C-026

Real-Time Linux による定周期処理の精度向上
Improving the accuracy of periodic processing with the Real-Time Linux

内田 修平† 玉田 竜一†
Shuhei Uchida Ryuichi Tamada

1. はじめに

マイコンの高性能化に伴い組込機器は高機能化が求められ、ソフトウェアは大規模の一途をたどっている。そこで、開発効率の向上を狙って豊富なミドルウェアを備える Linux の採用が増えているが、RTOS と違い Linux は制約の時間内に処理を完了するように設計されていない。このため、スレッドの切り替えが発生する定周期処理の周期にはバラつきが生じ、高速な周期でデータを収集するうえで問題となる。

本稿では、この問題解決に対し、リアルタイム性能を向上した Real-Time Linux の非適用時と適用時に対し定周期処理の間隔と本来の周期との差異を測定することにより、その有効性を評価したので報告する。

2. 富士電機における取組

当社では、工場の製造装置の予防保全などを目的に、ネットワークを介して製造装置からデータを収集する組込製品を提供している。この組込製品の開発に Linux の適用を進めている。これは、Linux がサポートする豊富な通信プロトコルスタック等のミドルウェアを利用することで、製品開発を効率よく進めることを目指している。

一方、収集したデータは、製造装置の故障予知などの今後の変化を予測するためのデータ解析に使用される。ここでは、同一時刻で生じる複数の測定ポイントのデータを収集し、それを時系列で分析することで、変化を予測する。このため、データの測定周期の精度が解析の精度に影響する。

3. 取組にあたっての課題

Linux でも定周期処理を実行可能であるが、RTOS と違い Linux は制約の時間内に処理を完了するようには設計されていない。このため、スレッドの切替が発生する定周期処理の周期にバラつきが生じる。特に高速な周期でデータを収集する場合には、このバラつきが問題となる。

本稿では、周期のバラつきを小さくすることを課題としている。（目標：100ms 周期を±0.5%以内で実行）

4. 解決策

The Linux Foundation が提供する Real-Time Linux[1]は、優先度の高い処理を早く実行可能にするため、割込みハンドラで実行する処理を可能な限り Linux のスレッドで実行すること等して、割込みを禁止する時間帯を可能な限り短くしている。これを踏まえ、Real-Time Linux を使用し、定周期処理の優先度を高くすることで、定周期処理の実行が他の処理により遅延しにくくなり、周期のバラつきが小さくなることが期待できる。

Real-Time Linux は Linux カーネルのパッチとして提供されており、利用している Linux カーネルにパッチファイルを適用のうえで、カーネルコンフィギュレーションを適切に設定することにより、Real-Time Linux を利用可能である。なお、Linux のバージョン 6.12 以降では、パッチファイルを適用することなく、カーネルコンフィギュレーションの設定のみで利用可能であり、比較的に適用しやすい環境が整っていると言える。また、Real-Time Linux の詳細は、The Linux Foundation がドキュメント[2]を提供している。

5. 評価内容と結果

5.1 評価対象

Real-Time Linux が、定周期処理の周期のバラつきを抑えることが可能かを評価する。

また、製品への適用を踏まえ、既存のソフトウェア処理を想定した負荷を与えることとした。具体的には他の処理およびメモリアクセスが実行中の CPU 上で定周期処理を実行した状態で、定周期処理の周期のバラつきを測定した。

5.2 評価方法

Arm 社の Quad Cortex-A53、6GB の LPDDR4 等を搭載した SoC（System on Chip）の評価ボード上で、表 1 に示す環境下で、定周期処理の実行間隔を以下 2 種類の条件に対し測定した。

条件 1：Real-Time Linux 非適用
条件 2：Real-Time Linux 適用

表 1 条件 1、2 共通の測定環境

Linux カーネルのバージョン	6.6
負荷	対象：CPU コア、RAM、I/O 負荷によるロードアベレージ：3.0 方法：stress コマンドの実行
定周期処理の周期	100ms
定周期処理の実行方法	Linux の timerfd を使用

ロードアベレージ： 実行可能または I/O アクセス待ちのスレッド数の平均

timerfd： 定周期処理の実行に利用可能な Linux の仕組みであり、設定された周期に基づき計算された時刻に read のブロッキングが解除されるファイルディスクリプタ[3]

測定手順を以下に示す。

(1) 定周期処理の先頭で GPIO の信号をトグル（low であれば high に、high であれば low に変更）
(2) GPIO の信号変化時刻をオシロスコープで測定し時刻の差を計算することで、定周期処理の実行間隔を測定

† 富士電機株式会社 Fuji Electric Co., Ltd.

5.3 評価結果

前節で示した 2 条件それぞれに対し 200 回測定した際の定周期処理の実行間隔の最大値・平均値・標準偏差・最小値を表 2 に、分布を図 1 に、それぞれ示す。表 2 より、条件 1（Real-Time Linux 非適用）と比べ、条件 2（Real-Time Linux 適用）は、最小値と最大値の双方が本来の周期 100ms に近づいた。このことは、図 1 において、条件 1 では 97～99ms または 101～103ms となることがあったものの、条件 2 では 99～101ms に全データがあることに対応する。これらのことから、周期のバラつきが小さくなり、100ms 周期を±0.5%以内で実行可能であることを確認した。

表 2 100ms 定周期処理の周期

項目	条件 1（非適用）	条件 2（適用）
最大値	101.28 ms	100.37 ms
平均値	100.00 ms	100.00 ms
標準偏差	0.17 ms	0.12 ms
最小値	98.72 ms	99.62 ms

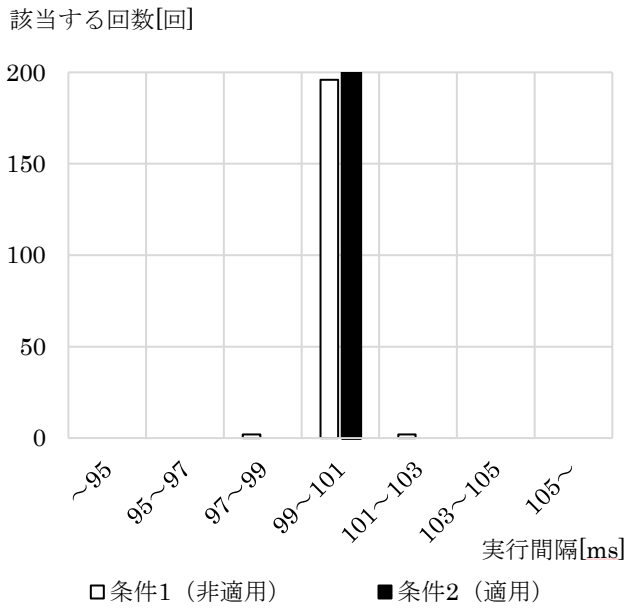

図 1 100ms 定周期処理の周期の分布

6. 考察

条件 2（Real-Time Linux 適用）では、処理の切替方法が変更になっており、既存のソフトウェア動作に対して影響を及ぼす可能性がある。そこで、5.2 節と同様に GPIO の信号変化時刻をオシロスコープで測定し、時刻の差を計算することで、スレッドの切替時間を確認した。

表 3 スレッド切替時間

項目	条件 1（非適用）	条件 2（適用）
最大値	0.057 ms	0.092 ms
平均値	0.054 ms	0.078 ms
標準偏差	0.001 ms	0.004 ms
最小値	0.052 ms	0.073 ms

スレッドの切替時間を 20 回測定したところ、測定結果は表 3 に示す通りとなり、スレッドの切替時間が平均 0.054ms から 0.078ms に、0.024ms 増加することが分かった。

しかしながら、今回想定したデータ取得を行う定周期処理の間隔（100ms）に対して十分小さく、Linux OS の処理ジッタに対して十分に小さいことから、既存ソフトウェアの動作への影響はほぼないと判断した。

7. おわりに

今回、製品への適用を踏まえ CPU および RAM に対する負荷を与えた状態でも Real-Time Linux が定周期処理の周期のバラつきを小さくすることに有効であることが確認できた。また、考察では、Real-Time Linux を適用することにより、スレッドの切替時間が増加することを明らかにした。

今後は、Real-Time Linux を当社製品に適用時、製品としての機能を満足することを確認する。Real-Time Linux の適用により生じるスレッドの切替時間の増加が既存ソフトウェアの動作に影響しないと判断したが、より詳細な製品試験を通して、製品として問題無いことを確認する。

参考文献

[1] Real-Time Linux, https://wiki.linuxfoundation.org/realtime/start
[2] Real-Time Linux のドキュメント, https://wiki.linuxfoundation.org/realtime/documentation/start
[3] Linux の timerfd_create のマニュアル, https://manpages.org/timerfd_create/2