CRC32.EXE

CRC (Cyclic Redundancy Codes) are methods of error detection that calculate a "check sum" for a given stream of bytes to ensure their integrity and accuracy from transmission to reception. The codes used involve both Polynomical Arithmetic and Binary Arithmetic with no carries (better known as XOR). It starts with a "generator polynomial" that becomes an accepted standard that is used, for instance, by ZIP files using DOS PKZIP, WinZip, and the ZIP capabilities now built into the operating system such as Windows 7. For those who want to explore the somewhat complex explanation of CRC math in detail, I have included CRCError.html which is a duplicate historical document published on the web in 1993.

I developed my DOS Version, CRC32.EXE for use in file error detection as well as for use with my encryption programs. Let me explain.

My encryption program, ASYMETRY.EXE require seeding a pseudo-random generator within the encryption program and then prompting for an offset within the pseudo-random stream which then becomes the beginning point for the start of encryption. The RANDOMIZE statement in QB is seeded with a single precision number, which can be up to a ±7-digit number, and which can include decimals. So the seed can be from -9999999 to +9999999, with all numbers in between, such as 99999.99 or 666.66 for example. Of course, the offset within the random stream has to be positive only with up to a 7-digit integer.

My encryption program 324&423.EXE requires an offset within the ASCII character set from 0 to 192 and an increment of that offset also from 0 to 192.

Obviously, if you had to write down the seed, offset and/or increment for each encrypted file in a list, this would defeat the purpose of encryption because someone might find, copy and/or steal your list. In short, it would be like loosing your keys. So then the temptation would be to encrypt all files with the same seed, offset and/or increment so that you could remember only one set of numbers. However, if someone tried to crack your files by brute force, this would make their task much easier - one success would be like having a master key to all the other encrypted files.

My solution to this problem was to include in CRC32.EXE the ability to also calculate the CRC32 value of a typed-in string - and even the ability to reverse this entered string which will calculate a totally different CRC32 value. This allows you to ONLY have to remember ONE EXACT string of characters that you alone choose and keep secret (and NEVER write down), such as a lyric from a song or a line from a poem; and an EXACT procedure as explained below.

AN EXAMPLE PROCEDURE:

Using CRC32.EXE to calculate a CRC32 of a string, type out your secret string of characters. I highly recommend to type everything in UPPER CASE without ANY punctuation such as commas, semicolons or periods because if you change anything later - even an extra space - the CRC32 calculation is totally different!

After typing out your secret string, hit <Enter> to go to the next line.

On the next line, type out the EXACT encrypted file name in UPPER CASE. This file will be created by whichever encryption program you decide to use after you get the CRC32 calculation. Then hit <Enter> again.

For example, assume that my secret string is:

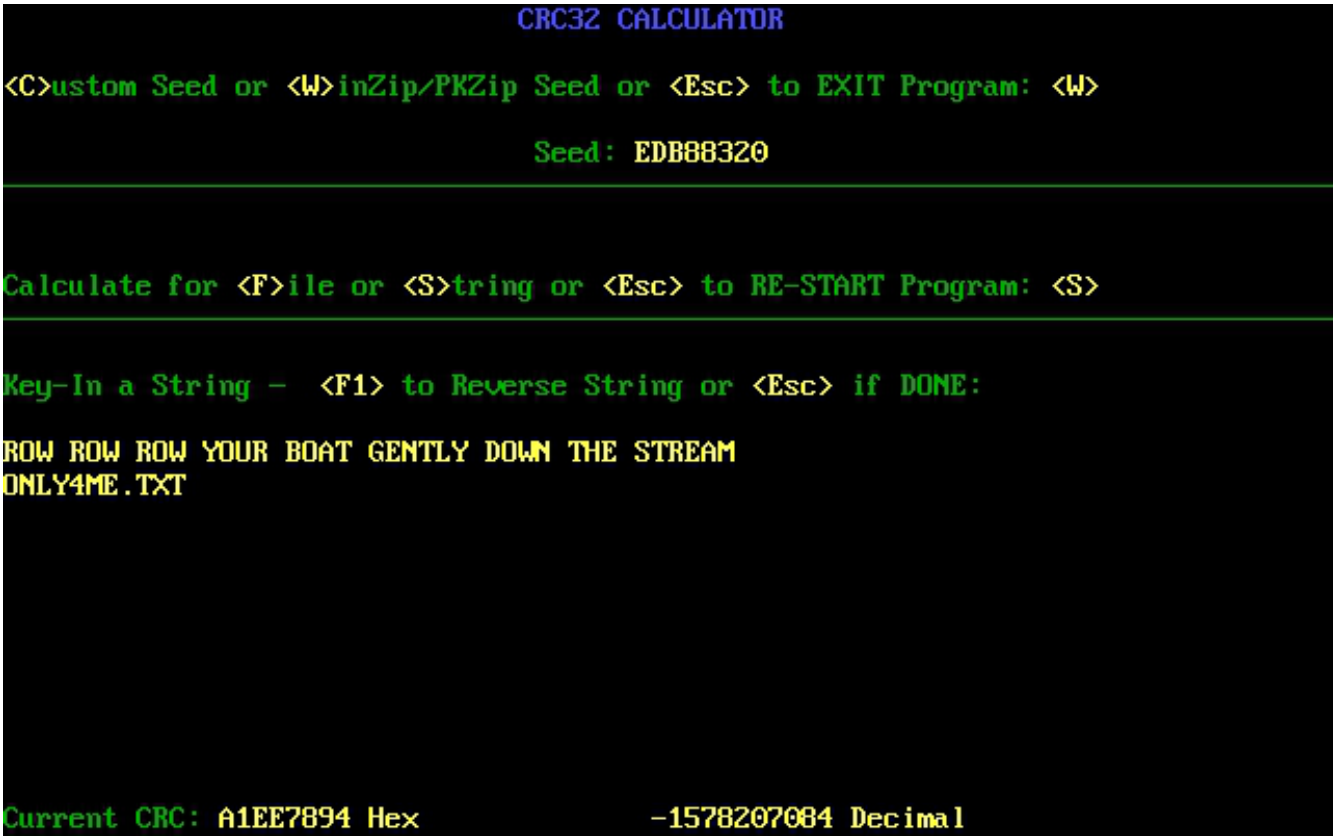
ROW ROW ROW YOUR BOAT GENTLY DOWN THE STREAM

Then assume that my encrypted file is to be named: ONLY4ME.TXT

I would type out:

ROW ROW ROW YOUR BOAT GENTLY DOWN THE STREAM

ONLY4ME.TXT



The decimal CRC32 calculation for this is: -1578207084 (Using the WinZip/PKZip "generator polynomial" seed, EDB88320). Notice that you can use a custom "generator polynomial" seed if you desire; however, that just means you have one more thing to memorize, so I do not normally recommend using it. Please note that ANY DEVIATION WHATSOEVER will result in a different CRC32 calculation. For instance, if you placed a <Space> before or after the secret string; or if you did not hit <Enter> on the last line, then the CRC32 calculation would not be correct. Also note that the CRC32 calculation updates as you type.

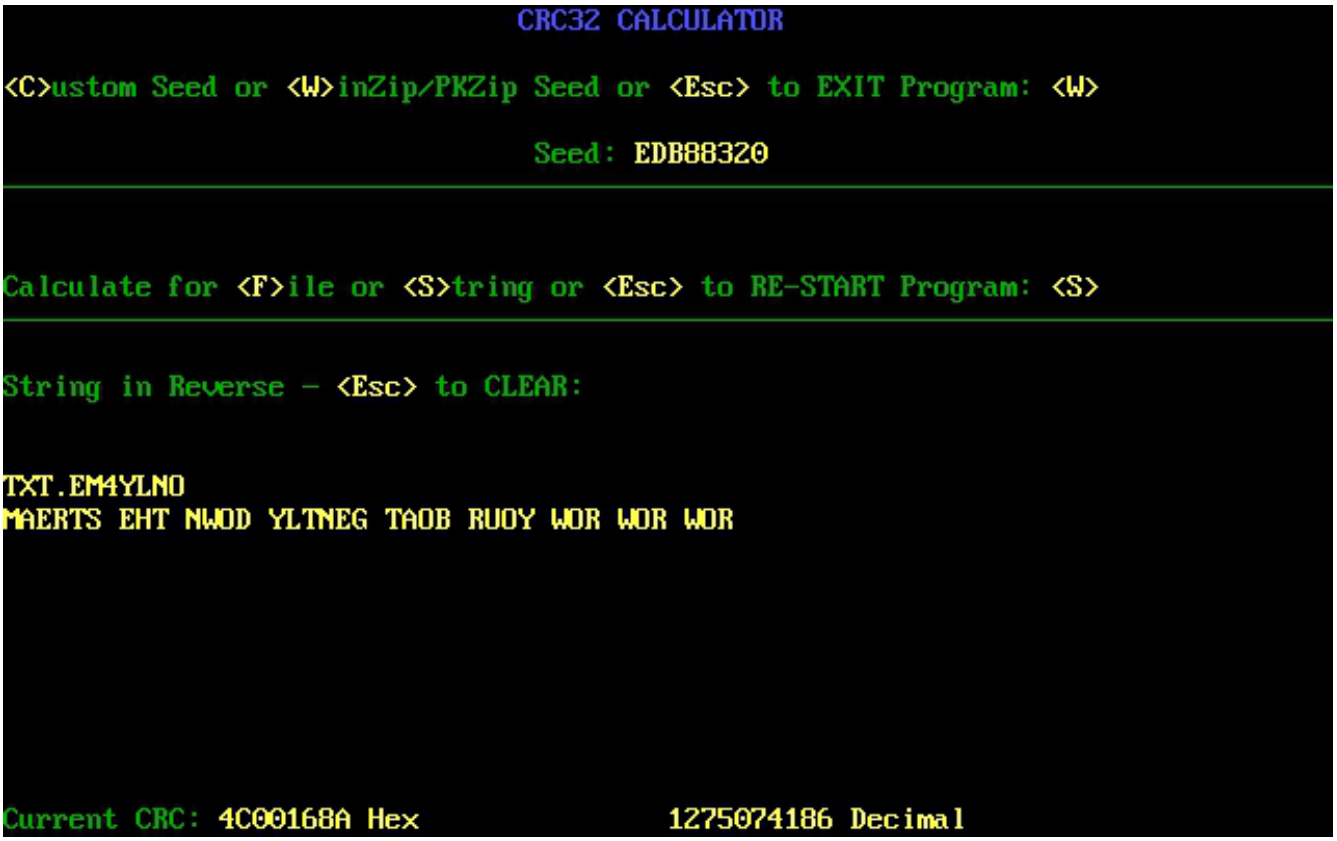
If I was using 324&423.EXE, I would use Offset 15, increment 78; then offset 20, increment 70; then offset 84, increment 0.

If I was using ASYMETRY.EXE, I would seed the pseudo-random generator with -15782.07 and an offset into the pseudo-random stream of 84.

You also have the option of using the CRC32 calculation of everything reversed by hitting <F1>. This would show:

TXT.EM4YLNO

MAERTS EHT NWOD YLTNEG TAOB RUOY WOR WOR WOR



The decimal CRC32 calculation for this is: 1275074186. So, whatever procedure you decide to use, you must be consistent for the procedure to always work.

I have also included CRCTABLE.EXE which will generate the complete table that is associated with any CRC32 "generator polynomial". This is only useful for those who wish to explore the complexities of CRC32 theory as discussed in CRCError.html.