Chapter 9 -- INPUT MODES

In the LTPlus environment, a *mode* is considered to be a condition which, when initiated, influences all subsequent command actions until it is either cleared or replaced by another command mode execution. In the *Input Modes* category, there are 14 condition-setting commands, two condition-implementing commands, one command which retracts previous actions, and one command which simply deactivates any current mode.

CONDITION-SETTING MODES

A condition-setting mode command establishes a specific condition which will affect subsequent action commands; it produces no other results, visible or otherwise. Included in this group are: **atrbt_contur, atrbt_dig, connect, decrement, draw, erase, erase_seg, increment, query, separate, vertex_move, wink, recall** and **bridge_gap**. Once a mode has been established, the action associated with the mode is generally accomplished by executing the **start** and/or **continue** command(s), usually from the mouse or tablet cursor button presses which activate those command(s). As a quick example, if a user executes the **draw line** mode command, nothing happens until subsequent **start** and **continue** sequences are executed. In the case of **draw line**, a **start x,y** command begins a line and subsequent **continue x,y** command(s) cause straight line drawing between each x,y position entered. A new line may be initiated with another **start x,y** command. Typically, the **start/continue** and associated x,y positions are entered with the mouse or tablet cursor.

The **atrbt_contur** command sets the graphics input mode to allow the user 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 an elevation is specified with the **atrbt_contur** command, that value will be assigned to the first line crossed. Otherwise, the beginning elevation attribute will be acquired from the first contour line crossed which must already have a valid, numeric contour elevation attribute. Subsequent contour lines will be assigned elevation attributes based upon an **increment/decrement** command and the contour interval value from the header file. The action code argument, included as part of the **atrbt_contur** command, will establish whether or not multiple attributes are allowed (for checking). The action codes permitted are:

- **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.

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 pressing **'undo'** will remove all attributes from the contours just mislabeled. While this doesn't restore the correct labels, it removes the incorrect labels and identifies the contours which need to be re-labeled.

The **atrbt_dig** command sets the graphics input mode to allow the user to assign attributes to objects using the current, active graphics input device (GIN). The attribute string specified as part of the command is applied to the specified object type as defined by the specified action code. This mode is activated with a **continue** command. The action applying the attribute string to the object type will occur at every x,y coordinate position defined by any subsequent **continue x,y** function execution. The **start**, **increment** and **decrement** commands will behave like **continue**. The object type codes include:

- **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)

The action codes for this function are defined in the **atrbt_contur** command description, above. The attribute parameter submitted with the **atrbt_dig** command has some special capabilities. The following special variables will perform as described:

$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 checks the last attribute assigned to see if it contains a number and stores it. The first number found is stored. If no number is found, then 0 is stored. When the attribute is assigned, the $++ variable is replaced by the stored number incremented by 1. For example: if the last attribute assigned was 'a255 965zz' and the string entered with the command is 'qbc$++ stupid', then the next object is assigned an attribute of 'qbc256 stupid'. The next attribute will be 'qbc257 stupid', and so on.

The **connect** 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 commands. 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 **continue**. 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** commands will behave like the **continue** command.

The **decrement** mode allows the user to add 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 **decrement** point will be assigned a numeric attribute in decrements of the contour interval value stored in the **header** file. The attribute value will be assigned to the entire line, even if only a portion of the line is visible on the screen display.

The **draw** command activates the graphics input **draw** mode with a defined shape type and with an optional **-s** (automatic snap) capability. A variety of shape types are permitted, including **point** (single pixel), **line** (single-pixel width), **fat_line** (3-pixel width), **rectangle** (defined by a **start x,y/continue x,y** pair of positions) and **wedge** (defined by one start x,y and two continue x,y positions). 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 radius is defined as the **search_radius** value, in pixels, stored in the **header** file. Use **start** to begin drawing an object and **continue** for all subsequent vertices for that object. Then, use **start** to begin drawing another object. Note that the current mode/action will be displayed in the upper-left screen tile.

The **erase** mode allows erasing (paint eraser) in the specified shape. It behaves like the **draw** command, except that it eliminates rather than creates pixels. The **-s** option activates snap mode (see **draw**). Use the **start** command to begin erasing an object, and the **continue** command to **erase** subsequent x y positions as determined by the shape type. Use the **start** command to re-position for the start of a new eraser sequence or action. Note that the current mode/action will be displayed in the upper, left screen tile.

The **erase_seg** mode allows the user to **erase** the segment closest to location specified by the entered x,y position. The file is searched for a segment which lies within the **search_radius** (in pixels) defined in the **header** file. A segment is defined as a sequence of pixels which lie between two nodes. The entire segment is erased. To erase a portion of a line which does not lie between two nodes, the user should use one of the other **erase** commands or create two false nodes with the **wink** command and then use the **erase_seg** command to eliminate the portion so designated.

The **increment** mode allows the user to add 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 contour elevation attribute in increments of the contour interval value stored in the **header** file.

The **query** command sets the active mode to **query**. The object of the **query** is determined by the object type argument. Current display units are the default. Once in **query** mode, additional queries are made using the **continue** command. As an example, once the **query** mode is set for **polygon** type objects, when an x,y position is entered (from the mouse or tablet cursor usually), the record and serial number as well as any associated attributes are displayed for the polygon in which the **x,y** position is found. The x,y position, in the current display units, may also be entered from the keyboard, allowing the user to determine which object type is located at the specified coordinate position.

The **separate** mode turns pixels off in a straight, unit-width line between two 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).

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 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).
>
> 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 unchanged.

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**.

Note that this command affects the assembled (vectorized) data file only; the raster file is not modified. The map must have been previously assembled. Re-assembly of the raster after use of this command will effectively undo the **vertex_move** modifications.

The **wink** command sets the graphics input mode to **wink** to turn an *on* pixel *off* or an *off* pixel *on*. Use **start** or **continue** to **wink** subsequent locations. Note that the current mode/action is displayed in the upper, left screen tile.

The **recall** mode allows the user to sequentially recover the raster data last stored with the save command. The raster pixels are extracted from the file with a shape type specified with the **recall** command. The shape types include:

point	(single pixel)
line	(1-pixel width)
fat_line	(3-pixel width)
arectangle	(internal area of a rectangle)
brectangle	(1-pixel wide perimeter of rectangle)
wedge	(walking triangle)
separate	(same area as the **separate** mode)
connect	(same area as the **connect** mode)

The primary purpose of the **recall** command is to repair operator-generated mistakes; it is similar to the **undo** command. Once the **recall** mode has been implemented, the **start** command will begin the recovery of a specific type object. Subsequent **continue** commands will recover sequential vertices for that object. Another **start** command will begin a new shape type object. The current mode/action will be displayed in the upper-left screen tile.

The **bridge_gap** command sets the graphics 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 spur endpoint. The search will then **continue** for a second, logical spur endpoint to which the first can be connected. When two, logical points 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: the current mode/action is displayed in the upper-left window.

CONDITION-IMPLEMENTING MODES

A condition-implementing command generates some sort of change by implementing the current mode against the map data. Only two such commands exist: **start** and **continue**.

The **start x,y** command initiates the current mode starting at a location defined by the **x,y** coordinates (in the current display units). Note that the current mode/action will be displayed in the upper-left screen tile. The **start** command is usually mapped to one of the mouse/tablet cursor buttons such that depressing the button executes the function with the x,y coordinate pair representing the absolute position of the mouse/cursor. The command and x,y position may also be entered via the keyboard. The x,y position is defined by the current display units (see **coord_type**).

The **continue x,y** command continues execution of the current mode/action at the **x,y** coordinate position, in the current display units (see **coord_type**). 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.

RETRACTING A PREVIOUS COMMAND

The **undo** command will restore the area of the previous interactive raster change to the way it is on the last stored raster. It will also remove attributes assigned while in **atrbt_contour** mode. It WILL NOT fix changes from other **atrbt_** modes or changes in vector data. It WILL NOT fix raster changes made by such commands as **del_spur**s, etc. (for these, try re-acquiring the map). Successive uses of the **undo** command will **undo** successively earlier changes up to a limit of 32 previous changes. **Undo** WILL NOT **undo** an **undo** command as it is already undone, so there! Are you unready?

MODE DEACTIVATION

The sole purpose of the **clear_mode** command is to deactivate any current and active mode. The current mode is also deactivated by execution of another mode command. It is a good idea to execute a **clear_mode** command to avoid generating garbage data by inadvertantly activating **start** and/or **continue** commands while in a [forgotten] active mode state.