4-1 Activity: Exceptions

Paloma Rodriguez

CS-350: Emerging Sys Arch & Tech

November 12th, 2023

In Embedded C programming, sticking to best coding practices is very important for ensuring the safety, security, and reliability of the codebase, as emphasized by Richard Bellairs (2018). One crucial feature involves proficient bit manipulation which is use of bitwise operators and operations to efficiently manipulate individual bits within a binary representation. Bit manipulation encourages developers to use these operators thoughtfully and document the purpose for enhanced code clarity. The planned usage of the 'volatile' keyword is another important practice, especially in cases where variables are shared between an interrupt service routine and the main code to prevent unwanted compiler transformations. Thinking about how we use memory highlights the need to make the best use of global variables, avoid using too much dynamic memory, and efficiently use memory-mapped peripherals to control devices. Then cross-compiling, a common practice in embedded systems makes it necessary to carefully select compilers and maintain version consistency across the development environment. Equal importance should be shown in the implementation of finite state machines to ensure sturdy control flow with detailed definition and documentation of states and transitions. We should be careful about where we use variables in our code to make it easier to read and maintain. It's better to use fewer global variables and instead use static variables inside functions thoughtfully. Also stack management considerations involve monitoring stack usage to prevent overflow, avoiding deep function call nesting, and utilizing stack analysis tools. However, despite all of these best practices we could implement, there are common downfalls that developers should be extremely cautious of. Not setting up variables properly can cause problems in your code, and using memory in a not in a good way can make the computer use more power and work slower. It's important to deal with interruptions, test your code well, and explain your code to avoid issues like interruptions, hidden mistakes, and overall, not getting confused. Also, it’s best to think about time limits and make sure your system works well in real life situations. If there's an error, it's crucial to handle it and fix it properly.