Known Bugs in ETEC Version 2.40

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)
V2.23B-5 (2014-Mar-18)	internal	In some cases when making a fragment call, and the fragment is contiguous with the calling code (i.e. jump can be eliminated), the link-time optimizer mistakenly optimizes out code it should not.	2	This situation, if encountered, can be corrected by re-arranging the code to prevent the fragment call and fragment code from being continguous.	V2.00A and newer	TBD
V2.23B-7 (2014-Jun-6)	customer	The C preprocessor is currently allowing the same macro to be expanded in multiple replacement passes, which causes the preprocessor to break when such "recursion" is encountered.	3	Avoid self-referencing preprocessor macros.	All versions	TBD
V2.40A-1 (2015-Jul-9)	customer	The C preprocessor issues a false error when a file ends with a string or character literal, with no space or newline following the " or ' character.	4	Add a newline / carriage return to end the file (the C99 standard actually says a source file not ending in such a way is malformed).	All versions	V2.40C
V2.40A-2 (2015-Jul-16)	customer	The eTPU has a 1K global memory physical limit. In certain cases when this limit is exceeded no error message generated and instead an internal diagnostic message is generated or the linker can crash. An error message is now always generated when the limit is exceeded.	3	Reduce global memory & scratchpad usage below the 1KB limit.	All versions	V2.40C
V2.40A-3 (2015-Jul-20)	internal	In some cases having a trailing backslash on an include path (as specified by the -I option) was causing a crash when the C Preprocessor was called.	4	Remove trailing slash from include path.	All versions	V2.40C

V2.40A-4 (2015-Jul-20)	internal	If an include file in a different directory than the original source references an include file back in the original source directory, a compiler crash could result.	3	Avoid include files in other directories referencing include files in the original source directory. Or easier, add "." to the include search path (-i=.).	V2.40A	V2.40C
V2.40A-5 (2015-Jul-22)	internal	Initializing a pointer variable to a non-zero constant value was resulting in a crash.	3	Initialize the pointer value at runtime.	All versions	V2.40C
V2.40A-6 (2015-Jul-29)	customer	The map file scratchpad section is listing some scratchpad variables that have been optimized away.	4	Ignore bad information in map file.	All versions	V2.40C

V2.40A-7 (2015-Aug-2)	internal	A case has been found where a register used to pass a function parameter can incorrectly get re-used for a local variable, resulting in incorrect code generation.	2	Should the problem manifest, the work-around is to make use of the parameter variable before the local variable with which the conflict occurs - this can be accomplished with a dummy transfer. E.g.
				
				void SomeFunction(int a)
				{
				a = a; // make use of parameter variable a before any other variable usage
				// ...
				}
					V2.00A and newer	V2.40C

V2.40C-1 (2015-Aug-14)	internal	Functions with 32-bit return values are failing to compile by issuing a false error (32-bit int types), or by generating incorrect code (32-bit tag/struct type).	3	The workaround is to pass the data back via a pointer parameter, rather than as a return value.	All versions	V2.40E
V2.40C-2 (2015-Aug-14)	internal	When a very large eTPU code base is linked, especially if the code contains very long identifiers and man external declarations, there is the possibility of a linker crash due to buffer overflow in ELF/DWARF2 file generation.	3	Suppress ELF file output, if not needed for simulation or debug. Reduce unnecessary external declarations.	All versions	V2.40E

V2.40C-3 (2015-Aug-24)	customer	Under some conditions an extra space gets added to the replacement list of macro definitions, resulting in an extra space at each later replacement of the macro.	4	Leave no spaces after the replacement list in the macro definition.	All versions	V2.40E

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.