These tutorials are a simplified introduction, and are not sufficient on their own to achieve system safety. You are responsible for the safety of your system.

11/8/2018

Good Fences

Make

Good Neighbors

© 2017 Philip Koopman
Critical System Isolation

Anti-Patterns for Isolation:

- Low-SIL software can access critical data
- Low-SIL software can block critical tasks

Need isolation between different SILs

- Lower SIL assumed to compromise High SIL
  - Higher SIL $\rightarrow$ “trusted” (critical tasks)
  - Lower SIL $\rightarrow$ “untrusted” (non-critical tasks)
    » Corrupts high-SIL data values, timing, configuration
- Hardware isolation is best option
  - Different SILs separated on different chips
  - Different networks for safety vs. non-safety data
    » Network data exchange is safety critical
Mixed-SIL Interference Examples

- **Memory value interference**
  - Non-critical task modifies critical variables
  - Non-critical ISR causes critical task stack overflow
  - Non-critical task memory leak; heap exhaustion

- **CPU time interference**
  - Non-critical task runs at high priority; starves critical tasks
  - Non-critical task disables interrupts; delaying critical tasks

- **Watchdog timer**
  - Non-critical task kicks watchdog regularly
  - Non-critical task disables watchdog

- **System configuration**
  - Non-critical task changes digital output to input

- **Network**
  - Non-critical node sends unsafe critical message
Mitigating Cross-SIL Interference

- **Develop all software at highest SIL**
  - Avoids isolation, but increases expense

- **Hardware solution – separate CPU chips**
  - Multi-core provides only partial isolation

- **High-SIL RTOS approaches**
  - Hardware memory protection (MMU)
  - Hardware CPU time isolation (e.g., multi-core)
  - Virtualization of I/O and configuration

- **Other techniques can help for Low-SIL**
  - Variable mirroring (two one’s complement copies)
  - Critical tasks run at high priorities or in ISRs
  - Non-modifiable watchdog timer configuration

- **Self-test is insufficient for High-SIL integrity**
  - Fault in high SIL hardware can subvert self-test
Isolation and Security

- **Lower-SIL task is ~ a malicious attacker**
  - How can it disrupt higher-SIL software?
  - Consider:
    - memory corruption, timing, configuration, network

- **Implications for safety:**
  - A weaker fault model means making assumptions
  - Lower-SIL update means revisiting assumptions

- **Implications for security:**
  - Higher-SIL functions more resistant to attack if isolated
  - Bad pattern: everything on one CPU with desktop OS
  - Better pattern: isolated CPUs with high-SIL critical RTOS

http://i.imgur.com/rGtgr.jpg
Best Practices For Critical System Isolation

- Use as much hardware isolation as you can
  - Consider:
    - Data value isolation
    - CPU time isolation
    - Configuration corruption
    - Shared resource isolation
  - Applies to any different SILs
    - Crucial for non-SIL $\Leftrightarrow$ SIL 3/4

- Pitfalls:
  - Multi-core CPU isn’t enough on its own (other shared resources!)
  - IEC 60730: Arguing that low-SIL software won’t interfere...
    ... requires re-arguing after every low-SIL change

https://goo.gl/6kFQb9