"TEDIT BEHIND EXEC WINDOW"
PRELIMINARY DESIGN NOTES
JIM BLUM
6/3/85

This paper is a list of issues and alternatives to designing and implementing TEDIT behind EXEC windows.

KNOWN ISSUES TO BE SOLVED

1. Certain functions which normally expect a display stream as the argument have to be changed to accept a textstream as a valid type of stream (or whatever kind of stream it ends up being, for now I will refer to it as a TEDITSTREAM). Some example functions are TTYDISPLAYSTREAM, (or whatever mechanism we use to replace WFROMDS functionality).

2. There has to be a place holder or mechanism by which it is know to TEDIT where the already processed text ends, and the current edible text starts and where/when it becomes processed, so that unprocessed may be edited and already processed input may not be.

3. The input may be different or at least have different "looks" (echoing) than the output. Examples, raising lower case to upper case; confirmation by carriage return may be replaced by "Yes" in the output stream, etc. How do we handle this?

4. Other functionality of TTYIN which at the time of this writing I am not yet familar yet.

5. To what level do we support display stream graphic operations, such as MOVETO, DRAWLINE, clippingregion, XY coordinates, etc. Do we just paint graphics and don't capture them in any way, do we create imageobjects, or what?

6. Should we support two kinds of EXEC, one which uses the display stream to insure that all old programs will work as before, and a new type of EXEC which uses TEDITSTREAMS.

7. Do we implement this in one window in which it operates as one TEDITSTREAM, or do use a readonly TEDITSTREAM for backing, and use a separate window below the TEDIT window (with no border) to contain the current text which would use TTYIN to do the editing.

One advantage to implementing it as one TEDITSTREAM is that we could provide one standard method of backing display streams and handling text (ie, change the standard TEDIT interface to work this way) and possibly eliminate duplication of code such as TTYIN, thus making the system easier to maintain. On the other hand, it may turn out, that trying to handle this case as a general case in TEDIT is too envolved or too slow, and keeping them in two separate windows makes more sense.