Release Notes for XXICC revision 0.0a

This document describes enhancements and bug fixes for XXICC revision 0.0a, released in August 2011.

1. GalaxC Language

- We have added several new GalaxC operators to make programs more readable (we hope) and concise. GalaxC requires a Boolean expression in *if-then* and *while-do* statements, whereas C allows the condition expression to be an `int` or a pointer. Thus GalaxC requires the statement:

```
if x != 0 then ...
```

  instead of C's more concise:

```
if (x) ...
```

  We have added the '`?`' unary operator to convert a `ulong` expression to Boolean so you can write:

```
if ?x then ...
```

  '`?`' is the counterpart of '`!`' and follows the same syntax rules. Here is how they are defined in `gxclib.gal` for `ulong` and subtypes:

```
ulong arg x,
Boolean inline {!x} = intrinsic NOT,    // Equivalent to x == 0.
Boolean inline {?x} = intrinsic NEZ;    // Equivalent to x != 0.
```

  Here is how they are defined for pointers:

```
@T arg p,
def !p = !address(p),       // Equivalent to "p == NULL".
def ?p = ?address(p),       // Equivalent to "p != NULL".
```

- We have also dcfined new operators for testing whether bits are set in a `ulong` value. Instead of writing "`(x & 0x100) != 0`" you can now write "`x &? 0x100`" and instead of "`(x & 0x100) == 0`" you can write "`x !& 0x100`". The '`&?`' operator is Bit Test (`BIT`) and the '`!&`' operator is `NOT AND` (`NAND`). Here is how they are defined in `gxclib.gal`:

```
ulong arg {x, y},
def {x &? y}  = ?(x & y),       // Bit Test: x intersects y.
def {x !& y}  = !(x & y);       // NAND: x does not intersect y.
```

  These operators are useful for checking the presence of status and option bits.

- We have added mathematical symbols from the Symbol font to supplement the standard C relational symbols. These can improve the appearance of GalaxC programs, but remember that not all users

may have access to the Symbol font. Here are the current definitions from `gxclib.gal`:

```
spclarg {a, b},
def a ≡ b = a == b,     // Symbol ≡ is equivalent to '=='.
def a ≠ b = a != b,     // Symbol ≠ is equivalent to '!='.
def a ≤ b = a <= b,     // Symbol ≤ is equivalent to '<='.
def a ≥ b = a >= b;     // Symbol ≥ is equivalent to '>='.
```

Using `unknown spclargs` makes these generic macro definitions so they can be used with relational operators for all types. We also define π as a `double` floating-point number:

```
def π = 3.14159265358979323846;
```

These updates are in *Programming in the GalaxC Language*, revision 0.0a.

2. GalaxC Compiler

The only significant change is that we have added a prototype Make capability to GalaxC's compiler. For details, see *Compiling GalaxC Programs* revision 0.0a.

3. XXICC Object Editor

Using BREAK XOs, you can now indicate that text should be in an Exclusion Class, which may or may not be printed. We use this feature for copyright and licensing notices that are included when printing a single file, but excluded when printing a complete book since the notices are at the front of the book. For details, see Chapter 5 of *The XXICC Anthology* revision 0.0a.

4. Bug Fixes

- Issue #1 "In XOE, identation may sometimes affect rest of file" may be fixed. We recently found a way to reproduce one manifestation of the bug, which was trivial to fix once reproducible. However, there may be other bugs that produce similar effects so we're keeping the issue open for now.

- Issue #5 "In XOE, Describe list items do not have orphan control" is probably fixed.

- We have fixed a trivial bug that prevented dynamic link libraries (`.dll` and `.so`) from being loaded reliably from the command line. This is now documented in *Installing and Running XXICC* revision 0.0a, §4.