

## Known Bugs in ETEC Version 2.41

| Bug Identifier             | Source   | Problem/Bug Description  | Severity | Workaround Description   | Affected Releases | Fixed Release |
|----------------------------|----------|--|----------|--|-------------------|---------------|
| V1.00D-5<br>(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 of the desired type instead: sizeof(int)   | All versions      | TBD           |
| V1.20A-14<br>(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 <code>_OptimizationBoundaryAll()</code> or <code>#pragma optimization_boundary_all</code> if there is concern that an interrupt may cross a critical RAM access.   | All versions      | TBD           |
| V1.25A-11<br>(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<br>(2009-Dec-9)   | internal | The <code>_STACK_SIZE_</code> 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 <code>_STACK_SIZE_</code> 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 <code>&lt;func/class name&gt;_STACKBASE_</code> macros are defined. | All versions      | TBD           |

|                           |                     |  |   |   |                  |   |
|---------------------------|---------------------|--|---|---|------------------|---|
| V1.25B-7<br>(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<br>(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 contiguous. | V2.00A and newer | TBD   |
| V2.23B-7<br>(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   |
| V2.41C-1<br>(2015-Nov-30) | customer            | The auto-generated code (<>_scm.c) and initialized data (<>_idata.c) 'C' files contains #include directives of their associated .h files. These #include directives are mistakenly using any output path specified via the -out=<path> linker option, which can make them difficult to use as-is. The fix will involve ensuring that no path is output in these #include of the <>_scm.h and <>_idata.h files. | 4 | Either do not specify an output path via the -out=<path> linker option, or do not use the auto-generated <>_scm.c and <>_idata.c files.         | All versions     | V2.42A  |
| V2.41C-2<br>(2015-Dec-20) | internal            | If an _eTPU_constant_table definition is preceeded by its extern declaration, compilation fails.   | 3 | If an extern declaration of an _eTPU_constant_table is needed, do not include this declaration before the definition.                           | All versions     | V2.42A  |

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.