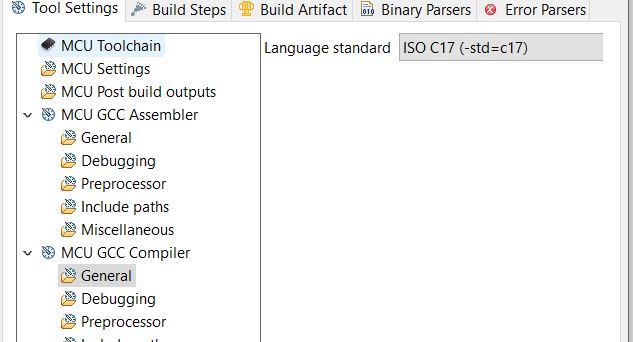
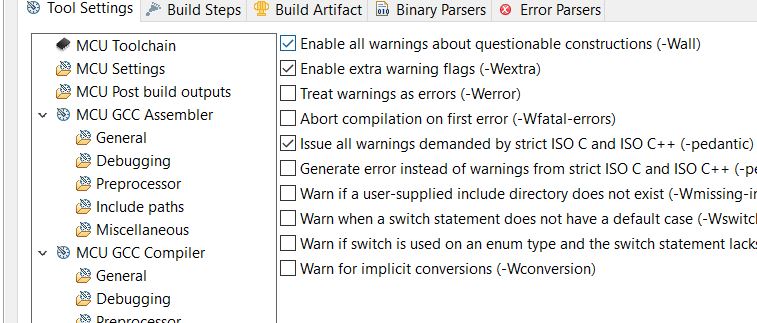
Alex Menelli, Nick Marsalek, Logan Young

JPL Compliance

**1 Language Compliance**

1. Do not stray outside the language definition
   1. 
2. Compile with all warning enable; use static source code analyzers
   1. 

**2 Predictable Execution**

1. Use verifiable loop bounds for all loops meant to be terminating
   1. No while(1) loops outside either player working
2. Do not use direct or indirect recursion
   1. No function calls itself
3. Do not use dynamic memory allocation after task initialization
   1. No use of malloc/free at all
4. Use IPC messages for task communication
   1. There is a queue used for communication
5. Do not use task delays for task synchronization
   1. No loops used to delay in any function. Timing controlled by pong\_main
6. Explicitly transfer write-permissions (ownership) for shared data objects
   1. Ownership of pong\_game and all other objects are from pong\_main. All functions except one write to it. Const used when needed.
7. Place restrictions on the use of semaphores and locks
   1. None used.
8. Use memory protection, safety margins, barrier patterns
   1. See ram\_health checks
9. Do not use goto, setjmp or longjmp
   1. Done in library, assembly is not used in code
10. Do not use selective value assignments to the elements of an enum list.
    1. Look into pong\_enums.h and pressed.h for all enums

**3 Defensive Coding**

1. Declare data objects at the smallest possible level of scope
   1. Ok! Check pong\_main
2. Check the return value of non-void functions, or explicitly cast to void.
   1. Ok! Check pong\_gameplay, pong\_main
3. Check the validity of values passed to functions
   1. Ok! Check pong\_gameplay and pong\_main
4. Use static and dynamic assertions as sanity checks
   1. Check pong\_main for some assertions
5. Use U32, I16, etc instead of predefined C data types such as int, short, etc
   1. Violated, check circle\_queue.h for violation note for head and tail.
6. Make the order of evaluation in compound expressions explicit.
   1. Sounds good!
7. Do not use expressions with side effects
   1. No expressions have side effects

**4. Code Clarity**

1. Make very limited use of the C pre-processor.
   1. Only one use of pre-processor in circle\_queue.h
2. Do not define macros within a function or a block
   1. One macro defined within a header file
3. Not done
4. Not done
5. Not done
6. Not done
7. Not done
8. Not done
9. \*Do not use non-constant function pointers
   1. Any violations are documented in the code. Function pointer work as intended.
10. Do not cast function pointers into other types.
    1. Function pointer are not cast into other types.
11. Do no place code or declarations before and #include directive.
    1. #include statements are the first statements in every files.