M. David Johnson
https://www.bds-soft.com
info@bds-soft.com

Bible Quiz Mechanics

by M. David Johnson
2022/05/03

Abstract

A Bible Quiz Mechanics System is presented to provide some guidelines for selecting the number of verses for each quiz question in a quiz covering a pre-chosen selection from the scriptures.

For example, in my first Bible Quiz, GENPARTA.BAS, covering Genesis 1-11, this System was used to determine that the quiz would be covering 299 verses in 50 questions at an approximate average of six verses per question.

This system is used to develop Bible Quizzes for the 64K Color Computer 2, but it is actually written in Awk and Perl on a Debian 11 Linux server.

——

This paper and its associated code are available online at:

http://www.bds-soft.com/cocoPapers.php .

=====

Table of Contents

=====

Introduction

Bible Quizzes are somewhat different from other quizzes (e.g. math, geography, etc.). Math Quizzes lend themselves to numeric progressions: 2 + 2, 2 + 3, 2 + 4, etc. Geographic Quizzes can be self-limiting, e.g. for the Captials of U.S. States and Territories, there are only 55 such geographic entities.

But Bible Quizzes will address a specific section of the scriptures and I want the collection of questions to represent a balanced treatment of those passages. For example, the first Bible Quiz comprises 50 questions on Genesis 1-11. That's 50 questions covering 299 verses, or approximately six verses per question.

———

As another example, suppose I want to develop a 50-question Bible Quiz covering Hosea through Haggai. The Starting Book Abbreviation = HOS, Chapter Number = 1, Verse Number = 1. The Ending Book Abbreviation = HAG, Chapter Number = 2.

Running `KJVLastVerse.pl` with HAG 2 reveals that the last verse of the selection is HAG 2:23.

Running `KJVRunVerse.pl` with HOS 1:1 reports Running Verse Number 22096.

Running `KJVRunVerse.pl` with HAG 2:23 reports Running Verse Number 22879.

22879 - 22096 = 783 verses total.

783 / 50 = 15.66 verses per question, average.

This is probably more verses per question that I would normally prefer, but it serves as an adequate example, nonetheless.

Thus the question verse ranges should be a repeating 15-16-16 pattern, i.e.:

HOS 01:01 - 02:04	15 verses	Question 01
HOS 02:05 - 02:20	16 verses	Question 02
HOS 02:21 - 04:08	16 verses	Question 03
HOS 04:09 - 05:04	15 verses	Question 04
HOS 05:05 - 06:05	16 verses	Question 05
HOS 06:06 - 07:10	16 verses	Question 06
• • •		
HAG 01:08 - 02:07	15 verses	Question 49
HAG 02:08 - 02:23	16 verses	Question 50

Note that these ranges will not always come out even at the end like this one did. Also note that these verse ranges are advisory only - I will certainly deviate from them wherever I see that the Quiz would benefit from some adjustment in any given range or ranges.

But these question verse ranges will serve as a general guide to help keep the question list from becoming unbalanced.

Again, this system is used to develop Bible Quizzes for the 64K Color Computer 2, but it is actually written in Awk and Perl on a Debian 11 Linux server.

———

In works of this complexity (at least for me) typos and other errors are bound to sneak in. Please let me know about any you discover so I can note and correct them.

M.D.J. 2022/05/03
info@bds-soft.com

=====

Chapter 01: The Starting Point

On my primary Windows 10 desktop (a.k.a. “Doc”) I’ve long had an Microsoft Access database table listing the number of verses in each of the 1,189 chapters of the Bible (KJV). It also includes a running total of the 31,102 verses. Here’s a brief except from that table:

ID	RunNumChapters	BookAbbrev	ChapterNum	NumVerses	NumPrevVerses	RunNumVerses
1	1	GEN	1	31	0	31
2	2	GEN	2	25	31	56
3	3	GEN	3	24	56	80
4	4	GEN	4	26	80	106
5	5	GEN	5	32	106	138
6	6	GEN	6	22	138	160
7	7	GEN	7	24	160	184
8	8	GEN	8	22	184	206
9	9	GEN	9	29	206	235
10	10	GEN	10	32	235	267
11	11	GEN	11	32	267	299
12	12	GEN	12	20	299	319
13	13	GEN	13	18	319	337
14	14	GEN	14	24	337	361

• • •

ID	RunNumChapters	BookAbbrev	ChapterNum	NumVerses	NumPrevVerses	RunNumVerses
1183	1183	REV	16	21	30955	30976
1184	1184	REV	17	18	30976	30994
1185	1185	REV	18	24	30994	31018
1186	1186	REV	19	21	31018	31039
1187	1187	REV	20	15	31039	31054
1188	1188	REV	21	27	31054	31081
1189	1189	REV	22	21	31081	31102

To begin, I saved this table as a comma-separated variables (.csv) file and then copied that file to my Debian 11 Linux development server (a.k.a. “Sleepy”), to wit:

```
1,1,"GEN",1,31,0,31
2,2,"GEN",2,25,31,56
3,3,"GEN",3,24,56,80
4,4,"GEN",4,26,80,106
5,5,"GEN",5,32,106,138
6,6,"GEN",6,22,138,160
7,7,"GEN",7,24,160,184
```

```
8,8,"GEN",8,22,184,206
9,9,"GEN",9,29,206,235
10,10,"GEN",10,32,235,267
11,11,"GEN",11,32,267,299
12,12,"GEN",12,20,299,319
13,13,"GEN",13,18,319,337
14,14,"GEN",14,24,337,361

...

1183,1183,"REV",16,21,30955,30976
1184,1184,"REV",17,18,30976,30994
1185,1185,"REV",18,24,30994,31018
1186,1186,"REV",19,21,31018,31039
1187,1187,"REV",20,15,31039,31054
1188,1188,"REV",21,27,31054,31081
1189,1189,"REV",22,21,31081,31102
```

=====

Chapter 02: The First Step

Once verseData.csv was on Sleepy, I used the following Awk script to expand the data into verseList.csv, which produced a running-numbered list of all 31,102 verses. The Awk script was short and simple:

```
#! /bin/awk -f

#####
#
# verseData.awk
# MDJ 2022/03/30
#
# Reads the verseData.csv file
#   (taken from the VerseData
#   table of the VerseData
#   MS Access Database on Doc)
#   That table includes one
#   line per KJV chapter.
#
# And outputs the verseList.csv
#   file which includes one
#   line per KJV verse.
#
# To Run:
#   cd
/media/mdavidjohnson/Elements/workingdir/Bible/verseDa
ta
#   verseData.awk verseData.csv
#
#####

# verseData input:
#   $1 = ID  = ID
#   $2 = RNC = RunNumChapters
#   $3 = BA  = BookAbbrev
#   $4 = CN  = ChapterNum
#   $5 = NV  = NumVerses
#   $6 = NPV = NumPrevVerses
#   $7 = RNV = RunNumVerses

# verseList output:
#   ID  = ID (Uses RVN)
#   RVN = RunVerseNum
#   BA  = BookAbbrev
#   CN  = ChapterNum
```

```
#   VN  = VerseNum

BEGIN {
     # Specify the csv Field Delimiter
     FS = ","
     # Running Verse Number
     RVN = 0
     print " "
     print "Start of verseData Expansion Run"
     print " "
}

{
     print $0
     for(j=1; j<=$5; j++)
     {
          RVN = RVN + 1
          RecordData = RVN "," RVN "," $3 "," $4 "," j
          print "\t" RecordData
          if (RVN == 1)
          {
               print RecordData >
"/media/mdavidjohnson/Elements/workingdir/Bible/verseD
ata/verseList.csv"
          }
          else
          {
               print RecordData >>
"/media/mdavidjohnson/Elements/workingdir/Bible/verseD
ata/verseList.csv"
          }
     }
}

END {
     print " "
     print "End of verseData Expansion Run"
     print " "
}

#####
#
# EOF
#
#####
```

Here's a brief excerpt from the resulting verseList.csv file:

```
1,1,"GEN",1,1
2,2,"GEN",1,2
3,3,"GEN",1,3
4,4,"GEN",1,4
5,5,"GEN",1,5
6,6,"GEN",1,6
7,7,"GEN",1,7
8,8,"GEN",1,8
9,9,"GEN",1,9
10,10,"GEN",1,10
11,11,"GEN",1,11
12,12,"GEN",1,12
13,13,"GEN",1,13

...

31095,31095,"REV",22,14
31096,31096,"REV",22,15
31097,31097,"REV",22,16
31098,31098,"REV",22,17
31099,31099,"REV",22,18
31100,31100,"REV",22,19
31101,31101,"REV",22,20
31102,31102,"REV",22,21
```

As a safety backup, I copied verseList.csv back to Doc, and added it as a new table in the database:

ID	RunVerseNum	BookAbbrev	ChapterNum	VerseNum
1	1	GEN	1	1
2	2	GEN	1	2
3	3	GEN	1	3
4	4	GEN	1	4
5	5	GEN	1	5
6	6	GEN	1	6
7	7	GEN	1	7
8	8	GEN	1	8
9	9	GEN	1	9
10	10	GEN	1	10
11	11	GEN	1	11
12	12	GEN	1	12
13	13	GEN	1	13

• • •

ID	RunVerseNum	BookAbbrev	ChapterNum	VerseNum
31095	31095	REV	22	14
31096	31096	REV	22	15
31097	31097	REV	22	16
31098	31098	REV	22	17
31099	31099	REV	22	18
31100	31100	REV	22	19
31101	31101	REV	22	20
31102	31102	REV	22	21

=====

Chapter 03: The Second Step

I established the KJVData database under MariaDB on Sleepy and used two Perl scripts to respectively insert the verseData and verseList tables into that database. The verseData table was inserted using the following `vD01.pl` script:

```
#! /usr/bin/perl -w

#####
#
# vD01.pl
# MDJ 2022/04/06
#
# Read the VerseData.csv file,
# split the data into variables,
# insert the data into the
# verseData Table in the
# KJVData datebase, and
# then display the data.
#
# Be sure to enter the proper
# mysql password before running
# this script.
#
# To Run:
#   cdVD
#   vD01.pl
#
#####

# Pre-setup
use 5.30.0;
use warnings;

# Perl's Database Interface Module
use DBI;

# Data Source Name
my $dsn = "dbi:mysql:dbname=KJVData";

# mysql Username
my $user = "mdavidjohnson";

# mysql Password
#my $password = 'intentionallyObfuscated';
```

```
# mysql Attributes
my %attr = (RaiseError => 1, PrintError => 0);

# Open the Database Connection
my $dbh = DBI->connect($dsn, $user, $password,
\%attr);

# Open the Input Data File
open(IN, '<VerseData.csv')
  or die "Can't open 'VerseData.csv' for reading: $!";

print "\nData Insertions starting\n\n";

while(<IN>)
{
     # Get a line from the file
     my $LineIN = $_;

     # Remove the line-ending "\n"
     chomp($LineIN);

     # Get the variables from the line
     (my $IDVal,
      my $RunNumChaptersVal,
      my $BookAbbrevVal,
      my $ChapterNumVal,
      my $NumVersesVal,
      my $NumPrevVersesVal,
      my $RunNumVersesVal)
     = split(/,/, $LineIN);

     # Insert the values into the Database Table
     $dbh->do("INSERT INTO verseData VALUES
     ($IDVal,
      $RunNumChaptersVal,
      $BookAbbrevVal,
      $ChapterNumVal,
      $NumVersesVal,
      $NumPrevVersesVal,
      $RunNumVersesVal)");

     # Combine the variables
     my $sep = " -- ";
     my $LineOut =  $IDVal.
       $sep.$RunNumChaptersVal.
       $sep.$BookAbbrevVal.
       $sep.$ChapterNumVal.
```

```
        $sep.$NumVersesVal.
        $sep.$NumPrevVersesVal.
        $sep.$RunNumVersesVal."\n";

      # Print the combined variables
      print $LineOut;
}

print "\nData Insertions Complete\n\n";

# Close the Input Data File
close IN
  or die "Error closing 'VerseData.csv': $!";

# Close the Database Connection
$dbh->disconnect;

#####
#
# EOF
#
#####
```

And the verseList table was inserted using the following `vL01.pl` script:

```
#! /usr/bin/perl -w

#####
#
# vL01.pl
# MDJ 2022/04/06
#
# Read the verseList.csv file,
# split the data into variables,
# insert the data into the
# verseList Table in the
# KJVData datebase, and
# then display the data.
#
# Be sure to enter the proper
# mysql password before running
# this script.
#
# To Run:
#   cdVD
#   vL01.pl
#
```

```
#####

# Pre-setup
use 5.30.0;
use warnings;

# Perl's Database Interface Module
use DBI;

# Data Source Name
my $dsn = "dbi:mysql:dbname=KJVData";

# mysql Username
my $user = "mdavidjohnson";

# mysql Password
#my $password = 'intentionallyObfuscated';

# mysql Attributes
my %attr = (RaiseError => 1, PrintError => 0);

# Open the Database Connection
my $dbh = DBI->connect($dsn, $user, $password,
\%attr);

# Open the Input Data File
open(IN, '<verseList.csv')
  or die "Can't open 'verseList.csv' for reading: $!";

print "\nData Insertions starting\n\n";

while(<IN>)
{
     # Get a line from the file
     my $LineIN = $_;

     # Remove the line-ending "\n"
     chomp($LineIN);

     # Get the variables from the line
     (my $IDVal,
      my $RunVerseNumVal,
      my $BookAbbrevVal,
      my $ChapterNumVal,
      my $VerseNumVal)
     = split(/,/, $LineIN);
```

```
            # Insert the values into the Database Table
            $dbh->do("INSERT INTO verseList VALUES
            ($IDVal,
             $RunVerseNumVal,
             $BookAbbrevVal,
             $ChapterNumVal,
             $VerseNumVal)");

            # Combine the variables
            my $sep = " -- ";
            my $LineOut = $IDVal.
              $sep.$RunVerseNumVal.
              $sep.$BookAbbrevVal.
              $sep.$ChapterNumVal.
              $sep.$VerseNumVal."\n";

            # Print the combined variables
            print $LineOut;
        }

        print "\nData Insertions Complete\n\n";

        # Close the Input Data File
        close IN
          or die "Error closing 'verseList.csv': $!";

        # Close the Database Connection
        $dbh->disconnect;

        #####
        #
        # EOF
        #
        #####
```

=====

Chapter 04: Last Verse Number

Using the KJVData database, one useful function is to be able to report the number of the last verse in any given chapter of any given book. The `KJVLastVerse.pl` script provides that function:

```
#! /usr/bin/perl

#####
#
# KJVLastVerse.pl
# MDJ 2022/04/11
#
# Given:
#   Book Abbreviation
#   Chapter Number
# Returns:
#   Last Verse Number
#   for that Chapter
#
# To Run:
#   KJVLastVerse.pl
#
# ** WARNING **
# When saving perl scripts in
# Textpad, you should use
# the "Save As" command and
# change the Line Ending Box
# to "UNIX".
#
# Otherwise, you may get:
# /usr/bin/perl^M No such file
#   ( ^M = \r\n ).
#
#####

# Pre-setup
use 5.30.0;

# Perl's Database Interface Module
use DBI;

# Data Source Name
my $dsn = "dbi:mysql:dbname=KJVData";

# mysql Username
```

```
my $user = "mdavidjohnson";

# mysql Password
#my $password = 'intentionallyObfuscated';

# mysql Attributes
my %attr = (RaiseError => 1, PrintError => 0);

# Open the Database Connection
my $dbh = DBI->connect($dsn, $user, $password, \%attr)
  or die "failed to connect to MySQL database:DBI-
>errstr()";

print "\nStarting KJVLastVerse Run\n\n";

# Declare variables
my($bkAbbrev) = "";   # Book Abbreviation
my($chapNum) = "";    # Chapter Number
my($versNum) = "";    # Verse Number

# Get the inputs and convert as required

print "Enter the Book Abbreviation: ";
$bkAbbrev = <STDIN>;
# Truncate to remove "\n"
chomp($bkAbbrev);
# Convert to all uppercase
$bkAbbrev = uc($bkAbbrev);

print "Enter the Chapter Number: ";
$chapNum = <STDIN>;
# Truncate to remove "\n"
chomp($chapNum);
# Convert to numeric
$chapNum = 0 + $chapNum;

# Call the function
$versNum = &findLastVerseNum
  ($bkAbbrev, $chapNum);

# Display the result
print "\n";
print "Book Abbreviation    = ".$bkAbbrev."\n";
print "Chapter Number       = ".$chapNum."\n";
print "Verse Number         = ".$versNum."\n";

print "\nKJVLastVerse Run Finished\n\n";
```

```
# Close the Database Connection
$dbh->disconnect;

exit;

sub findLastVerseNum
{
     # $bkA = Book Abbreviation
     # $chN = Chapter Number
     # $vsN = Verse Number

     my($bkA, $chN) = @_;

     # Prepare the SELECT Query
     my $sth = $dbh->prepare
     ("SELECT NumVerses
       FROM  verseData
       WHERE BookAbbrev = '$bkA'
       AND   ChapterNum = '$chN'")
     or die "prepare statement failed: $dbh-
>errstr()";

     # Execute the SELECT query
     $sth->execute();

     # Get the result of the query
     my($vsN) = $sth->fetchrow_array();

     # Complete the query
     $sth->finish();

     return $vsN;
}

#####
#
# EOF
#
#####
```

=====

Chapter 05: Running Verse Number

Using the KJVData database, one of the two most important functions is to be able to report the Running Verse Number; given the KJV Book Abbreviation, Chapter Number, and Verse Number. The `KJVRunVerse.pl` script provides that function:

```
#! /usr/bin/perl

#####
#
# KJVRunVerse.pl
# MDJ 2022/04/11
#
# Given:
#   Book Abbreviation
#   Chapter Number
#   Verse Number
# Returns:
#   Running Verse Number
#
# To Run:
#   KJVRunVerse.pl
#
# ** WARNING **
# When saving perl scripts in
# Textpad, you should use
# the "Save As" command and
# change the Line Ending Box
# to "UNIX".
#
# Otherwise, you may get:
# /usr/bin/perl^M No such file
#   ( ^M = \r\n ).
#
#####

# Pre-setup
use 5.30.0;

# Perl's Database Interface Module
use DBI;

# Data Source Name
my $dsn = "dbi:mysql:dbname=KJVData";

# mysql Username
```

```
my $user = "mdavidjohnson";

# mysql Password
#my $password = 'intentionallyObfuscated';

# mysql Attributes
my %attr = (RaiseError => 1, PrintError => 0);

# Open the Database Connection
my $dbh = DBI->connect($dsn, $user, $password, \%attr)
  or die "failed to connect to MySQL database:DBI-
>errstr()";

print "\nStarting KJVRunVerse Run\n\n";

# Declare variables
my($bkAbbrev) = "";   # Book Abbreviation
my($chapNum) = "";    # Chapter Number
my($versNum) = "";    # Verse Number
my($runVersNum) = ""; # Running Verse Number

# Get the inputs and convert as required

print "Enter the Book Abbreviation: ";
$bkAbbrev = <STDIN>;
# Truncate to remove "\n"
chomp($bkAbbrev);
# Convert to all uppercase
$bkAbbrev = uc($bkAbbrev);

print "Enter the Chapter Number: ";
$chapNum = <STDIN>;
# Truncate to remove "\n"
chomp($chapNum);
# Convert to numeric
$chapNum = 0 + $chapNum;

print "Enter the Verse Number: ";
$versNum = <STDIN>;
# Truncate to remove "\n"
chomp($versNum);
# Convert to numeric
$versNum = 0 + $versNum;

# Call the function
$runVersNum = &findRunVerseNum
  ($bkAbbrev, $chapNum, $versNum);
```

```
# Display the result
print "\n";
print "Book Abbreviation    = ".$bkAbbrev."\n";
print "Chapter Number       = ".$chapNum."\n";
print "Verse Number         = ".$versNum."\n";
print "Running Verse Number = ".$runVersNum."\n";

print "\nKJVRunVerse Run Finished\n\n";

# Close the Database Connection
$dbh->disconnect;

exit;

sub findRunVerseNum
{
     # $bkA = Book Abbreviation
     # $chN = Chapter Number
     # $vsN = Verse Number
     # $RVN = Running Verse Number

     my($bkA, $chN, $vsN) = @_;

     # Prepare the SELECT Query
     my $sth = $dbh->prepare
     ("SELECT RunVerseNum
       FROM  verseList
       WHERE BookAbbrev = '$bkA'
       AND   ChapterNum = '$chN'
       AND   VerseNum   = '$vsN'")
     or die "prepare statement failed: $dbh-
>errstr()";

     # Execute the SELECT query
     $sth->execute();

     # Get the result of the query
     my($RVN) = $sth->fetchrow_array();

     # Complete the query
     $sth->finish();

     return $RVN;
}

#####
```

```
#
#  EOF
#
#####
```

=====

Chapter 06: KJV Information

Using the KJVData database, the other of the two most important functions is the reverse of the first; i.e. to be able to report the KJV Book Abbreviation, Chapter Number, and Verse Number; given the Running Verse Number. The `KJVInfo.pl` script provides that function:

```
#! /usr/bin/perl

#####
#
# KJVInfo.pl
# MDJ 2022/04/11
#
# Given:
#   Running Verse Number
# Returns:
#   Book Abbreviation
#   Chapter Number
#   Verse Number
#
# To Run:
#   KJVInfo.pl
#
# ** WARNING **
# When saving perl scripts in
# Textpad, you should use
# the "Save As" command and
# change the Line Ending Box
# to "UNIX".
#
# Otherwise, you may get:
# /usr/bin/perl^M No such file
#   ( ^M = \r\n ).
#
#####

# Pre-setup
use 5.30.0;

# Perl's Database Interface Module
use DBI;

# Data Source Name
my $dsn = "dbi:mysql:dbname=KJVData";

# mysql Username
```

```
my $user = "mdavidjohnson";

# mysql Password
#my $password = 'intentionallyObfuscated';

# mysql Attributes
my %attr = (RaiseError => 1, PrintError => 0);

# Open the Database Connection
my $dbh = DBI->connect($dsn, $user, $password, \%attr)
  or die "failed to connect to MySQL database:DBI-
>errstr()";

print "\nStarting KJVInfo Run\n\n";

# Declare variables
my($runVersNum) = "";     # Running Verse Number
my($bkAbbrev) = "";       # Book Abbreviation
my($chapNum) = "";        # Chapter Number
my($versNum) = "";        # Verse Number
my(@Info) = ("", "", ""); # Information Array

# Get the input and convert as required

print "Enter the Running Verse Number: ";
$runVersNum = <STDIN>;
# Truncate to remove "\n"
chomp($runVersNum);
# Convert to numeric
$runVersNum = 0 + $runVersNum;

# Call the function
@Info = &findKJVInfo
  ($runVersNum);

# Break out the results
($bkAbbrev, $chapNum, $versNum) = @Info;

# Display the result
print "\n";
print "Book Abbreviation    = ".$bkAbbrev."\n";
print "Chapter Number       = ".$chapNum."\n";
print "Verse Number         = ".$versNum."\n";
print "Running Verse Number = ".$runVersNum."\n";

print "\nKJVInfo Run Finished\n\n";
```

```
# Close the Database Connection
$dbh->disconnect;

exit;

sub findKJVInfo
{
     # $RVN = Running Verse Number
     # $bkA = Book Abbreviation
     # $chN = Chapter Number
     # $vsN = Verse Number
     # @kjv = Information Array

     my($RVN) = @_;

     # Prepare the SELECT Query
     my $sth = $dbh->prepare
     ("SELECT BookAbbrev, ChapterNum, VerseNum
       FROM  verseList
       WHERE RunVerseNum = '$RVN'")
     or die "prepare statement failed: $dbh-
>errstr()";

     # Execute the SELECT query
     $sth->execute();

     # Get the result of the query
     my($bkA, $chN, $vsN) = $sth->fetchrow_array();
     my(@kjv) = ($bkA, $chN, $vsN);

     # Complete the query
     $sth->finish();

     return @kjv;
}

#####
#
# EOF
#
#####
```

=====

Conclusions and Future Work

As presented, this Bible Quiz Mechanics System is useful and complete. For any given Bible Quiz, all I need to do is get the beginning and ending Running Verse Numbers, subtract the first from the last, and divide by the number of quiz questions to set the average number of verses per question. Then I can simply step through the passage accordingly.

Although it might be interesting and instructive to develop a capstone Perl script to perform those calculations and output a list of the verses selected for each question, that probably would be somewhat of an overkill in this case.

=====

Appendix A: New BDS Software License

This New Software License applies to all software found on the BDS Software site, and supersedes all previous copyright notices and licensing provisions which may appear in the software itself or in any documentation therefor.

All software which has previously been placed in the public domain remains in the public domain.

All other software, programs, experiments and reports, documentation, and any other material on this site (other than that attributed to outside sources) is hereby copyright © 2018 (or later if so marked) by M. David Johnson.

All software, documentation, and other information on the BDS Software site is available for you to freely download without cost.

Whether you downloaded such items directly from this site, or you obtained them by any other means, you are hereby licensed to copy them, to sell or give away such copies, to use them, and to excerpt from them, in any way whatsoever, so long as nothing you do with them would denigrate the name of our Lord and Savior, Jesus Christ.

I make absolutely no warranty whatsoever for any of these items. You use them entirely at your own risk.

If they don't work for you, I commiserate.

If they crash your system, I sympathize.

But I accept no responsibility whatsoever for any such consequences. Under no circumstances will BDS Software or M. David Johnson be liable for any negative results of any kind which you may experience from downloading or using these items.

BDS Software's former mail address at P.O. Box 485 in Glenview, IL is no longer valid. Any mail sent to that address will be rejected by the U.S. Postal Service. See my Contact page.

M.D.J. 2018/06/08

=====