Edit Buffer

The edit buffer is used to easily edit the contents of a variable on the calculator. Without the edit buffer, every time you needed to grow or shrink a variable, you would have to use InsertMem or DelMem, and then update the size bytes. For a variable that is constantly changing size, ie. one that's currently having data inputted into it, this is a waste of time for both the CPU and the programmer. The edit buffer solves this problem.

When a variable is edited, all of available RAM is inserted into it. That way, all the other variables in RAM only have to be moved around twice (once when the edit buffer is opened, and once when it's closed), instead of every time a byte needs to be added or removed from the variable. However, this also means that while an edit buffer is open, the application can't allocate any other memory. This means no creating or resizing other programs, and no pushing to or popping from the floating point stack.

Opening an Edit Buffer

This example assumes the type of variable you're opening in the edit buffer is a program. To edit a currently empty program, use the **SetEmptyEditPtr** entry point. If you need to edit a program that already has data in it, use some code like this:

```
SetupEditStuff:
       ld     hl,progname
       rst    rMov9ToOP1
       B_CALL EditProg
       set    editOpen,(iy+editFlags) ;Editor is running
       ld     hl,(iMathPtr1)
       inc    hl
       inc    hl
       ld     (editTop),hl             ;Top of edit session
       push   hl
       ld     hl,(iMathPtr2)
       ld     (editCursor),hl
       push   hl
       ex     de,hl
       ld     hl,(iMathPtr3)
       B_CALL CpHLDE
       jr     nc,SetupEditProgSwapSkip
       ex     de,hl
SetupEditProgSwapSkip:
       ld     (editBtm),hl
       ld     (editTail),hl
       pop    hl
       pop    de
       or     a
       sbc    hl,de
       ret    z
       ld     c,l
       ld     b,h
       ld     hl,(editCursor)
       dec    hl
       ld     de,(editTail)
       dec    de
```

```
lddr
inc    hl
ld     (editCursor),hl
inc    de
ld     (editTail),de
ret
```

Edit Buffer Variables

The edit buffer uses four memory variables to keep track of data in the edit buffer:

editTop Holds a pointer to the beginning of the edit buffer. This will be directly after the size bytes for the variable. The value in editTop does not change during the course of the edit session.

editCursor Holds a pointer to the equivilant location in the edit buffer for the on-screen cursor. The data between editTop and editCursor is all the data to the left of the cursor. The value in editCursor is variable and depends on how much data is to the left of the cursor.

editTail Holds a pointer to the all the data that's after the cursor. This data fills the area between editTail and editBtm. The area between editCursor and editTail is the free space for data to grow into.

editBtm Holds a pointer to the end of the edit buffer. The value in editBtm is constant and doesn't change during the edit session.

Navigating the Edit Buffer

Luckily, you don't have to modify the above variables manually; there are entry points provided to help navigate through the edit buffer.

BufClear Clears all the data in the edit buffer.

BufDelete Deletes the current token from the edit buffer.

BufInsert Inserts a token into the edit buffer, before the current character.

BufLeft Moves the edit pointers to the left.

BufReplace Replaces the current token in the edit buffer.

BufRight Moves the edit pointers to the right.

IsAtBtm Check if the cursor is at the bottom of the edit buffer.

IsAtTop Check if the cursor is at the top of the edit buffer.

Closing an Edit Session

To close an edit session, use **CloseEditEqu**. You will probably want to close the edit session as soon as you are done with it, but in any case, it needs to be closed before the app returns to the OS. If you are using GetKey while the edit buffer is open, this means you will you will need to use Put Away notification to close the edit buffer before returning to the OS.