Chapter 10 -- MAP MANIPULATION

This chapter contains the Map Manipulation commands which, in conjunction with the Raster Manipulation commands discussed in the *Chapter 11*, comprise the very heart of LTPlus. Map manipulation commands are those functions which interact with the data elements which belong to a specific map. In contrast to the raster manipulation commands, many of the map manipulation functions operate with the map data in vector format.

This category of commands is sort of a mixed bag. Included are commands for constructing topology, repairing crashed files, preprocessing import files, updating files created by a previous version of LTPlus, copying or creating registration information, logging various types of map conditions, creating and executing map lists, overlaying two maps, and performing vertex snap and thin functions.

THE REGISTRATION PROCESS

Every map processed in LTPlus must be registered (not for the Selective Service but to the real world). 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 coordinate system or another, are known. Whenever possible, all four corners should serve as the map registration control. Ground coordinates may be entered as latitude and longitude, or as X (*easting*) and Y (*northing*) coordinates in *State Plane* feet or *UTM* meters. 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, geodetic registration may be satisfied with a single corner entry. The current registration data for a map may be displayed by executing the **register** command with a **-d** argument. The current registration data for a map may be printed by executing the **register** command with a **-p** argument. When using the **register** command to enter registration information, the coordinate type which is to be entered must be defined; the following table summarizes the available coordinate type codes:

COORDINATE SYSTEM CODES

0 = Raster pixels
1 = N. longitude, W. latitude (0,0), decimal degrees
2 = UTM meters
3 = State Plane feet
4 = Digitizer1, absolute coordinates
5 = Digitizer2, absolute coordinates
6 = Map inches
10= User-defined units
11= User-defined units
12= User-defined units
13= User-defined units
14= User-defined units
15= User-defined units
16= User-defined units
17= User-defined units
18= User-defined units
19= User-defined units

REGISTERING A SCANNED MAP

A scanned map is loaded directly into LTPlus as a raster. To register a scanned map, the user must advise LTPlus about the relationship between the scanned map corner positions and their real world locations. To accomplish this, the **register** command is executed with a coordinate type code of **0** to indicate that the coordinate positions which follow represent the location in raster pixel units. Using the mouse or tablet cursor in relative mode, the user then identifies each corner position within the scanned map by positioning the cursor over the raster image of the corner and entering a **start** or **continue** command (usually via a pre-programmed button press). Each **register** command so initiated also contains a corner identifier code to indicate which corner is being submitted. Once the corner positions within the scan have been identified in this manner, the map should be geodetically registered. This activity will identify true ground positional information for each of the four corners. It is this correlation (between the raster and the true ground) which allows LTPlus to determine the true ground positions of all of the vertices which will be **assemble**d from the scanned, raster data.

For geodetic registration, a single corner identifier parameter should be supplied for maps which are rectangular with respect to latitude and longitude and for which **map_ht** and **map_wd** values have been previously defined in the **header** file. The user need only enter one pair of coordinates for this type of map, and will do so after defining which corner's values will be entered. The remaining three corners will be calculated. Maps which are not rectangular with respect to latitude and longitude will require the user to enter all four corner coordinate values (along with the appropriate corner identifier code).

COPYING REGISTRATION

The ability to copy a previous map's registration with the **reg_copy** command can be extremely important, particularly where a project involves multiple layers per map base. Rather than re-enter geodetic registration for each layer of the same map base, the initial entry can be duplicated and stored with each layer. The command requires the name of the registration file which is to be copied and at least one coordinate type code identifying which coordinate set to copy from the named file. Multiple coordinate type codes may be included in a single **reg_copy** command.

TOPOLOGICAL STRUCTURING

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. The **assemble** command is responsible for vectorizing the raster data to form the type of objects specified by the user in the **header** file (**maptype =**). The **assemble** command initiated with no associated argument also reduces the number of vertices to produce a smoothing affect to the linework. To suppress the data reduction action, the **assemble** command should be implemented with the **-n** argument. The result will be vectors which trace the raster pixel locations exactly and do not appear smooth (stairstep appearance). Use the **graphics** command after assembly to view the raster and vector data simultaneously. The contrast between the reduced and non-reduced **assemble** command usage will be readily (and graphically) apparent.

LOGGING ASSEMBLY ERRORS

The **log** command is a very valuable LTPlus tool. It provides the user with information critical to the map data editing process. After the assembly process is complete, the user may **log** any and all occurrences of isolated points, spurs and isolated holes which exist in the vectorized data set. Using the **step** or **goto** command, each of the logged error locations may be graphically visited and corrected, if necessary. The map may then be re-assembled. Once the **log** file is empty or contains only acceptable error locations, the assembly process may be considered complete. The **log** command is implemented with an action code to identify the type of occurrence to be logged. The **log** file into which the locations are recorded is additive in nature. The locations generated by multiple **log** commands are appended to the **log** file until a **log clear** command is executed. Typically, the user will establish a macro containing **log** commands with the desired action codes; when that macro is executed, all of the defined types of occurrences will be recorded in the **log** file for review. The first command in the macro should be **log clear** to insure that the locations generated by subsequent **log** commands are not appended to any previously found positions. The **log** actions include:

clear	=	clears the log file
point	=	log all isolated points
spur	=	log all spurs
hole	=	log all isolated holes
set	=	log all objects in set specified

FILTERING IMPORT DATA

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. The pre-processor filtering actions are entered as item=string arguments with the **b_edit** command. The input file name must also be included in the command. An output file name may be entered as well; if no output file name is designated, the input file will be overwritten with the filtered file. The filtering actions are:

-e = delete all extension characters
-c = delete all control characters
b = number of lines to be deleted from the beginning of the file
l = minimum length of line to be retained
c = number of characters to be deleted from the start of every line
i = decimal equivalent of ascii character to be deleted from file (all occurrences)

FIXING MAP DATA

Occasionally, something awful happens to map data which not only defies explanation, but usually cannot be repaired. It also always seems to happen to the most complex and/or urgent project. Well, folks....the answer to this nightmare may be the wonder function, **fix_map**. At least it's worth a try....

The **fix_map** command checks all files comprising the named map, retaining all consistent data and removing all corrupted data resulting in an accessible, albeit incomplete, map. If no name is entered, the current active map files are used. This function will repair, as best as possible, a map corrupted beyond use.

UPDATING DATA FILES

Although no significant change to the database structure is ever envisioned over the life of an application, such a drastic measure has been known to occur. If it should ever happen, the **mass_update** command will insure that old data is as good as new data. Its primary purpose is to convert old LTPlus files to ones which are compatible with a new version of LTPlus. The user will be given plenty of warning about when, if ever, use of this command will be required. All standard LTPlus updates do not require use of the **mass_update** command.

MAP OVERLAY PROCESS

The **overlay** command will cause the current, active map to be overlayed with the map name included with the command and the resulting data will be stored in the included named map's database (thus overwriting whatever was in there originally). If the original data is important, it should be copied to an offline media prior to executing this command. Unless the optional **-f** argument is supplied, only the linework will be merged. Including the **-f** parameter guarantees that both the linework and the attributes will be combined. Both maps must be assembled for this operation to work properly.

MAP LISTS

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

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 a macro.

VERTEX ACTIONS

The **vertex_** commands work upon the vector file only; hence, a map must be assembled prior to executing either of these commands against it. Keep in mind that the actions will not be reflected in the map's raster file and that the changes will be lost if the map is re-assembled from the unaffected raster file.

The **vertex_snap** command will automatically shift all vertices which lie within a specified number of coordinate units of the map border line, to an exact projected x,y position, in a user-specified coordinate system, on that line. This action is useful for eliminating *slivers* between adjacent maps for data that is to be shipped (exported) to a GIS application. Note that the map must be assembled (vectorized) for this command to have any meaning. The actual sequence of events is:

> The four vectors which comprise the map border (connecting the four corners) are calculated in a user-specified coordinate system (see **register** in this chapter, for coordinate system codes).
>
> Every vertex in the map vector file which lies within a user-specified number of coordinate system units of the the calculated border will be shifted to be on that border line and the vector x,y position modified in the database. The specified value will not be accepted if it is greater than the equivalent of three pixels.

The **vertex_thin** command is useful for reducing the number of x,y vector coordinate positions required to define a line. In effect, this command will remove vertices which lie within a user-specified distance, in units of a specified coordinate system, along the line segments which comprise the objects defined by a user-defined set. Nodes (intersection of two or more line segments) will not be removed. Only the vector file will be modified. 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).

Please review the *Command Language Reference* section of this manual for more specific usage information for each of the commands discussed in this chapter.