

## Re: Is there a simpler way?

**Source:**

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vc/2005-03/0623.html>

---

**From:** Severian ([severian\\_at\\_chlamydia-is-not-a-flower.com](mailto:severian_at_chlamydia-is-not-a-flower.com))

**Date:** 03/18/05

Date: Fri, 18 Mar 2005 18:10:43 GMT

On Thu, 17 Mar 2005 14:01:05 -0800, Fred Hebert <[fhebert@hotmail.com](mailto:fhebert@hotmail.com)> wrote:

>I am trying to learn VC.NET and convert some Borland C++ applications. The  
>syntax differences are killing me...

>

>Is there an easy way to convert a hex string, entered by the user, to a  
>binary string and back? This is the actual code I am trying to convert.

>

>HexToBin(KeyEdit->Text.c\_str(), BinKey, sizeof(BinKey));

>

>HexToBin is a generic function that converts a string like "1b34c38d" to  
>it's binary equivalent. There is also a BinToHex which reverses the  
>process.

>

>I know I could write a function to do it, but I threw all those routines  
>away 10 years ago when I started using Borland.

>

>I am hoping that someone can help, because I really don't want to have to  
>go back to writing all of those little conversion routines.

Binary value or binary string?

To convert a value <= ULONG\_MAX to an unsigned long binary value:

```
val = strtoul(strval, NULL, 0); // If strval as 0x1234abcd
```

```
val = strtoul(strval, NULL, 16); // if strval as 12341bcd
```

I'm sure there are STL and MFC classes that can do this too.

If you need to convert a hex string to a binary strings, I would use a table For efficiency (sample below), though some languages may provide built-in functions or extra formatting specifications.

Converting from hex to a binary value then to binary text might be faster, but this will handle arbitrarily-long hex and binary values.

Sample ANSI C code:

```
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
#include <stdio.h>

#define TESTING

#undef TRUE
#define TRUE 1
#undef FALSE
#define FALSE 0

/* Error codes */

#define eDIGIT 1
#define eTRUNC 2
#define eSEPAR 4

static int setdigit(int c, char *binval, int pos);

/* SevStrHexToBin()

Parameters:
    hexval – Hex value to convert. May be prefixed with h, H,
             x, X, 0x or 0X.
             May be suffixed with h, H, x, or X.
    groupchar – Group digits by this char (0 for no sep)
                groupchar is also ignored in the input string
                (groupchar cannot be any hex digit)
    binval – Where to place the binary string
    binmaxlen – Size of binval (with room for terminating \0)

Return values:
    0 – Success
    eTRUNC – binval not large enough
    eDIGIT – invalid hex character in input string
    eSEPAR – invalid groupchar
*/

int SevStrHexToBin(const char *hexval, int groupchar,
    char *binval, int binmaxlen)
{
    static const char hexdig[] = "0123456789abcdef";
    int e = 0, pos, c, diglen = groupchar ? 5 : 4;
    size_t hexlen = strlen(hexval);

    memset(binval, 0, binmaxlen); /* In case of error */
    if (!hexval || hexlen < 1)
```

Re: Is there a simpler way?

```

    return eDIGIT;
if (groupchar && strchr(hexdig, tolower((unsigned char)groupchar)))
    return eSEPAR;
if (hexlen >= 2 && hexval[0] == '0' && tolower(hexval[1]) == 'x')
    hexval += 2; /* Skip 0x if prepended */
else if ( (c = tolower(hexval[0])) == 'x' || c == 'h')
    hexval++; /* Skip x or h if prepended */
hexlen = strlen(hexval);
if (hexlen &&
    (((c = tolower(hexval[hexlen-1])) == 'x') || c == 'h'))
    hexlen--; /* Ignore trailing "hHxX" */
for (pos = 0; hexlen-- && ((c = *hexval++) != 0); pos += diglen) {
    if (pos + diglen - (groupchar && (hexlen == 0)) > binmaxlen)
        return e | eTRUNC;
    e |= setdigit(c, binval, pos);
    if (groupchar && hexlen != 0)
        binval[pos+4] = (char)groupchar;
}
return e;
}

```

```

static int setdigit(int c, char *binval, int pos)
{
    static const char errdig[] = "*****";
    static const char bindig[] =
        "0000 0001 0010 0011 " /* Spaces for readability */
        "0100 0101 0110 0111 "
        "1000 1001 1010 1011 "
        "1100 1101 1110 1111 ";
    int e = 0;

    if (c >='0' && c <='9')
        memcpy(binval+pos, bindig + 5*(c-'0'), 4); /* Decimal */
    else {
        c = tolower((unsigned char)c);
        if (c >='a' && c <='f')
            memcpy(binval+pos, bindig + 5*(c-'a'+10), 4); /* Alpha */
        else {
            memcpy(binval+pos, errdig, 4);
            e = eDIGIT; /* Invalid digit */
        }
    }
    return e;
}

```

```
#ifdef TESTING
```

```

static char *hexbinerrmsg(int e)
{
    switch (e) {
        case eDIGIT: return " (invalid)";
    }
}

```

```
case eTRUNC: return " (truncated)";
case eDIGIT|eTRUNC: return " (invalid & truncated)";
case eSEPAR: return " (invalid separator)";
default: return "";
}
}
```

```
int main(int argc, char *argv[])
{
    char binval[82] = "";
    int e = 0;

    if (argc < 2) {
        printf("%s\n", hexbinerrmsg(eDIGIT));
        return EXIT_FAILURE;
    }

    /* Simple conversion */

    e = SevStrHexToBin(argv[1], 0, binval, sizeof binval);
    printf("%s: %s%s\n", argv[1], binval, hexbinerrmsg(e));

    /* Space separators */

    e = SevStrHexToBin(argv[1], ' ', binval, sizeof binval);
    printf("%s: %s%s\n", argv[1], binval, hexbinerrmsg(e));

    /* Colon separators */

    e = SevStrHexToBin(argv[1], ':', binval, sizeof binval);
    printf("%s: %s%s\n", argv[1], binval, hexbinerrmsg(e));

    /* Invalid separators */

    e = SevStrHexToBin(argv[1], '0', binval, sizeof binval);
    printf("%s: %s%s\n", argv[1], binval, hexbinerrmsg(e));

    return EXIT_SUCCESS;
}
```

```
#endif
```

```
--
Sev
```