-
[fqvarfort](#)

Code to calculate the SHA256 (Secure Hash Algorithm) of a string.

```
    '============================================
    '
    ' Secure Hash Algorithm (SHA256)
    '
    ' This is a Liberty Basic implementation of
    ' of the SHA256 one-way encryption algorithm
    '
    ' Ported to Liberty Basic by fmtSoftware at
    ' http://www.fmtsoftware.com/
    '
    ' License:
    ' http://creativecommons.org/licenses/by/3.0/
    '
    ' You may use it in any application, including
    ' commercial products. A credit to the author
    ' and a link to the website must be included
    ' in your program's documentation or readme
    ' file, and in any source code released to
    ' the public.
    '
    ' http://www.fmtsoftware.com
    '
    '============================================
    ' Version 0.1.0
    '============================================


    'Global data for SHA256
    DIM SHA256.Data(0)
    GLOBAL SHA256.DataCount

    'Print a test SHA256 value
    Print SHA256.Evaluate$("test")

SUB SHA256.ConvertToWordArray sMessage$

    MODULUS.BITS = 512
    CONGRUENT.BITS = 448
    BITS.TO.A.BYTE = 8
    BYTES.TO.A.WORD = 4
```

```
    BITS.TO.A.WORD = BYTES.TO.A.WORD * BITS.TO.A.BYTE

    'Get the length of the source string
    lMessageLength = Len(sMessage$)


'Get padded number of words. Message needs to be congruent to 448 bits
,
'modulo 512 bits. If it is exactly congruent to 448 bits, modulo 512 b
its
    'it must still have another 512 bits added. 512 bits = 64 bytes

'(or 16 * 4 byte words), 448 bits = 56 bytes. This means lNumberOfWord
s must
    'be a multiple of 16 (i.e. 16 * 4 (bytes) * 8 (bits))
    lNumberOfWords = (Int((lMessageLength + _
        Int((MODULUS.BITS - CONGRUENT.BITS) / BITS.TO.A.BYTE)) / _
        Int(MODULUS.BITS / BITS.TO.A.BYTE)) + 1) * _
        Int(MODULUS.BITS / BITS.TO.A.WORD)

    'Resize the word array and store the number of
    'elements in the word array for later use
    ReDim SHA256.Data(lNumberOfWords - 1)
    SHA256.DataCount = lNumberOfWords


'Combine each block of 4 bytes (ascii code of character) into one long
    'value and store in the message. The high-
order (most significant) bit of
    'each byte is listed first. However,
    lByteCount = 0
    lBytePosition = 0
    Do Until lByteCount >= lMessageLength
        ' Each word is 4 bytes
        lWordCount = Int(lByteCount / BYTES.TO.A.WORD)
        lBytePosition = (3 - (lByteCount Mod BYTES.TO.A.WORD)) *
 BITS.TO.A.BYTE


' NOTE: This is where we are using just the first byte of each unicode

' character, you may want to make the change here, or to the SHA256 me
thod
        ' so it accepts a byte array.
        lByte = Asc(Mid$(sMessage$, lByteCount + 1, 1))
```

```
        SHA256.Data(lWordCount) = SHA256.Data(lWordCount) Or
 SHA256.LShift(lByte, lBytePosition)
        lByteCount = lByteCount + 1
    Loop


'Terminate according to SHA-256 rules with a 1 bit, zeros and the leng
th in
    'bits stored in the last two words
    lWordCount = Int(lByteCount / BYTES.TO.A.WORD)
    lBytePosition = (3 - (lByteCount Mod BYTES.TO.A.WORD)) * BITS.TO.
A.BYTE


'Add a terminating 1 bit, all the rest of the bits to the end of the
    'word array will default to zero
    SHA256.Data(lWordCount) = SHA256.Data(lWordCount) Or _
        SHA256.LShift(HEXDEC("&H80"), lBytePosition)


'We put the length of the message in bits into the last two words, to
get

'the length in bits we need to multiply by 8 (or left shift 3). This l
eft

'shifted value is put in the last word. Any bits shifted off the left
edge

'need to be put in the penultimate word, we can work out which bits by
 shifting
    'right the length by 29 bits.
    SHA256.Data(lNumberOfWords - 1) = SHA256.LShift(lMessageLength, 3)
    SHA256.Data(lNumberOfWords - 2) = SHA256.RShift(
lMessageLength, 29)

End Sub

SUB SHA256.Initialize

    DIM SHA256.K(64)

    'Initialize the K array
    SHA256.K(0) = HEXDEC("&H428A2F98")
    SHA256.K(1) = HEXDEC("&H71374491")
```

```
SHA256.K(2) = HEXDEC("&HB5C0FBCF")
SHA256.K(3) = HEXDEC("&HE9B5DBA5")
SHA256.K(4) = HEXDEC("&H3956C25B")
SHA256.K(5) = HEXDEC("&H59F111F1")
SHA256.K(6) = HEXDEC("&H923F82A4")
SHA256.K(7) = HEXDEC("&HAB1C5ED5")
SHA256.K(8) = HEXDEC("&HD807AA98")
SHA256.K(9) = HEXDEC("&H12835B01")
SHA256.K(10) = HEXDEC("&H243185BE")
SHA256.K(11) = HEXDEC("&H550C7DC3")
SHA256.K(12) = HEXDEC("&H72BE5D74")
SHA256.K(13) = HEXDEC("&H80DEB1FE")
SHA256.K(14) = HEXDEC("&H9BDC06A7")
SHA256.K(15) = HEXDEC("&HC19BF174")
SHA256.K(16) = HEXDEC("&HE49B69C1")
SHA256.K(17) = HEXDEC("&HEFBE4786")
SHA256.K(18) = HEXDEC("&HFC19DC6")
SHA256.K(19) = HEXDEC("&H240CA1CC")
SHA256.K(20) = HEXDEC("&H2DE92C6F")
SHA256.K(21) = HEXDEC("&H4A7484AA")
SHA256.K(22) = HEXDEC("&H5CB0A9DC")
SHA256.K(23) = HEXDEC("&H76F988DA")
SHA256.K(24) = HEXDEC("&H983E5152")
SHA256.K(25) = HEXDEC("&HA831C66D")
SHA256.K(26) = HEXDEC("&HB00327C8")
SHA256.K(27) = HEXDEC("&HBF597FC7")
SHA256.K(28) = HEXDEC("&HC6E00BF3")
SHA256.K(29) = HEXDEC("&HD5A79147")
SHA256.K(30) = HEXDEC("&H6CA6351")
SHA256.K(31) = HEXDEC("&H14292967")
SHA256.K(32) = HEXDEC("&H27B70A85")
SHA256.K(33) = HEXDEC("&H2E1B2138")
SHA256.K(34) = HEXDEC("&H4D2C6DFC")
SHA256.K(35) = HEXDEC("&H53380D13")
SHA256.K(36) = HEXDEC("&H650A7354")
SHA256.K(37) = HEXDEC("&H766A0ABB")
SHA256.K(38) = HEXDEC("&H81C2C92E")
SHA256.K(39) = HEXDEC("&H92722C85")
SHA256.K(40) = HEXDEC("&HA2BFE8A1")
SHA256.K(41) = HEXDEC("&HA81A664B")
SHA256.K(42) = HEXDEC("&HC24B8B70")
SHA256.K(43) = HEXDEC("&HC76C51A3")
SHA256.K(44) = HEXDEC("&HD192E819")
SHA256.K(45) = HEXDEC("&HD6990624")
SHA256.K(46) = HEXDEC("&HF40E3585")
SHA256.K(47) = HEXDEC("&H106AA070")
```

```
    SHA256.K(48) = HEXDEC("&H19A4C116")
    SHA256.K(49) = HEXDEC("&H1E376C08")
    SHA256.K(50) = HEXDEC("&H2748774C")
    SHA256.K(51) = HEXDEC("&H34B0BCB5")
    SHA256.K(52) = HEXDEC("&H391C0CB3")
    SHA256.K(53) = HEXDEC("&H4ED8AA4A")
    SHA256.K(54) = HEXDEC("&H5B9CCA4F")
    SHA256.K(55) = HEXDEC("&H682E6FF3")
    SHA256.K(56) = HEXDEC("&H748F82EE")
    SHA256.K(57) = HEXDEC("&H78A5636F")
    SHA256.K(58) = HEXDEC("&H84C87814")
    SHA256.K(59) = HEXDEC("&H8CC70208")
    SHA256.K(60) = HEXDEC("&H90BEFFFA")
    SHA256.K(61) = HEXDEC("&HA4506CEB")
    SHA256.K(62) = HEXDEC("&HBEF9A3F7")
    SHA256.K(63) = HEXDEC("&HC67178F2")

END SUB

FUNCTION SHA256.Evaluate$(sMessage$)

    Dim SHA256.W(64)

    'Initialize the SHA256 object
    CALL SHA256.Initialize

    'Initial hash values
    h0 = HEXDEC("&H6A09E667")
    h1 = HEXDEC("&HBB67AE85")
    h2 = HEXDEC("&H3C6EF372")
    h3 = HEXDEC("&HA54FF53A")
    h4 = HEXDEC("&H510E527F")
    h5 = HEXDEC("&H9B05688C")
    h6 = HEXDEC("&H1F83D9AB")
    h7 = HEXDEC("&H5BE0CD19")

    'Convert the string into an
    'array of numeric values (32-bit)
    Call SHA256.ConvertToWordArray sMessage$

    'Evaluate the SHA256 digest
    For i = 0 To (SHA256.DataCount - 1) Step 16
        a = h0
        b = h1
        c = h2
        d = h3
```

```
        e = h4
        f = h5
        g = h6
        h = h7
        For j = 0 To 63
            If j < 16 Then
                SHA256.W(j) = SHA256.Data(j + i)
            Else
                SHA256.W(j) = SHA256.AddUnsigned(SHA256.
AddUnsigned(SHA256.AddUnsigned(SHA256.Gamma1(SHA256.W(j - 2)), _
                    SHA256.W(j - 7)), SHA256.Gamma0(SHA256.W(j - 15))
), SHA256.W(j - 16))
            End If
            t1 = SHA256.AddUnsigned(SHA256.AddUnsigned(SHA256.
AddUnsigned(SHA256.AddUnsigned(h, SHA256.Sigma1(e)), _
                SHA256.Ch(e, f, g)), SHA256.K(j)), SHA256.W(j))
            t2 = SHA256.AddUnsigned(SHA256.Sigma0(a), SHA256.Maj(
a, b, c))
            h = g
            g = f
            f = e
            e = SHA256.AddUnsigned(d, t1)
            d = c
            c = b
            b = a
            a = SHA256.AddUnsigned(t1, t2)
        Next
        h0 = SHA256.AddUnsigned(a, h0)
        h1 = SHA256.AddUnsigned(b, h1)
        h2 = SHA256.AddUnsigned(c, h2)
        h3 = SHA256.AddUnsigned(d, h3)
        h4 = SHA256.AddUnsigned(e, h4)
        h5 = SHA256.AddUnsigned(f, h5)
        h6 = SHA256.AddUnsigned(g, h6)
        h7 = SHA256.AddUnsigned(h, h7)
    Next

    'Output the 256-bit digest
    SHA256.Evaluate$ = Lower$(Right$("00000000" + DecHex$(h0), 8) + _
        Right$("00000000" + DecHex$(h1), 8) + _
        Right$("00000000" + DecHex$(h2), 8) + _
        Right$("00000000" + DecHex$(h3), 8) + _
        Right$("00000000" + DecHex$(h4), 8) + _
        Right$("00000000" + DecHex$(h5), 8) + _
        Right$("00000000" + DecHex$(h6), 8) + _
        Right$("00000000" + DecHex$(h7), 8))
```

```
END FUNCTION

FUNCTION SHA256.BitNot(X)

  SHA256.BitNot = 0
  For a = 0 To 31
    IF ((X And (2 ^ a)) = 0) THEN
      SHA256.BitNot = SHA256.BitNot Or (2 ^ a)
    END IF
  Next

END FUNCTION

FUNCTION SHA256.Ch(X, Y, z)
    SHA256.Ch = ((X And Y) Xor ((SHA256.BitNot(X)) And z))
END FUNCTION

FUNCTION SHA256.Maj(X, Y, z)
    SHA256.Maj = ((X And Y) Xor (X And z) Xor (Y And z))
END FUNCTION

FUNCTION SHA256.S(X, n)
    SHA256.S = (SHA256.RShift(X, n)) Or SHA256.LShift(X, (32 - n))
END FUNCTION

FUNCTION SHA256.R(X, n)
    SHA256.R = SHA256.RShift(X, n) 'CInt(n And mlOnBits(4)))
END FUNCTION

FUNCTION SHA256.Sigma0(X)
  SHA256.Sigma0 = (SHA256.S(X, 2) Xor SHA256.S(X, 13) Xor
 SHA256.S(X, 22))
END FUNCTION

FUNCTION SHA256.Sigma1(X)
  SHA256.Sigma1 = (SHA256.S(X, 6) Xor SHA256.S(X, 11) Xor
 SHA256.S(X, 25))
END FUNCTION

FUNCTION SHA256.Gamma0(X)
  SHA256.Gamma0 = (SHA256.S(X, 7) Xor SHA256.S(X, 18) Xor
 SHA256.R(X, 3))
END FUNCTION

FUNCTION SHA256.Gamma1(X)
```

```
  SHA256.Gamma1 = (SHA256.S(X, 17) Xor SHA256.S(X, 19) Xor
 SHA256.R(X, 10))
END FUNCTION

FUNCTION SHA256.AddUnsigned(lX, lY)
  SHA256.AddUnsigned = (lX + lY) AND HEXDEC("&HFFFFFFFF")
END FUNCTION

FUNCTION SHA256.RShift(lValue, iShiftBits)

  For a = 1 To iShiftBits
    lValue = lValue AND HEXDEC("&HFFFFFFFE")
    lValue = Int(lValue / 2)
    lValue = lValue And HEXDEC("&HFFFFFFFF")
  Next
  SHA256.RShift = lValue And HEXDEC("&HFFFFFFFF")

End Function

FUNCTION SHA256.LShift(lValue, iShiftBits)

  For a = 1 To iShiftBits
    lValue = lValue AND HEXDEC("&H7FFFFFFF")
    lValue = lValue * 2
    lValue = lValue And HEXDEC("&HFFFFFFFF")
  Next
  SHA256.LShift = lValue And HEXDEC("&HFFFFFFFF")

End Function
```