Known Bugs in ETEC Version 2.22

Bug Identifier	Source	Problem/Bug Description	Severity	Workaround Description	Affected Releases	Fixed Release
V1.00D-5 (2009-Dec-15)	internal	When the sizeof operator is applied to a constant the wrong size may result, e.g. sizeof(1) may result in "1" rather than the expected "3" bytes.	2	Take the sizeof the desired type instead: sizeof(int)	All versions	TBD
V1.20A-14 (2009-May-20)	internal	Chan interrupt opcodes may be moved relative to adjacent RAM instructions by the optimizer. This may cause unexpected results, particularly in the case of a DMA interrupt.	3	Use _OptimizationBoundaryAll() or #pragma opimization_boundary_all if there is concern that an interrupt may cross a critical RAM access.	All versions	TBD
V1.25A-11 (2009-Sep-28)	internal	If pointer arithmetic generates a negative result, and the object pointed to is larger than 1 byte in size, ETEC code will generate an incorrect result. This is because an unsigned shift (or unsigned divide) is applied after the pointer arithmetic to convert from byte addressing to object indexing.	3	Keep pointer arithmetic results in the non-negative domain.	All versions	TBD
V1.25B-6 (2009-Dec-9)	internal	The _STACK_SIZE_ defines macro gets the calculated value of the worst-case stack depth. In certain rare cases, this value can be slightly larger than the actual worst-case. This can occur when a stack usage of a register save and restore (e.g. in a called C function) is eliminated via optimization. Such a register save requires 4 bytes of stack space, but the removal of it is not currently getting accounted for in the stack size calculation.	4	Care should be taken in that in some rare cases, a _STACK_SIZE_ value that is non-zero can still mean that no stack is actually utilized. Another way to verify that no stack is used is to make sure that no <func/class name>__STACKBASE_ macros are defined.	All versions	TBD

V1.25B-7 (2009-Dec-11)	internal & customer	The optimizer/analyzer does not yet support reentrant functions, whether they be callable C functions or ETEC code fragments. Reentrance is supposed to be detected and cause an error, but in some cases this detection failed, allowing for optimization to continue. Sometimes the result could be a linker crash, or sometimes invalid code generation, or in some cases working code resulted.	3	Avoid writing reentrant functions until the ETEC optimizer/analyzer fully supports them.	All versions	V1.25C (reentrance detection), TBD (support reentrance)
V2.22A-1 (2013-Dec-4)	internal	Win 7 users may experience issues with some demos due to the demo installation location and file access permissions.	4	The work-around is to either (1) change the permissions on the demo directories located under the main program installation directory to full access for all users, or (2) copy the demos to a location that does not have access issues; the compiler demo Mk.bat build scripts will need to be adjusted to properly find the tools.	All versions	V2.22B
V2.22A-2 (2013-Dec-10)	customer	C preprocessing of the same files from the same working directory simultaneously (i.e. running multiple processes at the same time) can in some cases result in a conflict that crashes the C preprocess task.	4	Do not simultaneously run compilations in the same working directory that acess the same source files, or simulations in the same working directory that acess the same script files.	All versions	V2.22B

V2.22B-1 (2014-Jan-30)	customer	A class of compilation optimizations that look for duplicate expressions can have a problem across function calls in some cases, if the expression in question is a function parameter and used just following the function call. An example of potential problem code is: array[index] = some_func(array[index]); The address of 'array[index]' gets computed just once but in some cases is not saved properly across the function call. A function call was being treated as an immediate block rather than an after block (after its children - its parameters - were processed).	2	Adding the command line option "-optDis=0x20" to the compilation command of the affected file disables all duplicate expression optimizations. Contact Ashware as disabling just a single optimization type with values 0x21, 0x22 or 0x23 is likely to be a better approach.	V2.00A - V2.22B	V2.23A

Bug Severity Level Descriptions:

1 – Problem causes complete work stoppage. No work-around is possible. The problem is likely to be hit by most users. This level of bug will typically trigger a new release or patch in a short time frame.

2 – A difficult problem to track down, such as incorrectly generated code. Typically there is a work-around available for this kind of bug.

3 – A bug that is easy to spot, and/or generally has a straight-forward work-around, or has minimal impact.

4 – Not truly a bug (i.e. tool is within spec.), but rather something that might affect compatibility or usability. Work-arounds available.