

Reliability assessment of FreeRTOS in Embedded Systems

Original

Reliability assessment of FreeRTOS in Embedded Systems / Bosio, A.; Rebaudengo, M.; Savino, A.. - ELETTRONICO. - (2022), pp. 28-30. (Intervento presentato al convegno 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks tenutosi a Baltimore, MD (USA) nel 27-30 June 2022) [10.1109/DSN-S54099.2022.00019].

Availability:

This version is available at: 11583/2971557 since: 2022-09-21T12:46:52Z

Publisher:

IEEE

Published

DOI:10.1109/DSN-S54099.2022.00019

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

IEEE postprint/Author's Accepted Manuscript

©2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

time window is user-defined, and the reported experimental results refer to a dynamic time window defined as 50% contribution in crashes, freezes, and delays vary a lot among more of the expected timing of the more extended tasks.

- **Delays:** if the execution presents no error, but the time exceeds the expected one, the difference can be interpreted as a missing deadline.

IV. EXPERIMENTAL RESULTS

Experimental data have been gathered resorting to the EEMBC® Automotive suite [2] to select the benchmark tasks. The complete set of benchmarks comprises 16 applications and, among them, three models have been extensively tested:

- **a2time** (Angle to time conversion): it simulates a counter which measures the real-time delay between pulses sensed from a toothed wheel (gear) on the crankshaft of an engine.
- **tblook** (Table lookup and interpolation): it stores a limited amount of data pairs coming from one or more resources (sensors, connections, calculations) and interpolates missing pairs.
- **idctrn** (Inverse Discrete Cosine Transform): the Inverse Discrete Cosine Transform is applied to an input data-set representing a matrix of 64 bits values.

The setup is completed by two other dummy tasks that are performing operations on some hardware resource that has been treated as a shared one to stress the synchronization mechanism also for single-threaded reference benchmarks, i.e. the **a2time** and the **tblook**.

Injection campaigns have been carried out by injecting faults per location list. Their number follows the approach presented in [3] to obtain statistically significant results with an error margin of 1% and a confidence level of 95%.

A. Results Analysis

Since there was no evidence that SDCs have been produced among all experimental results, all further classifications do not include the SDC label for simplicity.

Looking more closely, the experimental results demonstrate that the resiliency of a system depends on the OS too. They clearly show how the FreeRTOS is affected by some vulnerabilities that lead to crashes. Figure 1 shows that the effects are statistically very similar among all benchmarks. This is not entirely unexpected because most fault locations are pointers to memory structures and indexes used to access and index elements of lists or vectors. Moreover, the low number of freezes probably relates to the injection location, making the system more efficient to crash instead of leaving it in an endless loop of unresponsive tasks altogether.

Figure 1 also suggests that the overall number of crashes and freezes does not depend on the benchmark, which is consistent with the fact that no internal memory of the task cycle has been altered. On the contrary, delays are likely linked with the specificity of the benchmark, as the difference between **tblook** and the other two suggests. Still, on the broadside, it must be said that it also gives a wrong perspective among the several target groups, as Figure 2 is highlighting:

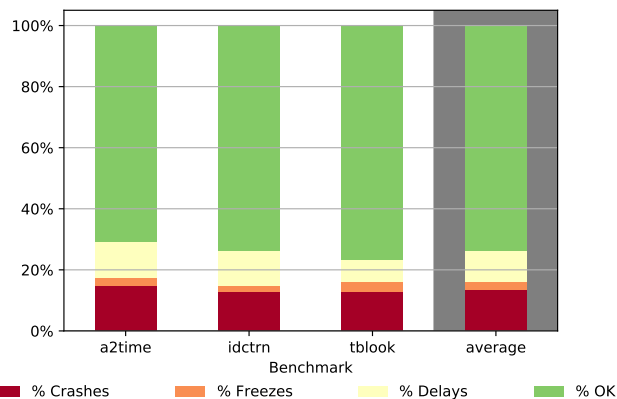


Fig. 1: Classification for all target locations by benchmarks

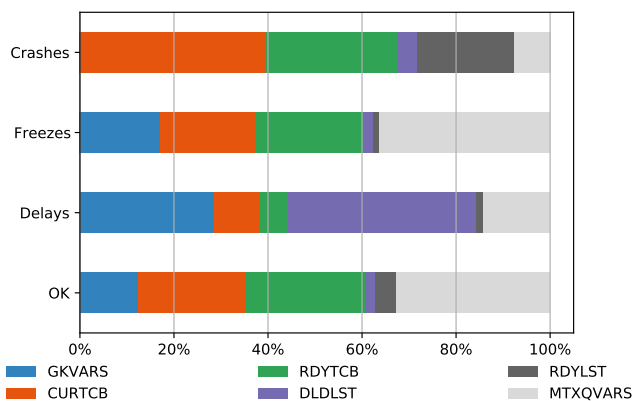


Fig. 2: Classification for all different target location groups

Figure 2 reports the classification across all six target location groups to appreciate the sensibility to the fault injection. It can be easily noticed how the **GKVARS** group does generate crashes, as **CURTCB** (almost 40%), **RDYTCB** and **RDYLST** do. Freezes are equally generated by alteration on any OS structure and the **RDYLST** and **DLDLST**. This is not forthcoming and probably relates to the high number of crashes when the **RDYLST** is corrupted and the even a higher number of delays produced by the **DLDLST** corruption. Regarding the delays, they also come from the **GKVARS** group, revealing a certain level of resiliency of the TCBs. When the corruption affects (parts of) the TCB that are not directly involved in the next cycle of scheduling, the outcome is still a properly running task but not necessarily the expected one. Eventually, the **MTXQVARS** group generates mainly freezes, delays, or no alteration at all, which confirms the accuracy of the tool since the **MTXQVARS** group includes all locations controlling mutex and semaphores.

V. CONCLUSIONS

This paper evaluates the dependability of FreeRTOS to SEU by building a Fault Injection Environment able to target data belonging to the operating system and trace the effects on the entire system. The results display the wide variety of impacts that FreeRTOS data structure disruption can lead to and allow us to suggest which are more sensitive to SEUs.

REFERENCES

- [1] Freertos. [Online]. Available: <https://www.freertos.org/index.html>
- [2] EEMBC, *Autobench - Software benchmark data book*. www.eembc.org: EEMBC, 2015.
- [3] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation Test in Europe Conference Exhibition*, April 2009, pp. 502–506.
- [4] G. S. Rodrigues and F. L. Kastensmidt, "Evaluating the behavior of successive approximation algorithms under soft errors," in *2017 18th IEEE Latin American Test Symposium (LATS)*, 2017, pp. 1–6.
- [5] A. Vallerio, A. Savino, A. Chatzidimitriou, M. Kaliorakis, M. Kooli, M. Riera, M. Anglada, G. Di Natale, A. Bosio, R. Canal, A. Gonzalez, D. Gizopoulos, R. Mariani, and S. Di Carlo, "SyRA: Early System Reliability Analysis for Cross-Layer Soft Errors Resilience in Memory Arrays of Microprocessor Systems," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 765–783, May 2019.
- [6] N. J. Wang, A. Mahesri, and S. J. Patel, "Examining ACE analysis reliability estimates using fault-injection," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07. New York, NY, USA: ACM, 2007, pp. 460–469.