enter:

level_dup 32 1

To add all of the object lines from level **42**, along with their attributes, to the current map, enter:

level_dup 42 3

To add the attributes of the objects in level **95** to the current map objects under them, enter:

level_dup 95 2

See also:

import, level_clear, level_icopy, level_snap

---

level_icopy

Changes the input mode to **level_icopy**, an interactive, one-line-at-a-time method to selectively copy lines fom import/display levels onto the map raster. When this input mode is implemented, the **start** command will locate the object from a level (**35 - 95**). The object found will be highlighted, but not placed on the raster yet. After an object has been highlighted, the **continue** command will copy the object onto the raster. This provides a best-possible (in the raster world) copy of concurrent, or coincident, lines. Note that after the raster has been assembled, the vertices can be snapped to the vector object from the import level for an even closer match (see **level_snap**).

level_icopy l

Initiates the **level_icopy** input mode as described above. Only objects which are on the specified level will be chosen and highlighted for possible copy. The display of all levels can be individually controlled using the **graphics** command.

Options/arguments:

l: level number from which to copy objects (**32-95**).

Example(s):

To enter **level_icopy** input mode and copy a vector object from level **45** near raster location **211 860** enter:

level_icopy 45
start 211 860
continue xxx yyy (**xxx** & **yyy** may be anything)

See also:

import, level_clear, level_dup, level_snap

---

level_snap

This command will snap all vertices of the objects in a given set (**0 - 31**) to the nearest vertex or side of an object on the specified level (**32 - 95**) if it is within a given search radius distance.

level_snap s l1 l2 r u p

Snaps all vertices of objects in set **s** (**0 - 31**) to the vertices/sides of objects on levels **l1** to **l2** (**32 - 95**), if they are within radius **r**. The units of radius **r** are defined by **u**, the units code, according to the preference code **p**.

Options/arguments:

s Set number (**0 - 31**); (**0** = all objects)

l1 Lower bound level number (of levels to be snapped to) valid entries are **32 to 95**, inclusive.

l2 Upper bound level number (of levels to be snapped to) valid entries are **32 to 95**, inclusive.

r Snap radius; must be a positive number and may be decimal. The units of **r** are defined by the units code, **u**.

u Length units code, where:
0 = pixels
1 = map inches
2 = ground feet
3 = ground miles
4 = map centimeters
5 = ground meters
6 = ground kilometers

p Snapping Preference code, where:
1 = snap each vertex to nearest line or vertex in level range specified.
2 = snap each vertex preferentially to a vertex in level range specified, even if a line is closer.
3 = snap each vertex to nearest line or vertex in level range specified. Then, add vertices between snapped positions so that all vertices will be exactly matched. This option is normally the one used to SNAP TO A TEMPLATE or FRAME.
4 = snap each vertex only to a node or segment endpoint. This option is normally the one used to SNAP TO ADJACENT MAPS (EDGMATCHING).

Example(s):

To snap the map border polygon (which has already been placed in set **15**) vertices, within 3 pixels of the geodetic window, to the window, enter:

level_snap 15 32 32 3 0 3

The geodetic window is automatically generated when the map's geodetics are registered; it follows longitude and latitude lines which are usually curved on the map, in the most frequently used projections.

To snap all map edge nodes (which have already be put in set **6**) to the nodes on levels **88-91** (edgemaps) for edgematching, within a tolerance of **2.5** pixels enter:

level_snap 6 88 91 2.5 0 4

See also:

import, level_clear, level_dup, level_icopy, level_snap, all set_ and crit_ commands

---

Chapter 10 Map Manipulation

The **log** command is a very valuable LTPlus tool. It provides the user with information critical to the map data editing process. User-specified criteria can be applied to the current, active map and all resulting error types or set objects are then recorded in the map's **log** file. This file can then be interactively and graphically viewed, using the **goto** command.

log

Logs all selected error types or objects in the **log** file.

log ltype [s]

Logs all selected error types or objects in the active map's **log** file.

Options/arguments:

ltype	Type of action to be logged, where: **clear** = clears the **log** file **point** = **log** all isolated points **junct** = **log** all pixel intersections **spur** = **log** all spurs **hole** = **log** all isolated holes **set** = **log** all objects in set **s**
s	Set number used only with set type. Valid entries are **1 - 24**, inclusive.

Example(s):

To log all occurrences of objects which belong to set **13**, enter:

log set 13

See also:

goto

---

macro_begin

Activates the **macro** collection mode.

macro_begin name string

Enters **macro** collection mode. All subsequent commands are stored in the file name and no further action is taken on the current map until a **macro_end** is encountered.

Options/arguments:

name Name of **macro** file to create
string Description of macro, displayed by **help** command

Example(s):

To start building a macro, called **pmac**, for processing, enter:

macro_begin pmac Purges map files

See also:

help, macro_end, macro_hist

---

macro_end

Exits from **macro** building mode.

macro_end

Exits from the **macro** collection mode.

Options/arguments:

None.

Example(s):

To complete construction of a **macro**, enter:

macro_end

See also:

help, macro_begin, macro_hist

---

macro_hist

Constructs a **macro** file from a **history** file.

macro_hist mname r1 r2 string

Constructs a **macro** file from the current, active map's history file, where **mname** represents the new macro name, **r1** and **r2** are the inclusive record numbers of the history file which are to be clipped and inserted into the new macro, and **string** is a descriptor of the new macro file which can be displayed with the **help** command.

Options/arguments:

mname	Name of macro file to create
r1	Beginning record within history file
r2	Ending record within history file
string	Description of macro, displayed by the **help** command

---

Example(s):

To copy records **20-47** from the history file for the current, active map into a **macro** file called **cub_mac**, enter:

macro_hist cub_mac 20 47 Thin and assemble

See also:

help, macro_begin, macro_end

The raster **margin** is a border around the entire raster image of a known, or specified width. The **margin** command may be executed to change the width. The **margin** is important to the edgematching process. It is the area in which adjacent map vector data may be displayed. **IMPORTANT NOTE: WHEN THIS COMMAND IS INVOKED AGAINST AN ASSEMBLED OR ATTRIBUTED MAP FILE, ALL VECTORIZATION AND/OR ATTRIBUTES WILL BE LOST. THE MAP WILL REQUIRE RE-ASSEMBLY AND RE-ATTRIBUTING. THE REGISTRATION WILL NOT BE LOST.**

Note also that if a **b-raster** (original scan) exists, it will be replaced automatically by a copy of the (new sized) editable raster. This is necessary because the raster sizes need to be identical.

If a map is imported as a raster (i.e. scan file), with pre-determined raster dimensions, the **margin** is created when the raster corners are identified (register raster); it is the raster area exterior to a rectangular envelope containing the four raster corners. If a map is first registered geodetically (lat/lon, utm or state plane registration) and is not an imported raster file with pre-determined raster dimensions, the **margin** width is calculated to be **10%** of the overall map width and the margin area is presumed to be the raster area exterior to a rectangular envelope containing the four map corners.

margin

Sets the raster margin to a specified pixel width.

margin w

Forces the raster margin to a specified minimum pixel **width** which may be more or less than the current margin width.

Options/arguments:

w Positive integer of margin width in pixels

Example(s):

To set the margin width to **20** pixels, enter:

margin 20

See also:

erase_margin

Similar to the **corridor** command, the **mask** command generates a perimeter of a specified radius around all of the objects in a specified set and then fills (turns pixels on within) the new polygonal areas.

mask

Overlays objects in set **s** with solid mask of radius **r**.

mask s u r

Options/arguments:

s	Set number, where: **0 - 24** are valid entries
u	radius units code, where: **0** = pixels **1** = map inches **2** = ground feet **3** = ground miles **4** = map centimeters **5** = ground meters **6** = ground kilometers
r	Radius in pixels

Example(s):

To overlay the objects in set **12** with a solid mask with a radius of **16** pixels, enter:

mask 12 0 16

See also:

open_solids, corridors

---

menuset_get

Load another menuset for the current map.

menuset_get name

Activates the named menuset for the current, active map.

Options/arguments:

name Name of menuset

Example(s):

To get the set of menus under the name **cub_menu**, enter:

menuset_get cub_menu

See also:

funct_add, funct_del, menu_define, menuset_save

Merges the displayed and stored (**temp0**) raster in a special way. Each pixel of the display raster is examined. If the pixel is off and one or fewer adjacent pixels is on, then it is replaced with the pixel in the stored file. This has the affect of merging the displayed open areas with the stored raster.

NOTE: The use of this command IS NOT RECOMMENDED. It is very slow and rarely accomplishes the desired results. JPD

merge_fill

Merges the temporary and display raster in a special way.

merge_fill

Merges the temporary (**temp0**) and display raster in a special way.

Options/arguments:

None.

Example(s):

To merge the display and **temp0** stored raster, enter:

merge_fill

See also:

brstr_delete, disprstr_ers, disprstr_mrg, disprstr_get

Chapter 10 Map Manipulation

A map list is a file containing a list of maps. Every map list file is stored under a directory called **map_lists**, which lies along the default **data_path** defined in the **config** file. The **mlist_create** command allows the user to create a map list. The map list is then available for use with any command or macro.

mlist_create

Creates and opens a map list file.

mlist_create name, mname1 [,mname2] ... [,mnames]

Creates for access a map list file in the **map_lists** directory. At least one map name must be specified. Map names in the list may contain wildcards (**?**, *****).

Options/arguments:

name	Map list file name
mnamex	Name of map entry in map list file

Example(s):

To create a map list file called **cub** and store the maps **cub_pk, quill_pk, pug_pt, nut_shell** in that file, enter:

mlist_create cub, cub_pk, quill_pk, pug_pt, nut_shell

To use a wildcard character to store the map names **lides.tt** and **linetrace.hy**, but not the map name **liez.yt**, into a map list file called prog, enter:

mlist_create prog li?e*.*

See also:

mlist_exec

Once a map list has been created, it may be used with any command or sequence of commands (a macro) by executing the **mlist_exec** command and specifying the map list name and an action (command) or macro.

mlist_exec

Executes specified action on named map list.

mlist_exec [-s] maplist action

Executes action (macro or function) on all maps specified within maplist. The **maplist** file is found under the **map_lists** directory.

Options/arguments:

-s	(optional) DO NOT save each map after processing
maplist	Name of maplist
action	Name of macro or function to execute

Example(s):

To purge all of the maps listed in the map list file called dump_list, enter:

mlist_exec dump_list purge map

To execute a macro called **mop_up** on all maps in a map list file called **cub_maps**, enter:

mlist_exec cub_maps mop_up

See also:

mlist_create

---

more

Allows the user to view ascii text, and binary files.

more ftype [fname] [col]

Invokes the LTPlus **more** command (similar to UNIX **more**, but also displays non-text characters as the decimal ascii (3 digit) value compressed into a single character position. All non-text characters will be displayed after the first occurance of a non-text character OTHER THAN line-feed, carriage-return, tab, or form-feed is encountered. This includes both control (ascii 0-31) and extension (128-255) characters. Column width of output is controlled with the **col** parameter.

Options/arguments:

ftype File type, where the valid options are:

config	**config** file (fname not req.)
cursor_key	a **cursor_key** file
default_head	a default **header** file (**fname** not required)
diary	current map's diary file (**fname** not required)
dictionary	an attribute dictionary file
digmenu	a **digmenu** file
digitizer	a digitizer device file
export	a file in the **export** directory
header	current map's **header** file (**fname** not required)
import	a file in the **import** directory
macro	a macro file
map_list	a **maplist** file
menu_set	a **menuset** file
other	any complete file name
pen	a plotter penfile
tolerance	the **tolerance** file (**fname** not required)

fname File name; **not used** for the following **ftype** files:

config, default_head, diary, header, tolerance

The **fname** parameter should not be a full path name except when **ftype** is **other**. The path name for all other file types is determined by the **ftype** argument (i.e., if **ftype** is a macro and the macro name is **cub_mac**, the full path is presumed to be:

/u[sr]/ltp/macros/cub_mac

col The number of screen columns to use when displaying the file. Default = 80.

Example(s):

To display a file called **bear_mt.rlc** which is in the **import** directory, enter:

more import bear_mt.rlc

See also:

edit

This command will switch a raster into its negative image. All pixels that are *off* will be turned *on* and vice-versa. Some scanners output scanned data as negatives, some as positives. This command can reverse the image to suit LTPlus which wants the background bits off.

negative_img

Changes the raster image into its negative.

negative_img

Changes the raster image into its negative. There are no parameters. Using this command twice returns the raster to its original state.

Options/arguments:

None.

Example(s):

To change the raster image into its opposite, enter:

negative_img

See also:

reduce_res, reflect, rotate

---

next_step

Prompts the user with the next logical action on a map.

next_step

Checks the **job_stats** and **register** files to determine current map status and prompts the user with the next logical processing step for the current map. This command is very useful for people who are learning to use LTPlus and for experienced users who need a reminder or have forgotten just what state a map was in when they last accessed it. This command is automatically invoked when a map is acquired.

Options/arguments:

None.

Example(s):

To be informed of a next logical processing step for the current map, enter:

next_step

See also:

help

The **open_solids** command turns *off*, or erases, all of the interior pixels of solid areas within the raster. The result is a clear polygonal area. The map need not be assembled.

open_solids

Erases interior pixels from solid areas.

open_solids

Erases the interior pixels of all soild areas within the raster. Assembly is not required.

Options/arguments:

None.

Example(s):

To erase all of the interior pixels within solid blocks of pixels within the current, active raster, enter:

open_solids

See also:

fill_holes, mask, corridors

The power of set operations can be experienced with the plotting operations via a **pen** file. This file may be generated by the user with a **p_create** function which permits (or prompts for) assignment of pen number(s) to set(s). With a multi-pen plotter, a user may utilize colors and/or tip sizes to represent various sets of information. Creating a **pen** file with the **p_create** command provides tremendous plotting flexibility.

p_create

Creates a **pen** file with assignments by set.

p_create file argn

Creates the specified file and writes to that file a **32**-digit number. Each digit of that number contains a value in the range of **0 - 8**. That value represents the pen number assigned to the set number which is defined by the position of the digit within the argument (increasing left to right, from **0 to 31**). If there are less than **32** digits specified in the argument, then sets not shown are assumed to be assigned to pen **0** (no plot). The named **pen** file is stored in the **/u[sr]/ltp/pens** directory.

Options/arguments:

file	Name of **pen** file
argn	Position dependent pen-to-set assignment

Example(s):

To create a **pen** file called **plot_a** with the pen assignments of set4:pen3, s7:p2, s8:p4, s9:p8, s10:p1, s17:p7, s28:p3, s31:p4, and all other sets assigned to pen0, enter:

p_create plot_a 00003002481000000700000000003004

To create a **pen** file named **plot1b** with pen assignments of set2:pen3, s3:p4, s5:p2, s6:p2, s7:p2, s8:p1, s10:p6, and all other sets assigned to pen0, enter:

p_create plot1b 00340222106

See also:

config, plot

---

pause

Displays a message and halts program for a number of seconds.

pause s text

This command displays a message in the lower (command) portion of the screen in black on a light red background. The message is centered on the second line of a three line background. The normal message display and any other text is temporarily removed. After the input number of seconds, the display is returned to normal and the program continues. This command is designed for use in demonstration macros to describe what the macro is doing.

Options/arguments:

s	A (whole) number of seconds (0 or greater)
text	Any text string up to 80 characters

Example(s):

To tell the viewer about a **zoom** display of the raster and **continue** with the macro after a 5 second **pause**, enter:

pause 5 This is what the raster looks like close-up

See also:

wait

Typically used with contour layer processing, the **pers_plot** command will produce a false perspective plot, which is useful for editing contour data and for manually un-labeling contours which are incorrect.

pers_plot

Generates a false perspective screen plot, and allow manual un-labeling of contours.

pers_plot d e v s

Creates a false perspective plot (x, y, z), with an orientation direction **d**, elevation sensitivity **e**, y-vertical factor **v**, and skew **s**. The perspective view will stay on the screen until the user hits the return key. While the perspective (wire-frame) view is displayed (before the return key is pressed), the user may move the cursor with the mouse, and using the **'start'** or **'continue'** keys indicate a contour line to be un-labeled. This allows the marking or un-labeling of labeling errors.

NOTE: A maximum of 100 contours can be un-labeled on a single perspective plot in this manner.

Options/arguments:

d Direction map is viewed, where:
t = view from the top
r = view from the right
b = view from the bottom
l = view from the left
Normally, the easiest views to visualize are those that look UP THE MAIN DRAINAGES. If the main drainages on the portion of the map in view flow off of the left side of the screen, the try a view direction of **l** (the letter L).

e Elevation sensivity factor.
1.0 = a 1:1 ratio of screen z and x scales. A factor of 5 would mean that the z (elevation) values would be exagurated by a factor of 5 times (relative to x values). Normally, factors ranging from **1.0** for the steepest ground to **5.0** for gentle ground are used.

v y-vertical factor.
This value indicates how much to tip the 3-D box up on the screen. The effect of a small value (say 0.1) is to view the terrain from the ground. A large value makes it appear that you are viewing it overhead. Normally, steep ground is viewed with a higher factor, say 0.6 or 0.8. Gentle ground is best viewed with a lower factor (say 0.2-0.3) to accentuate the modest hills/valleys.

s Skew ratio, rear-front offset : x scale.
Positive values skew the rear of the display to the right. Negative values move the rear of the displayed area to the left. A value of 0.5 would move the bottom rear box line 1/5 of its length to the right of the bottom front line.

NOTE: Normally, the easiest views to visualize are those that look UP THE MAIN DRAINAGES. After choosing the view from the correct side of the picture, use this (skew) parameter to 'fine tune' the view to look directly up the drainage. For example, if the screen contours show the main drainage to drain out the left of the screen, and up the drainage goes to the right and up accross the screen, then the direction would be l, and the skew might be 0.3 for a first try at a good view.

Example(s):

To generate a false perspective plot of the current display, which is of fairly steep slopes, with the main drainage running off the top of the screen, and is in a diagonal from the lower left corner, enter:

pers_plot t 1.8 0.6 -0.4

See also:

dem_view

Similar to the **area_plot** command, the **plot** function generates a plot of the current, active map. The user may designate the plotter, label type and size and optional pen:set assignments.

plot

Plots the current map on the specified device.

plot [-s] [pname] dev scale ltype [sizen] [fname] [-L]

This function will cause the current map to be plotted to plotting device **dev**, at a scale of **1**:**scale**, with label type **ltype** at character size **sizen**. Setting the **-s** option will initiate a selective plot, and will require an associated **pen** file name **pname** from which **set:pen** correspondence may be derived. No option defaults to plotting all objects in pen 1. In order to utilize a **pen** file, the **-s** option must be invoked, even if all objects are to be plotted (select 0 for all objects).

When creating an ascii plot file (**dev=3**), plotter origin is assumed to be **center** unless the **-L** argument is specified which indicates a lower-left origin. In all other cases, (**dev!=3**), plot views the **config** file entry for the designated plotter type (**plt1_type** or **plt2_type**); if the equivalency string begins with the letter **L**, then the origin will be assumed to be lower-left. If the string value begins with any other character, or has no entry at all, then the plot origin will be assumed to be the center.

To abort a **plot** command (once invoked), press the **ESC** key; the plotting may continue for a short period, as the plotter buffer will continue to function, even though the actual function has been stopped.

The plot function also includes a feature which will automatically roll the plotter media to a new frame at the end of the current plot. This allows the user to batch several plots without concern about being present to forward the media for the next plot.

Options/arguments:

-s	Requests a selective plot and requires **pname**
pname	Pen assignment file name
dev	Plotter device code, where: **0** = screen **1** = plotter1 **2** = plotter2 **3** = ascii file
scale	Plot scale denominator (24000 for 1:24000)
ltype	Label type, where: **0** = none **1** = record number **2** = serial number **3** = first attribute only **4** = all attributes (concatenated)
plot	Label size where value is character size in mils
fname	Ascii output file name; used only when **dev=3**
-L	Indicates plotter device has lower-left origin (i.e., HP7545)

Example(s):

To generate a plot of the active map on plotting device 2**,** using a scale of **1:24000**, with the first attribute labels displayed at 0.10 inches, for all objects, enter:

plot 2 24000 3 100

To selectively plot objects from the current map onto plotting device 1**,** with a scale of **1:36000**, with record numbers labeled at 0.15 inches in size, using the pen assignments defined in the file **pmap**, enter:

plot -s pmap 1 36000 1 150

See also:

header, mlist_create, plot_ascii, p_create, all set_ functions

---

Similar to the **area_plot** command, the **plot_ascii** function generates a plot of an existing HP-GL ascii file created by a previous **plot** command.

plot_ascii

Plots an *ascii* HP-GL file to the specified plotter.

plot_ascii dev fname

This function will cause the named ascii file specified by the **fname** argument to be plotted on the plotter device specified by **dev**.

Options/arguments:

dev	Plotter device type, where: **0** = plotter1 **1** = plotter2
fname	**ascii input** file name; if the name is a relative pathname, it is assumed to be in the **/u[sr]/tmp** directory.

Example(s):

To generate a plot of the **ascii plot** file **wogplot**, which is located in **/u/tmp**, to the plotter **1** device, enter:

plot_ascii 1 wogplot
OR
plot_ascii 1 /u/tmp/wogplot

See also:

area_plot, plot

This command **resets** the internal value used to replace the **$++** variable attribute string, and the string length of the **$++** string.

plus_plus

Resets the internal **$++** value.

plus_plus [n=xx] [l=yy]

This command **resets** the initial value used to replace the **$++** attribute variable. When **$++** is imbedded within an attribute string, it will be replaced with a **yy** character string beginning with **xx** (which is a whole number >= 0). Each time an attribute containing the **$++** string is assigned, the **xx** value is incremented by 1.

Options/arguments:

n=xx xx will be the next **$++** value to be assigned (optional).
l=yy yy will be the character length of the xx value inserted: if the xx value is less than yy

If yy = 0: the length of the xx string will vary, being only long enough to contain the xx value (e.g. if xx=482 the string would be: 482).

If yy > 0: the length of the xx string will be yy characters (1-30).

If yy is longer than needed for xx, xx will be 0 filled on the left. If yy is not long enough, xx will be truncated on the left (e.g. if xx=482 & yy=5, then sting is 00482 if xx=482 & yy=2 then string is 82).

NOTE: If neither option is specified, you will be prompted for both.

Example(s):

To assign incremental attributes set **5-00101** through set **5-00152** as a first attribute to all **52** objects in set **5**, enter:

plus_plus n=101 1=5
atrbt_set 5 1 set5-$++

To begin assigning the attribute **'unit5-stand002'** as a second attribute by digitizing polygon attributes, enter:

plus_plus n=2 1=3
atrbt_dig 2 2 unit5-stand$++

See also:

atrbt_dig, atrbt_rec, atrbt_serial, atrbt_line

---

prompt

Gathers user input for macro execution.

prompt n question

or

prompt -n directory question

Questions the user and puts the response in macro variable **n**, or questions the user and lets him/her choose a file from a specified directory and puts the response into macro variable n.

At any later time during that session, the response may be used in a macro (or from the command line) by placing **$promptN** in a string (where N is the number of the macro variable).

All macro variables (**$prompt01** through **$prompt99**) remain as filled until LTPlus is exited. At the beginning of each LTPlus session (**ltpgo**) all are set to null/blank. They are NOT reset when a new map is acquired.

_____ *

Options/arguments:

n	macro variable number to be filled. range 01 - 99. (must be 2 digits: 01 02 03 04 05 06 07 08 09 10 etc.)
directory	name of a directory from which a file is to be chosen.
question	any string up to 75 characters long (but total prompt command may not exceed 95 characters). This string will be displayed as a prompt (number 400) when the program expects the user to enter an answer or choose a file.

_____

Example(s) (Normally, these commands would be in a macro):

1. Get user input for plot scale only and execute plot commad:

   prompt 01 Enter plot scale.
   plot 1 $prompt01 3 100

2. Have the user choose a file from the **/u/junk directory** & have him/her edit the file.

   prompt -11 /u/junk Choose a file to be edited.
   edit other /u/junk/$prompt11

3. Let user choose a file from the import directory & import it as RLC scanned data with a reduction factor of 1.

   prompt -42 $config.import Choose an RLC file to be imported.
   import rlc $prompt42 1 1

_____

See also:

_____

Unwanted files may be removed from the system with the **purge** command. Keep in mind that this is a pretty *permanent* type of action. There is no wastebasket from which the file may later be retrieved.

purge

Purges specified data file type/name.

purge dtype name

Permits the user to delete a specific (**dtype**) type of data file named **name**.

Options/arguments:

dtype Data file type, including:
cursor_keys
import
dictionary
macro
digitizer
map
digmenu
maplist
export
menuset
help
penfile

name Mapname (actually a directory) or file name. A mapname will recursively delete all files associated with that map

---

Example(s):

To delete a **data dictionary** file named **cub_d**, enter:

purge dictionary cub_d

To delete the files associated with a map called **pug_point**, enter:

purge map pug_point

See also:

acquire, catalog, create

---

query

Activates the **query** mode for locating objects.

query otype

Sets the active mode to **query**. The object of the **query** is determined by the argument **otype**. Current maptype objects are the default. Once in **query** mode, queries are made using the **continue** command. When in **query** mode, the **start** or **continue** command will check for a nearby object (of the specified type on the specified level) and display information about it on the command portion of the screen. This information includes the object number and the first two attributes. For map objects (not level 32-95 objects) it also displays the length/perimeter and area of the object.

Options/arguments:

otype Object type code, where:
0 = any node
1 = any segment
2 = any polygon
xx0 = imported node on level **xx**
xx1 = imported segment on level **xx**
xx2 = imported polygon on level **xx**

Example(s):

To locate a polygon at a position identified as **x=456** and **y=777** (in currently displayed pixel units), enter:

query 2
continue 456 777

To **query** a node on level 42 near location x=458, y=1121 enter:

query 2
start 456 1121

Note that the actual x,y positions are usually automatically generated by positioning the mouse/screen cursor over the desired object and pressing the button which corresponds to the **continue** command.

See also:

coord_type, crit_type

Chapter 5 Map Information Operations

There are 18 files associated with every map created by LTPlus. For the most part, those files are of little direct interest to the user. Various commands access and massage the data within some of those files and then present the raw data in a format which is palatable to the user. However, LTPlus allows display of any of those data files, for those brave enough to ask. The **raw_print** command allows the user to view any raw data file in its entirety or any specified portion of selected raw data files. It also allows the user to view the current program status of file pointer currently open, the last 128 displayed command lines, the current malloc space, and digitizer variables.

raw_print

Prints all or a portion of named **raw data** file.

raw_print ftype [b e] [-d] [-f filename]

Prints the entire **raw data** file or a portion specified for the current active map, or the last 128 lines of command line history, or a list of all currently open files, or a list of all RAM currently malloced by the program.

Options/arguments:

ftype Type of raw data file, including:
diary
history
header
job_stats
objects
errors
register
seg_list
index_array
coord_list
index_list
attributes
v_attributes
v_objects
v_coordlist
cmd_line_hist
open_files
malloc_state
dig_variables

b Optional beginning record, required on fixed record length (non-ascii) files.

e Optional ending record, required on fixed record length (non-ascii) files.

-d Display output on screen only. No hardcopy.

-f filename Print to a file named as specified in filename.

Example(s):

To print the entire register raw data file, enter:

raw_print register 0 99999

To display records **450-871** of the objects **raw data** file without printing the file, enter:

raw_print objects 450 871 -d

To print the values of the internal digitizer variables to a file called **digvar** (in the export directory) enter (there are no record numbers for the **dig_variables** data):

raw_print dig_variables -f digvar

See also:

edit, more

---

recall

Activates the recall mode.

recall shape [b]

Activates the graphics input **recall** mode. This mode will read the raster as stored in the temp file the last time and return all pixels in the chosen shape to their former status (*on* or *off*). Its used is to repair operator mistakes, similar to the **undo** command. Use **start** to begin recalling a shape and **continue** for all subsequent vertices for that shape. Then, use **start** to begin another object. Note that the current mode/action is displayed in the upper-left window of the screen.

Options/arguments:

shape	Shape type, including: **point** (single pixel) **line** (pixel unit width) **fat_line** (3-pixel width) **arectangle** (area; includes center area) **brectangle** (border; does not incl center) **wedge** (walking triangle) **separate** (same area as separate mode) **connect** (same area as connect mode)
b	(optional) if present: recall from the **braster** (original scan), not the edited raster.

Example(s):

To invoke the **recall fat_line** draw mode starting with a position defined as **x=456** and **y=222**, enter:

recall fat_line
start 456 222

To continue the **recall fat_line** mode at a position defined as **x=488** and **y=225**, enter:

continue 488 225

To remain in the **recall fat_line** mode while moving, but not recalling, to a new position defined as **x=500** and **y=236**, enter:

start 500 236

To invoke the **recall arectangle** mode starting at the position defined as **x=790** and **y=345** from the original scan (**braster**), enter:

recall arectangle b
start 790 345

To re-position the starting position of the rectangle to **x=690** and **y=345**, enter (prior to any **continue** commands):

start 690 345

To complete the **rectangle recall** action to encompass the rectangular area bounded by **690,345** and **700,390** (lower-left and upper-right corners; any two, diagonally-opposed positions will suffice), enter:

continue 700 390

Note that the **recall brectangle** command recalls only a unit_width perimeter of the defined rectangular area. The **recall arectangle** command recalls the entire rectangular area, not just the perimeter.

To exit the **recall** mode, enter:

clear_mode

or enter a new graphics input mode.

See also:

clear_mode, continue, erase, start

The raster display may deteriorate during the editing process; screen cursor movement across lighted (*on*) pixels may inadvertantly turn them *off*. To recalculate and reconstruct the raster file and update its screen appearance, execute the **redisp_map** command.

redisp_map

Re-displays the reconstructed raster file.

redisp_map

Rebuilds the display file and re-displays the map to reflect the current raster exactly. Note that this affects only the map resolution display.

Options/arguments:

None.

Example(s):

To reconstruct and re-display the current raster file, enter:

redisp_map

See also:

graphics

The **reduce_res** command allows a raster to be reduced in size by any positive whole number **factor**. The reduction is done by combining a square block of pixels on the original raster into a single pixel which represents a larger area on the final raster. The user can specify a **threshhold**. The **threshhold** indicates how many original pixels (in one square block) must be on for the single final pixel, which represents that block, to be on. Note that all vector data IS ALSO REDUCED & IMMEDIATELY AVAILABLE.

This command is meant to be used just after importing scanned data, or (on high-resolution contour scans) immediately after and before attributing. The purpose of this command is to compensate for scanners which have a fixed resolution, or to scan and assemble at high resolution (to maintain line separation) and attribute at lower resolution (for faster attributing & smaller file size).

NOTE: If a **b-raster** (original scan) is present, it **too** will be reduced in resolution by the same factor as the editable raster.

reduce_res

Reduces the resolution of the active raster.

reduce_res n [t]

Reduces the resolution of the raster by a factor of **n**. If specified, the optional parameter **t** will be used as the **threshhold**; if not specified, **t** is assumed to be **(n*n)*0.40**.

Options/arguments:

n	Reduction factor; any whole number **>1**
t	**Threshhold**; any whole number between **1** and **n*n**

Example(s):

To reduce the current map's raster by a factor of **4** with a **threshhold** of **12**, enter:

reduce_res 4 12

See also:

negative_img, reflect, rotate

A raster image may be flipped, or reflected, from right to left or from top to bottom. The **reflect** command will produce a mirror image along the designated axis; the default will reflect along a horizontal axis (top to bottom), or the user may specify the **-h** option which will cause the raster to be reflected along a vertical axis (left to right). This command is particularly useful for correcting a map which was scanned improperly.

IMPORTANT NOTE: Invoking the **reflect** command against an assembled (vectorized) or attributed map is *very dangerous!* The vectorization and all attributes will be *permanently* LOST. The map will need to be re-assembled and re-attributed. Map registration will also be LOST. The map will require total re-registration (both raster and geodetic).

Note also that if a **b-raster** (original scan) exists, it will be replaced automatically by a copy of the (new sized) editable raster. This is necessary because the raster sizes need to be identical.

reflect

Reflects the active raster.

reflect [-h]

Reflects, by default, from top to bottom. The **-h** option will reflect from left to right.

Options/arguments:

-h Reflects left to right (horizontally), instead of top to bottom

---

Example(s):

To reflect the active map, left to right, enter:

reflect -h

See also:

negative_img, reduce_res, rotate

Every map processed in LTPlus must be **registered**. An imported scan file must be registered to the raster. A manually digitized map must be **registered** to the digitizer tablet. In most instances, it is also desirable to link the map to the real world by identifying control points for which ground coordinates, in some system or another, are known. Whenever possible, *all four corners* should serve as the map registration control. Geodetic coordinates must be entered as latitude and longitude. State plane feet and UTM meters must be entered as X (Easting) and Y (Northing) values. The user may also supply registration data for a local coordinate system (or several), if required. If **map_ht** and **map_wd** are preset in the **header** file and the map is rectangular with respect to lat/lon, then the geodetic registration is satisfied with a single corner entry. It is important to note that the control points do not need to properly bound the mapping area; that is, the control points can be inside the actual map limits. However, the *best* control will be generated when the control points are at the extreme outside of the desired map area.

register

Performs map or menu registration.

register [-d] [-p] c [i1=x,y]...[i4=x,y] [type=str]

This command performs map or menu registration. The -d option displays the current registration, while the -p option prints the current corner registration to the default primary system printer (see config file entry). The coordinate type (c) must be identified. Corner coordinates for the four corner positions are entered via the digitizer, mouse and/or keyboard. For geodetic registration, only a single corner is required if **map_ht** and **map_wd** are set properly in the **header** file. For geodetic registration, use [**i1=x,y**]...etc.

Options/arguments:

-d	Display registration values of active map
-p	Print registration values of active map
c	coordinate type code: **0** = raster **1** = geotetic (lat/lon) **2** = utm **3** = state plane **4** = digitizer 1 **5** = digitizer 2 **6**= map inches **7** = last digitizer1 pts. **8** = - reserved **9** = - reserved **10 - 19** = - user defined coord systems **20 - 29** = - reserved for system use
i1..i4	Corner identifier, including: SW NW NE SE **= x,y** Corner coordinates, in appropriate units
type	Description for user-defined system = **str** text string dexcription

Example(s):

To display the corner registration for the current, active map on the primary default printer, enter:

register -p

See also:

reg_copy

Any map which is located along the default **maps_path** path (see **config**) may be **renamed**. This is not the same as making a copy. This is equivalent to a UNIX/Xenix **mv** command.

rename

Renames a map.

rename name1 name2

Permits the user to **rename** an *existing* map (**name1**) to a *new* name (**name2**). The existing map must be along the default data path.

Options/arguments:

name1 Old file name
name2 New file name

Example(s):

To rename a map called **cub_creek** to **cub_cr**, enter:

rename cub_creek cub_cr

See also:

acquire, create, purge

The **report** function allows the user to specify the output device (screen, printer or *ascii* file), the units of length and area, the object type for both primary and secondary ordering of information, and a set number (**0** indicates all objects in the file).

report

Creates a **report** based on user-defined parameters.

report dev u area r1 r2 s [fname]

Creates a **report** based on the supplied parameters.

Options/arguments:

dev	Printer/display device, where: **0** = screen **1** = printer **2** = *ascii* file (**fname** required)
u	Length units code, where: **0** = pixels **1** = map inches **2** = ground feet **3** = ground miles **4** = map centimeters **5** = ground meters **6** = ground kilometers
area	Area units code, where: **0** = pixels **1** = square inches (map) **2** = acres **3** = square miles **4** = square cm (map) **5** = square meters **6** = square kilometers
r1	Primary order code, where: **0** = no sort **1** = object type **2** = record number **3** = serial number **4** = object length/perimeter **5** = area **6** = attribute
r2	Secondary order code (see **r1** table, above)
s	Set number, where: **0** = all objects **1 - 24** = specific set number
fname	*ascii* output file name (required only when **dev = 3**)

Example(s):

To generate a **report** to the screen which displays the length values in feet and the area values in acres, and in which the objects are primarily sorted by object type and secondarily sorted by record number, and for which the objects are selected from those in set **6**, enter:

report 0 2 2 1 2 6

See also:

crit_report, set_report

To recover map data stored offline with the backup command, use the **retrieve** command. The **bkup_path** device must be correct, as defined in the **config** file. If a listing of what files exist on the archive is desired, use the **-t** argument. No files are extracted when the **-t** option is exercised.

retrieve

Copies backup files back to the active map directory.

retrieve [+/-t]

Extracts all files on the archive file found at the **bkup_path** device. If the **-t** option is included, a listing of all files is produced and no files are actually extracted.

Options/arguments:

-t Produce a listing of files on backup medium only. No files are extracted.

Example(s):

To retrieve a map called **cub_peak** from its device specified by **bkup_path** in the **config** file, enter:

retrieve +t

See also:

config, backup

A raster may be turned, or rotated, about its center, in increments of 90 degrees (1/4 turn) by executing the **rotate** command. If no degree value is entered, the rotation is assumed to be 90 degrees. Counterclockwise rotation may be accomplished by entering a negative (**-**) value. Rotating a raw (unassembled) raster will require less time than rotating assembled data.

IMPORTANT NOTE: Invoking the **reflect** command against an assembled (vectorized) or attributed map is *very dangerous!* The vectorization and all attributes will be permanently LOST. The map will need to be re-assembled and re-attributed. Map registration will also be LOST. The map will require total re-registration (both raster and geodetic).

Note also that if a **b-raster** (original scan) exists, it will be replaced automatically by a copy of the (new sized) editable raster. This is necessary because the raster sizes need to be identical.

rotate

Rotates the active raster.

rotate d

Rotates the raster 90 degrees (default) or in increments of 90 degrees, clockwise. Note that a raw raster will rotate in less time than an assembled raster.

Options/arguments:

d Degrees, where valid entries are **90, 180, 270** or the minus values of same, for a counterclockwise rotation

---

Example(s):

To rotate the raster 180 degrees clockwise, enter:

rotate 180

See also:

negative_img, reduce_res, reflect

To avoid the *pain* and *misery* of lost map data (which is most often caused by an inappropriate or bizarre action by the user!), use the **save** command to store (and re-store) the active map's data files. You will be prompted for this action when creating or acquiring another map or exiting LTPlus. It is a good idea to be pro-active and initiate the **save** command every 1/4 to 1/2 hour. The active map will be saved under the map name directory which will be found along the default **maps_path** as defined in the **config** file. The current map may be saved to a second map name, thereby retaining the original map without the changes.

save

Saves the active map.

save [fname]

Saves the current map under the maps directory on the default data path device. The map's acquired or created name is used unless the optional **fname** option specifies a new map name.

Options/arguments:

fname Optional filename

Example(s):

To store the active map, enter:

save

See also:

acquire, backup, create, purge, retrieve

Chapter 9 Input Modes

Note: Discussion in Chapter 9 not currently included, it will be included in the final User's Guide.

seg_break

This input mode allows the user to break one segment (line) of a line or line/point type map into two lines. When in this input mode, the **start** command will identify (highlight between) the two verticies which bound the closest point of the closest segment. If the **continue** command is then used, the segment will be broken between these verticies. This input mode will *not* work on polygon type maps.

seg_break

Changes the **input** mode to **seg_break**.

Options/Arguments:

None.

Example(s):

To begin **seg_break** input mode and break a segment at point **250 411** enter:

seg_break
start 250 41
continue 250 411

See also:

level_snap

Activates the **separate** mode for separating touching lines.

separate

Activates the **separate** mode for separating touching lines.

separate

Activates the **separate** mode. This function turns pixels *off* in a straight, unit-width line between x,y coordinate pairs provided by the **start** command and the **continue** command (or the **increment** and **decrement** commands, which behave the same as the **continue** command).

Options/arguments:

None.

Example(s):

To **separate** the pixels lying between a position identified as **x=79** and **y=90** and another position defined as **x=76** and **y=100**, enter:

separate
start 79 80
continue 76 100

See also:

connect

The **set_comp** command will determine the complement of named set **s1** (i.e., all of the objects which are *not* in set **s1**) and store them in set **s2**.

set_comp

Stores the complement of **s1** into **s2**.

set_comp s1 s2 [str]

Takes the complement of a set and places the result in **s2**, and allows the user to tag the new set with a descriptor **string**. Any data stored previously in set **s2** will be overwritten.

The set created (**s2**) will be highlighted automatically, and the number of objects, total length/perimeter in miles, and total area in acres will be displayed on the command line area.

Options/arguments:

s1	Input set
s2	Destination set; complement of set **s1**
[str]	Set descriptor (user input), optional

Example(s):

To take the complement of a set called **cub_set1** and store it in a set called **cub_set2**, enter:

set_comp cub_set1 cub_set2

See also:

set_clear, set_exclude, set_intrsec, set_union

---

set_dig

Sets mode to assign objects to a given set.

set_dig set otype desc

Sets the **input** mode to digitize objects of a given **type** into (or out of) a specified **set**. To use this mode, the map must be assembled. The **set** being digitized will be highlighted automatically.

Options/arguments:

set	set number: **1 to 24** (add object to set) **-1 to -24** (remove object from set)
otype	Object type code, where: **0** = node **1** = segment **2** = polygon
desc	set description (text)

Examples:

To add the polygon around **255 1214** to set **10** and describe set **10** as 'another stupid set' enter:

set_dig 10 2 another stupid set
start 255 1214

To remove the line near **816 591** from set **24** and change the set description to **'ha ha, I fooled you!'** enter:

set_dig -24 1 ha ha, I fooled you!
continue 816 591

See also:

crit_exec, set_ commands

---

set_erase

Erases the raster under a set of vector objects

set_erase s

Performs an **erase_obj** operation on all objects in set **s**. Only the raster is changed. Assembly is required to use this command.

Options/arguments:

s Set number:
0-31

Example(s):

To erase the raster under all vector objects in set 15 enter:

set_erase 15

See also:

erase_obj, erase_seg, recall, undo, header

The **set_exclude** command compares two sets, **s1** and **s2**, and determines which objects are in set **s1** but not in set **s2** and stores them in set **s3**.

set_exclude

Stores objects in **s1**, but not in **s2**, into **s3**.

set_exclude s1 s2 s3 [str]

Finds all elements of set **s1** which are not in set **s2** and stores the result in set **s3**. Any data already in set **s3** will be overwritten.

The set created (**s3**) will be highlighted automatically, and the number of objects, total length/perimeter in miles, and total area in acres will be displayed on the command line area.

Options/arguments:

s1	Input set [result set elements must be in **s1**]
s2	Input set [result set elements must not be in **s2**]
s3	Destination set
[str]	Set descriptor (user input), optional

Example(s):

To perform an exclusion on set **cub_s1** with **cub_s2** and store the results in **cub_s3**, enter:

set_exclude cub_set1 cub_set2 cub_s3

See also:

set_clear, set_comp, set_intrsec, set_union

The **set_intrsec** command stores all objects which are shared by sets **s1** and **s2** into set **s3**.

set_intrsec

Intersects two sets and stores to third set.

set_intrsec s1 s2 s3 [str]

Performs the intersection of 2 sets and places the result in **s3**. Any data already in **s3** will be overwritten.

The set created (**s3**) will be highlighted automatically, and the number of objects, total length/perimeter in miles, and total area in acres will be displayed on the command line area.

Options/arguments:

s1	Input set
s2	Input set
s3	Destination set
[**str**]	Set descriptor (user input), optional

Example(s):

To intersect **cub_set** with **pug_set** and store the resulting objects in **cub_pug**, enter:

set_intrsec cub_set pug_set cub_pug

See also:

set_clear, set_comp, set_exclude, set_union

Reports the names and number of elements in all sets. The report can be written to the screen, the printer, or a file.

set_report

Write a report of number of elements and names of all sets.

set_report p [fname]

Reports on all sets. Report is written to the device specified by **p**.

Options/arguments:

p	print dev: **0** = screen **1** = printer **2** = ascii file
fname	Name of file (used only when **p = 2**)

Example(s):

To display current set information on the screen enter:

set_report 0

To write a **set_report** to a file called **kettle**, enter:

set_report 2 kettle

See also:

all set_ and crit_ functions

The objects in sets **s1** and **s2** are combined and stored in set **s3** with the **set_union** command.

set_union

Performs union of two sets and stores in third.

set_union s1 s2 s3 [str]

Performs the union of sets **s1** and **s2** and stores the resulting objects in set **s3**. Any data already stored in set **s3** will be overwritten.

The set created (**s3**) will be highlighted automatically, and the number of objects, total length/perimeter in miles, and total area in acres will be displayed on the command line area.

Options/arguments:

s1	Input set
s2	Input set
s3	Destination set
[str]	Set descriptor (user input), optional

Example(s):

To perform a union of **cub_1** with **cub_2** and store the resulting objects in **cub_end**, enter:

set_union cub_1 cub_2 cub_end

See also:

set_clear, set_comp, set_exclude, set_intrsec

Chapter 3 System Management

UNIX is a multi-user, multi-tasking environment. LTPlus provides a tool which allows the user to escape temporarily from the program, perform other operations, and return to the exact point of departure with no detrimental effects. Executing the **shell** command compels LTPlus to suspend interactive operations temporarily and open another shell, or partition (the **csh** shell in this case), in which the user may interactively function. To leave the shell and return to the initial point of departure in LTPlus, press **Ctl-D**.

shell

Exit LTPlus temporarily.

shell

Exit from LTPlus temporarily and open a **C shell** in the current path. **Ctl-D** closes the shell, and returns to LTPlus.

Options/arguments:

None

Example(s):

To leave LTPlus for a temporary **C shell**, enter:

shell

See also:

set_clear, set_exclude, set_intrsec, set_union

---

start

Initiates current mode at position provided.

start x y

Initiates the current mode starting at a location defined by the x,y coordinates (in the current display units).

Note: The current mode/action is displayed in the upper-left window.

Options/arguments:

x **New x coordinate**
y **New y coordinate**

Note: Normally the **start, continue** (and **increment, decrement**) commands are entered as button presses on the mouse or digitizer cursor. In these cases the input coordinates will be automatically supplied. If the command has been entered from the keyboard without coordinate arguments, you will be prompted to enter a position with the cursor (mouse/digitizer).

Example(s):

To move to a position defined as **x=500** and **y=510**, in the current mode, enter:

start 500 510

See also:

continue, increment, decrement

The **header** file is comprised of data about the map which is supplied by the user. In contrast, executing the **status** command will produce a summary of the internal status flags maintained by the LTPlus system. The command usage is as follows:

status

View the **status** of the active map.

status [-p]

View the **status** of the active map. A **-p** option will route the information to the default printer.

Options/arguments:

-p **Route to printer**

Example(s):

To print the **status** of the active map, enter:

status -p

The **status** command accesses a **job_stats bin** file, located under the directory **/u[sr]/ltp/maps/(map_name)**, and displays the conditions listed below:

Job_stats condition:
0. file version number (for file undates)
1. maximum raster column number (lt col is 0)
2. maximum raster row number (bottom is 0)
3. Object file record # of first node
4. Object file record # of first island node
5. Object file record **#** of first segment
6. Object file record **#** of first island set.
7. Object file record **#** of first polygon
8. maximum serial number used
9. format code for raster file:
   0 = Bit Map format
   1 = Run Length Encoded format
10. number of records in the **objects** file
11. number of records in the **coord_list** file
12. number of records in the **seg_list** file
13. number of records in the **attributes** file
14. number of records in the **errors** file
15. number of records in the **index_list** file
16. number of records in the **v_attribute**s file
17. number of records in the **v_objects** file
18. number of records in the **v_coord_list** file
19. record number of last entry in **history** file.

/* the following are 0/1 (no/yes) flags */

20. type data has been imported.
21. type data has been imported.
22. type data has been imported.
23. type data has been imported.
24. type data has been imported.
25. type data has been imported.
26. type data has been imported.
27. type data has been imported.
28. type data has been imported.
29. type data has been imported.
30. type data has been imported.
31. type data has been imported.
32. type data has been imported.
33. type data has been imported.
34. type data has been imported.
35. type data has been imported.
36. type data has been imported.
37. type data has been imported.
38. type data has been imported.
39. type data has been imported.
40. raster data has been edited to some extent.
41. thinning has been done since last edit
42. assembly has been done since last edit.
43. a crt plot has been done since last assembly
44. a paper plot has been done since last assembly
45. a report has been done since last assembly
46.
47.
48.
49.
50. type data has been exported.
51. type data has been exported.
52. type data has been exported.
53. type data has been exported.
54. type data has been exported.
55. type data has been exported.
56. type data has been exported.
57. type data has been exported.
58. type data has been exported.
59. type data has been exported.
60. type data has been exported.
61. type data has been exported.
62. type data has been exported.
63. type data has been exported.
64. type data has been exported.
65. type data has been exported.
66. type data has been exported.
67. type data has been exported.

68. type data has been exported.

69. type data has been exported.

Thus, **status** provides the user with a detailed report on the current map's operational condition.

The **step** command provides incremental map display movement in the specified direction.

step

Shifts map display in specified direction.

step d

Shifts map display in the direction of **d**.

Options/arguments:

d Direction of shift, including:
left = shift screen to the left
right = shift screen to the right
up = shift screen upward
down = shift screen downward

Example(s):

To shift the display screen to the left one increment, enter:

step left

See also:

view

The **thin_lines** command is used to reduce the width of all lines in a raster by deleting rows of pixels off of each side of each line. This is most useful for scanned data in which linework is often greater than unit width. Two arguments control the type of thinning action which occurs: the **t** value is a code which defines the thinning type and is a mandatory parameter; the default is **0**, as described below. An optional [**p**] value identifies the number of thinning passes that LTPlus should make through the data; the default is **0**, which thins until all lines are a single pixel in width.

thin_lines

Thins raster lines.

thin_lines t [p]

Removes one pixel row from both sides from all lines for each pass specified. The thinning type action can be defined.

Options/arguments:

t Thinning type. Zero (**0**) is the default.
0 = Nondestructive, leave spur ends, remove square corners and stairsteps.
1 = Destructive, leave spur ends, leave square corners and stairsteps.
2 = Destructive, removes all edge pixels, including all 1-pixel wide lines!!
(This is for special uses only, not for general consumption.)

p Number of passes for thinning. The default is zero, which thins until single pixel width is reached.

Example(s):

To thin the raster lines in a non-destructive manner removing corners and stairsteps, enter:

thin_lines 0

See also:

assemble

This command will restore the area of the previous interactive **raster** change to the way it is on the last stored raster. It WILL NOT fix changes in attributes or vector data. It WILL NOT fix raster changes made by such commands as **del_spurs**, etc. (for these, try re-acquiring the map). Successive uses of the **undo** command will undo successively earlier raster changes up to a *limit* of **32** previous changes. **Undo** will not undo **undo** as it is already undone. So there.

undo

Restore last interactive **raster** change.

Options/arguments:

None.

Example(s):

To **undo** the previous line portion drawn (drawn with the **start** and successive uses of the **continue** command). Note this only undoes the effects of the last press of the **continue** key.

undo

To undo the previous 3 uses of the **continue** key while in **separate** mode:

undo
undo
undo

See also:

recall, acquire

The **vertex_move** mode is handy for shifting a vertex position (X Y coordinates) on one map to match one on an adjacent map. This activity is the digital form of manual edgematching and it helps to insure that the maps comprising a project will form one large map with integrity in the destination GIS application. As previously mentioned, **vertex_move** is a mode; when it is on, the **start** and **continue** commands will perform the actual vector coordinate shifting. These commands are most efficiently activated with the mouse or cursor supplying the X Y positions. The normal sequence of actions would be:

Activate the **vertex_move** mode (kbd entry, or menu selection), and specify the level to which to snap.

Position the screen cursor (using the mouse or digitizer cursor) on or near the point (vertex) to be moved and activate the **start** command (kbd, function key, cursor key press). LTPlus will search until it locates the closest actual vertex position and highlight it.

Move the screen cursor to a position on or near the vertex to which the highlighted location should be moved and activate the **continue** command (kbd, function, cursor key press). LTPlus will search for any vertex that is within a 1-pixel radius of that position and modify the previous first vertex to match. If no vertex is found within the 1-pixel radius, the **X Y** values will remain unchanges.

Repeat the **start-continue** steps for all vertices which need to be moved. End this activity by activating a new mode or by executing clear mode.

vertex_move l

Sets the graphics input mode to **vertex_move** (for edgematching). When in this mode, **start** will search for the nearest vertex (assembly required) to x y, **continue** will search for the nearest vector vertex. If one is found within a 1-pixel radius, it will snap the original vertex to it. If none is found, it will place the original vertex at x y. Note that this affects only the vector file (not the raster), and that assembly is required.

Options/arguments:

l Level to which to snap: **32-95**

Example(s):

To activate the **vertex_move** mode and snap to level **40**, and move the vertex near **600,432** to the level **40** vertex near **604,430** enter:

vertex_move 40
start 600 432
continue 604 430

The **vertex_thin** command will remove vertices controlled by a user input radius and set number. Smoothing is also allowed. Nodes (intersection of two or more line segments) will not be moved or removed. Only the **vector** (not the raster) is modified. This command operates only on map data, not on imported vector data (not data on levels 32-95).

vertex_thin

Reduces vectorized data.

vertex_thin s c r m

Reduces vectorized data in set **s** by eliminating vertices based on a tolerance (corridor radius) **r**. **r** is in units of units code **c**. All vertices that lie within distance **r** of a straight line (in raster coordinates) between trial endpoint vertices are eliminated. Affects only the vectorized data. Sharp corners can be rounded by adding vertices smoothing. Assembly required. NOTE: do not confuse **vertex_thin** (which deletes vertices from vector files) with **thin_lines** (which erases pixels on the raster until raster lines are 1-pixel thick). If the **m** parameter is non-zero, then vertices will be added to sharp corners to smooth them.

Options/arguments:

s	Set number of objects to vertex_thin: **0-31**
c	Coordinate system code (see coord_type)
r	radius of tolerance for vertex thinning.
m	smoothing code: **0** = none **1** = round sharp corners

Example(s):

To reduce the vector data in set **13** by eliminating the vertices which are within **25** feet of one another without smoothing, enter:

vertex_thin 13 2 25 0

See also:

assemble, vertex_move

With the **view** command, the map may be displayed at one of four resoultion definitions as specified by the user.

view

Display the raster at the defined resolution.

view vtype x y

Displays raster at one of four resolutions, defined by vtype.

Options/arguments:

vtype	View type, where:	
	map	= entire raster within a 400 x 400 pixel square
	block	= 4:1 (approx. 1600 pixels high)
	full	= 1:1 (approx. 400 pixels high)
	zoom	= 1:8 (approx. 50 pixels high)
	last	= previously viewed resolution
x,y	Can be user specified, but are normally set by the program using current cursor location.	

Example(s):

To **view** the raster at full resolution, enter:

view full

To **view** the raster at a 1:8 resolution (approx.), enter:

view zoom (program will prompt you to press the cursor at the position to zoom on)

See also:

graphics

---

wait

Displays a message and halts program until [RETURN] pressed.

wait text

This command displays a message in the lower (command) portion of the screen in black on a light red background. The message is centered on the second display of a three line background. The normal message display and any other text is temporarily removed. A prompt in brown is displayed informing the user to press [RETURN] when ready to continue. This command is designed for use in demonstration macros to describe what the program is doing.

Options/arguments:

text any text string up to 80 characters.

Example(s):

To tell the viewer about a zoom display of the raster:

wait This is what the raster looks like close-up.

See also:

pause

---

wink

Sets the mode to **wink**.

wink

Sets the graphics input mode to **wink** which turns an *on* pixel *off* or an *off* pixel *on*. Use **start** to begin and **continue** to **wink** subsequent locations.

Note: The current mode/action is displayed in the upper-left window.

Options/arguments:

None

Example(s):

To turn an *on* pixel *off* (at various locations), enter:

wink
start 57 89
continue 34 200
continue 500 432
continue 1 89

See also:

continue, start