Sign Extend an Integer

Often times a value stored somewhere may not be the same width as an MMBasic Integer variable, being commonly 8, 9 or 16 bits wide. For example, the temperature register (&H11) of the popular DS3231 RTC chip is a two's-compliment byte. Reading (theoretically) +127 to -128C.

If this number is placed directly in an integer variable without sign extending bit 7 through to the end of the integer (bit 63) the temperature will be incorrect if negative - it will read over 128C. It is necessary to extend bit 7 of the register through the remaining 56 bits to obtain a true 64-bit integer value of the reading that will play nicely with integer maths... Of course this isn't necessary if you are not expecting sub-zero temperatures and are OK with taking the chance.

The following function will return a value sign-extended from any given bit. There is no error checking - if you feed it rubbish or try to crash it you'll probably succeed.

Syntax:

=SgnX(Bit,Value)

Examples: (better understood looking at the binary)

```
Print Bin$(SgnX(7,&h7f),64)
```

<pre> 0000000000000000000000000000000000000000000000000000000001111111

```
                                                        ^        bit 7
```

</pre>

```
Print Bin$(SgnX(7,&h8f),64)
```

<pre> 1111111111111111111111111111111111111111111111111111111110001111

```
                                                        ^        bit 7
```

</pre>

Consider the following:

```
RTC GetReg 17,x
```

Here x contains the 8-bit, two's compliment value for the temperature with bit 7 being the sign (1=negative). But, as the remaining (top) 56 bits of x are zero when placed in a variable, the value will always be treated as positive by MMBasic because its sign bit, 63, is zero. Suppose the value read was 254 (&hFE), which is -2 degrees. When assigned to the variable x, the value is placed in the lower n bits (in this case 8) and so will be treated as the absolute value of the register because all bits above 7 will be set to zero - including the integer sign bit, 63.

```
Print Bin$(x,64)
```

```
0000000000000000000000000000000000000000000000000000000011111110

Print x
254
```

To correct this, the sign from the register value (here, bit 7) needs to be extended leftwards through (copied-to) all remaining 56 bits from 8 to 63 which then makes the integer variable x hold the correct 64 bit version of the original value:

```
x=SgnX(7,x) 'extends (i.e. copies) bit 7 all the way to the top

Print Bin$(x,64)
1111111111111111111111111111111111111111111111111111111111111110

Print x
-2 ' got the intended value
```

As another example, if a value read is 9 bits, to be interpreted as two's compliment - 8 bits for the value plus sign bit, simply extend bit 9, or for a traditional 16 bit signed integer, extend bit 15 etc. The function will take any bit as the sign and return a correctly formatted 64 bit MMbasic integer.

Code:

```
    Function SgnX(b As Integer,v As Integer) As Integer' sign extend bit b
of value v
        If v And 2^b Then 'extend 1
            SgnX=v Or (-1 << (b+1))
        Else 'extend 0
            SgnX= v And ((2^(b+1))-1)
        EndIf
    End Function
```