RENUM.BAS

renumber.zip

This module is part of the original MMBasic library. It is reproduced here with kind permission of Hugh Buckle and Geoff Graham. Be aware it may reference functionality which has changed or is deprecated in the latest versions of MMBasic.

RENUM.BAS

RENUM.BAS is written in MMBasic and requires at least version 4.3a.

Renum.bas renumbers basic files, the same way the RENUMBER command used to work in version of MMBasic prior to Version 4.3a, and the RENUMBER.BAS file in the library used to work. The RENUMBER command has been removed from MMBasic and the old program has been removed from the library.

Because of the length of the program (around 1900 lines) it is modularised (**Modules are in the attachments tab, top right**) and uses the Chain command that was introduced in MMBasic 4.3. CONFIG.BAS, the second module, chains back to the program held in variable MasterFile$. If you use CONFIG.BAS for your own programs remember to set this variable - or change the code to suit your needs.

The program has some enhancements compared with the older versions. - a line out-of-order check. A priority, because out-of-order lines caused an embarassing bug for me a while back; - it ignores potential labels and line numbers if they are within remarks or print statements; - it attempts to indent based on the length of the longest line number (but see problems below); - it has options to increment the line numbers, even if the line number is blank, and to add line numbers to blank lines; - it has some extra validation checks; - it stores your preferred values in a configuration file.

It also has problems. - It has meaningful comments in the code, so it's big. - it can't properly format programs that start off with bad formatting, especially if there's a mix of code with and without line numbers. - it uses a lot of memory. To counter this there is a check at the beginning of RENUM.BAS to switch colour MMs to Mode 1 to conserve memory. The code also releases memory after use where possible. - there is a lot of debug code still in the program. There's an entry in the renum.cfg file that turns it on and off. If you find line numbers in your code (ie, not those ones at the start of the line) with leading astrisks, you have debug enabled. Edit the config file and set bDebug to FALSE. I'll remove the debug code once I know the program works well in the 'real world'. - The input and output filenames must be different. Once I know the program works reliably I'll add the 3 or 4 lines needed to close the input file, rename the original to a .bak file, then rename the output file to the original filename.

HOW TO RUN THE PROGRAM You can run RENUMBER.BAS from the MMBasic prompt with no parameters. It prompts for: 1) The name of the program to be renumbered. The .bas extension is optional. 2) The name of the output file. Again, .bas is optional. It cannot be the same as the input file. 3) The start line number in the basic program where you want to start renumbering. 4) The first line number in the output file. 5) The increment you want between line numbers.

It then prompts for whether you want to put line numbers on those lines that do not have them, or, if you don't want to do that, whether you want to increment the line number counter over those lines without line numbers.

For example, consider a file with a mix of lines with and without line numbers: 10 line 1 20 line 2

```
line 3 with no line number
line 4 with no line number
```

120 END

If you choose to put line numbers on those without them, you will get 10 line 1 20 line 2 30 line 3 with no line number 40 line 4 with no line number 50 END

If you answer “N” to the option above, you are prompted as to whether the line numbering should increment when the program finds a lines without line numbers. If you answer “Y” you get 10 line 1 20 line 2

```
line 3 with no line number
line 4 with no line number
```

50 END

If you answer “N”, you get 10 line 1 20 line 2

```
line 3 with no line number
line 4 with no line number
```

30 END

Finally, it prompts whether you want to use the FORMAT.BAS program. If you do, it prompts for: 1) the indent (accepts 1 to 6, with a default of 2). 2) whether the format should display the formatted program on the screen, a screen at a time, as it is processed.

Your answers are saved into a configuration file called renumber.cfg. The program initially asks if you want to use the same answers as the last time the program was run. If you answer “y” the values are loaded from the configuration file.

The renumbered program is written to the output file.

You can also pass parameters on the command line if you use the MMBasic Implied Run format. Type RENUMBER /? for a list of the options. Note: if you use the MMBasic Implied Run format, you can use the undocumented /D parameter to toggle debug mode.

PROGRAMMING DECISIONS I planned to offer an option to align the start of lines with the end of the longest label (as it currently does with line numbers), but decided not to because many programmers have the label on a separate line with no code following, and aligning to those labels would not add much to the code layout, but would waste valuable space on the LHS of the screen.

I was trying to get line formatting perfect but that would have needed code analysis to get the indent right. I had tried to replicate the indenting in the original code, but it was too complex - especially if lined and unlined code were mixed and started in different columns. I'll leave that up to Hugh's FORMAT program.

I decided not to code an option to strip out all linenumbers that weren't a target (eg GOTO target). If anyone decides to implement it, I made the 'lines' array 3 dimensional and tagged any lines that were

targets (ie follow a GOTO, GOSUB etc.) in pass 2. You would have to move this check to pass 1 (not trivial) or make a third pass and reprocess the input file once the second pass had identified the targets. Probably the easiest way would be to scan the array and replace all line numbers in dimension 2 of the array with 0 if they weren't a target. ie, if lines(linenum,3) = 0 then lines(linenum,2) = 0. Then use the array value when adding the new linenumber to the line of code. Those with 0 will have spaces instead of a line number.

USEFUL ROUTINES This program calls CONFIG.BAS. I tried to make CONFIG.BAS as stand-alone as possible. All the code that has to be customised for your program is in the last two functions. And to make it REALLY EASY, I wrote a program called CONCODER.BAS to create all the code for those two functions. Set up your config file (I'd suggest copying from an old one and editing it with a text editor), and run CONCODER. It will prompt you for the name of the config file and the file you want to have the output written to.

Here's an example of the contents of a Config file: ' This is the config file for EXAMPLE.BAS (comments are ignored by the config processor). CITY=Bangkok COUNTRY=Thailand LineCount=22 LastFileOpened=ABCD.TXT CompletedSuccessfully=TRUE

Note that: Comments are ignored by the processor. All entries start at the first column. There are no spaces between the config tag and the =, and the = and the value. Config tags for strings do not have a $ on the end. There are no quotes around string values. You can use boolean (logical) values. True and False are defined at the top of CONFIG.BAS and bad things will happen if you remove the definition :) You can use other types of value, but you'll have to write the code to support them.

The concept is pretty simple. Values are read from the config file and allocated to global variables. The main program may change these values. At the end of the main program you call the config program again to store the global values back into the config file. The two customised functions at the end are responsible for converting the config tags into global variable names and back again, and has code to convert the values into something that can be stored and retrieved from the config file.

Before you call config.bas, set a variable called bReadConfig to TRUE. This will make it call the code to read the config file. At the end of your program set bReadConfig to FALSE and run config.bas again. This will make it call the code to write the values back to the config file. The program chains to the program held in variable MasterFile$, so set that up too.

ConCoder will read each line from the config file and prompt you for the “type” of the config item. Currently it accepts b (boolean), n (numeric), s (string) and o (other). It also asks whether the tag is exactly the same as the global variable name that the value will be assigned to. It won't be for strings because there's no $ on the end of the tag. You may also decide to use global variable names that are different to the tag for other reasons. Anyway, you can specify a different name if you want to. ConCoder will create the code to read the data from the config file and write it back again.

If you specify the type as “other” you will have to write the code to support it. I've left a gap in the functions for you to insert it, and an END statement to stop the program if you don't.

Once ConCoder has run, select all the text from the output file and overwrite the two functions at the bottom of CONFIG.BAS.

If you keep all your MMBasic programs in one folder you will have to have a separate CONFIG.BAS for each program, so they will all have to have different names. I suppose you could have one config.bas that managed all config files, but I didn't write it for that situation and it could need a bit of recoding. It would also create unneeded global variables in your programs.

You could set up different config files to suit different test situations and save having to type in the values each time. Just change the name of the config file being used.

Another possibly useful routine is fileExists. It checks for the existence of a file in both the DOS and normal versions of MMBasic. It has a routine to convert file error numbers into their text descriptions.

If you have any questions or suggestions contact me at jdeakins@pcug.org.au. Just make sure it has a reasonable subject heading so I don't delete it with the daily flood of spam. Alternately, PM me at The Backshed Forum as James_From_Canb.

James Deakins March/April 2013.