DISCLAIMER

Compiler Personality Generator (CPG) assists the user in constructing compiler personalities. CPG is not guaranteed to produce personalities that are 100% correct, and therefore users are expected to cross-check each item in a generated personality with the documentation for the particular compiler environment under consideration.

Please note that use of incorrect personalities may lead to incorrect analysis results, masking defects in code bases.

Compiler Include Paths - If the compiler you are generating a personality for is not configured correctly, CPG may not be able to deduce the include paths.

The include paths can be added to the exe command string in the CPG wizard in quotes.

Compiler Include Path Order – If a compiler has multiple include directories (e.g. sub directories for different targets that have an identical header file structure), CPG may get the order of the include directories wrong. The user is alerted to this in the output log.

Compiler Defines - In the case where "cross-compiler" is selected, and the compiler does not provide sufficient error output regarding the value of a define, the CPG may not be able to determine correctly the values of pre-defined macros. The user will be alerted to this in the output log, and can add the entries in by hand to the compiler personality. Note the values of these macros will also have to be determined manually.

Intrinsic Type size (size_t, wchar_t, ptrdiff_t) - if the compiler does not have any implicit include paths or if the intrinsic types are not defined in the standard location (stddef, stdarg or stdio.h) then the CPG will report that it cannot deduce these types correctly. These should be reviewed.

Bit Field Interpretation - The test for bit-field interpretation being signed or unsigned, will only be conducted if "cross compiler" is not selected. If "cross compiler" is selected, the default value (unsigned) will be used.

This may lead to incorrect results on certain cross compilers and should be reviewed in these cases.

Non-standard (Extension) Keywords - A keyword that is not handled by the correct context test may be given an "ignore" value in the resulting compiler personality which is not correct. The CPG will alert the user to the definite situations where it could not deduce the nature of a keyword extension.

Non-standard (Extension) Keywords - in the situation where a non-standard keyword extension can be used in multiple contexts (i.e. __interrupt and __interrupt(1)) the CPG will select the "ignore" value corresponding to the first test that successfully compiles (as per name order in the *_keyword_tests directory). In this case it may require further manual review of the keyword and an alternative mechanism to handle it (i.e. substitute headers).

Non-standard (Extension) Keywords - on occasion the CPG may incorrectly add in the handling of a standard system function using the "ignore" options. This is due to the situation where a compiler has an implicit list of "known" functions that the names cannot be re-used (even if no system header files are included), and the keyword tests fail due to a "redefinition". The CPG looks for the word "redefinition" in the output of the test, however this is not guaranteed to catch the situation. The compiler personality may need to be reviewed to remove any entries that look like standard function names (i.e. strcmp=_ignore_paren would not be correct in a generated compiler personality, and this is reached as compiling the file with only the line "int strcmp;" fails due to an implicitly known redefinition of a standard function).

Non-standard (Extension) Keywords - another situation is when a compiler requires an accompanying keyword or pragma in order to enable an extension keyword. For example some embedded C compilers require #pragma asm_enabled to be present in a source file to allow the use of inline assembler (via the asm keyword) inside a function. The CPG cannot add in #pragma's to standard test files as this will not be generic to all compilers, and hence it may not successfully produce an "ignore" result for keywords that rely on accompanying keywords.

Non-standard (Extension) keyword Data Types - in the situation where a compiler implicitly has an extension keyword data type (i.e. __int64) and the size of this extension datatype matches more than one base data type size (i.e. int and long are equal and __int64 matches this size), then the CPG must select one of the base data types to apply. In this case the CPG selects the first datatype in the list (ordered from char - int - short --> long double) of the two base types, note that it does use int in precedence over short to try to avoid any further implicit conversion messages. This may need to be reviewed.

External Identifier Significance Length - CPG tests for identifier significance by compiling test files with varying length identifiers and noting the length of the identifier when the file no longer compiles (e.g. extern int aa; extern int ab; ... extern int aaaaa; extern int aaaab; ... etc.). There may be the case in some environments where the compiler has a different significance length than the linker. In this situation the CPG will produce a result as provided by the compilation of the file rather than other programs such as the linker.