

Unsigned left shift in VB.NET?

Asked 14 years ago Modified 10 years, 8 months ago Viewed 3k times



This should be an easy one for folks, but how do I pull off an unsigned left shift in [VB.NET](#) while using `Option Strict`?

2



Maybe I am doing it wrong, but while trying to implement my own IP2Long function (I have my reasons), I'm testing things to make sure I have my head wrapped around the conversion process properly. I tried a few tests, and all seem to cause errors.



```
Dim a As Int32
a = CUInt(172 << 24) 'Constant expression not representable in type 'UInteger'
a = DirectCast((172 << 24), UInt32) 'Value of type 'Integer' cannot be
converted to 'UInteger'
a = Convert.ToUInt32(172 << 24) 'Compiles, but throws an OverflowException
```

The last one is especially befuddling. `172 << 24` is a mere 2,885,681,152, well under the limit imposed by the UInt32 data type. My assumption is .NET is doing the left-shift in signed mode, then tries to convert, to unsigned, and this tosses up some kind of error.

Basically, my question boils down to this: why do unsigned numerics have to act like such hacks to the .NET framework at times? Is it really that hard for Microsoft to make unsigned data types intrinsic to the framework?

`vb.net` `bit-shift`

Share

Improve this question

Follow

edited Apr 8, 2014 at 18:27



Peter Mortensen

31.6k ● 22 ● 109 ● 133

asked Dec 23, 2010 at 1:10



Kumba

2,428 ● 3 ● 36 ● 61

1 Answer

Sorted by: Highest score (default)



4



The last one is especially befuddling. `172 << 24` is a mere 2,885,681,152, well under the limit imposed by the UInt32 data type. My assumption is .NET is doing the left-shift in signed mode, then tries to convert, to unsigned, and this tosses up some kind of error.



This error is not wrong. 172 takes up 8 bits. You shift it 24 bits and the 8th bit is now the 32nd bit. This is a reserved sign bit. Therefore it is technically overflowing.



Treat integers like C#

VB.NET will check for an integer overflow unlike in **C#**.

To get **VB.NET** to ignore **OverflowExceptions** goto:

```
Project properties->Compile->Advanced Compiler Option->"Remove integer overflow checks"
```

Or [compile](#) with

```
vbc foo.vb /removeintchecks
```

Explicit state bit operations

If you are determined to leave overflow checks in your code, you have to explicitly tell it you are using bit operations using the [BitConverter](#) Class:

```
'//This will work  
a = BitConverter.ToInt32(BitConverter.GetBytes(172 << 24), 0)
```

Visual Basic literals

Also keep in mind you can add [literals](#) to your code in VB.NET and explicitly state constants as unsigned.

Share

Improve this answer

Follow

edited Apr 8, 2014 at 18:29



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Dec 23, 2010 at 1:16




user295190

- 1 Ah, you removed the bit about literals. That was actually a good answer. I thought literals were a GCC-specific thing, but I guess that's one of those little footnotes in VB.Net that's easily overlooked. Not too interested in disabling overflow checks, however. That doesn't work around the actual problem, just sweeps it under the rug. – [Kumba](#) Dec 23, 2010 at 1:38

@Kumba: I think it has to do more with the history of each language. C# is based upon C-like syntax where integers overflowed. Visual Basic is an introductory language and it might confuse new programmers if their integers went from 2,147,483,647 to -2,147,483,648.

– user295190 Dec 23, 2010 at 1:48

1 @snmcdonald: Agreed about the language differences. But this seems to extend into the framework itself. I've read that UInt32 is a structure of some type (I haven't investigated yet), rather than an intrinsic data type built into the CLR itself, and that unsigned values really aren't native to VB.NET (and C#). I suppose C# makes it easier (to an extent), since casting is `(datatype)Foo`, whereas in VB, you've got the ugly syntax of `DirectCast(Foo, datatype)`. – Kumba Dec 23, 2010 at 2:12

1 @Kumba: Even C# forces some explicit casting with the unchecked syntax: stackoverflow.com/questions/737781/... Although, I do not see that as a negative, I think it forces the programmer to be aware and enforces stronger typing discipline. Don't get me wrong I love C but it is considered a weakly typed language. – user295190 Dec 23, 2010 at 2:33 

1 @snmcdonald: Per the byte shift, I wasn't clear enough (these comment fields are really lacking at times on this site). What I meant was `172L << 24` is going to have a different answer than `172S << 24` (and ditto for 172 as a byte). Pause and think, and it makes perfect sense why this is so. But at a first glance, it isn't an immediate understanding. – Kumba Dec 23, 2010 at 2:41
