Back to the roots: Tracing and debugging as a way to increase efficiency

Debugging embedded software is often a time-consuming activity, both in terms of chasing down a specific bug and as a general project activity. Further, as an activity, it is often an eclectic mix of desperation, perspiration, and a fair bit of magical thinking. In this article, I will cover techniques and tactics that might not completely eliminate all the hassles of debugging but can at least minimize the magical part. If you are a relative newcomer to the embedded software world, you might pick up some useful nuggets of information. If you are a seasoned pro, you are probably aware of these topics, but you might then re-discover some techniques that you already know that you should practice.

Code quality as a base

We know for a fact that newly written software is seldom, if ever, completely bug-free. However, we also know that there are actions we can take up front to help us reduce the number of issues we have to deal with in our code, which is another way to say that we have less debugging to do. An obvious place to start is to lay down some basic rules for code hygiene. Here is a summary of some rules:

- Give some extra thought to how you use memory. For example, do you really need dynamic memory management? Is the stack really a good place to store complex data structures? Standards for functional safety and high-integrity software often advise strongly against dynamic memory management, and storing complex or large data structures on the stack, and these are for good reasons.
- If your toolchain supports worst-case stack depth analysis, the investment to read up on and use that functionality pays off quickly.

The power of breakpoints

So, let us take a break from the preaching to look at the different types of breakpoint available. A breakpoint can, in its simplest form, be a stop sign at a particular source statement, so execution breaks unconditionally when reaching the right spot. A decent debugger will then let you examine the content of variables, registers, and the call stack as well as memory in general.

Software Engineering | Debugging

In the context of software engineering, debugging is the process of fixing a bug in the software. In other words, it refers to identifying, analyzing, and removing errors. This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software. It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

Debugging Process: Steps involved in debugging are:

- Problem identification and report preparation.
- Assigning the report to the software engineer to the defect to verify that it is genuine.
- Defect Analysis using modeling, documentation, finding and testing candidate flaws, etc.