仮想化技術による実機レステスト環境の構築

Software Testing Environment Using Virtualization Technology to Eliminate Need for Actual Equipment

原嶋　秀次　■HARASHIMA Shuji
蔭山　佳輝　■KAGEYAMA Yoshiteru
河込　和宏　■KAWAGOME Kazuhiro

仮想化技術の進展により，パソコン（PC）上で様々なCPUやデバイスのエミュレーションを効率的に実行することが可能になっている。OSS（オープンソースソフトウェア）であるQEMU（Quick Emulator）は，ARM™(注1)，SH-4，PowerPC(注2)などのCPUの動作を別のCPU上でエミュレーションするモジュールを提供している。CPUだけでなく，タイマやメモリなどの周辺デバイスも，エミュレーションのための枠組みが与えられており，短期間でPC上での組込みボードエミュレータを構築することが可能になっている。

東芝は，QEMUをベースとした組込みボードエミュレータを使って実機レステスト環境の構築を行っている。組込みボードエミュレータを利用することで，実際のボードの完成を待たずにソフトウェアのテストができる。また，デバイスの故障模擬機能などを組込みボードエミュレータに実装することで，実ボードでは困難なハードウェア障害時の動作テストを容易に行うことが可能になり，短期間で高品質の組込みソフトウェアを開発できる。

With the progress of virtualization technology, it has become possible to emulate many types of central processing units (CPUs) and devices efficiently on a PC. QEMU is an open source software (OSS) providing modules that can emulate ARM™, SH-4, PowerPC®, and other CPUs on a different CPU. In addition to emulating CPUs, QEMU provides an implementation framework for device emulators such as timers, memories, and other devices used on embedded boards. A whole control board emulator can therefore be developed in a short period of time.

Toshiba is developing a software testing environment that eliminates the need for actual equipment, using an embedded board emulator based on QEMU. The embedded board emulator allows software to be tested even before completion of the hardware on which the software is being implemented. The fault tolerance of software can also be tested by implementing hardware failure modes on a device emulator. Such tests are difficult to execute on an actual embedded board. Using the emulator, high-quality embedded software can be developed in a shorter period.

1　まえがき

東芝は，電力監視・制御システムなど多くの監視・制御システムを開発している[1]。これらのシステムでは，それぞれに求められる機能や性能，信頼性に対する要求から，専用の組込みボードを用いることが多い。組込みソフトウェアは，この専用ハードウェア上で機能や性能，信頼性を満たすよう実装されている。組込みソフトウェアの開発規模は増大しており，数百人体制での開発も珍しくない状況になっている。一方，顧客の事情に合わせてシステムを短期間で開発し提供することも重要である。

実機レステスト環境は，組込みソフトウェアをPC上で動作させて実機なしでテストできるようにするものである。これを利用することで，ボードの完成を待たずにソフトウェアのテストができる，海外を含めた広域な分散開発拠点間で常に最新のテスト環境を容易に共有できる，ハードウェア故障時のテストが容易にできる，などのメリットがある。

実機レステスト環境を構築するにはボードのエミュレータを用いるが，エミュレータを短期間でCPUやデバイスの進歩に追随させるのは困難である。近年ではQEMU（Quick Emulator）[2]に代表されるOSS（オープンソースソフトウェア）の異種CPUエミュレータが公開されており，最新ハードウェアに対応するエミュレータを容易に入手できるようになった。

当社は，QEMUをベースとした実機レステスト環境の構築を行っている。ここでは，組込みソフトウェア開発の概要と課題，及び実機レステスト環境の適用による課題解決について述べる。

2　組込みシステムにおけるソフトウェア開発

当社は，電力監視・制御システムなど社会インフラとして生活基盤を支えたり，企業の生産システムを支えたりするシステムを多く開発している。これらのシステムに求められる機能と性能の実現はもちろんのこと，高い信頼性の確保が重要である。また，同じ種類の制御システムでも顧客によって入出力方

（注1）　ARMは，英国ARM Limitedの商標。
（注2）　PowerPCは，IBM Corporationの商標。

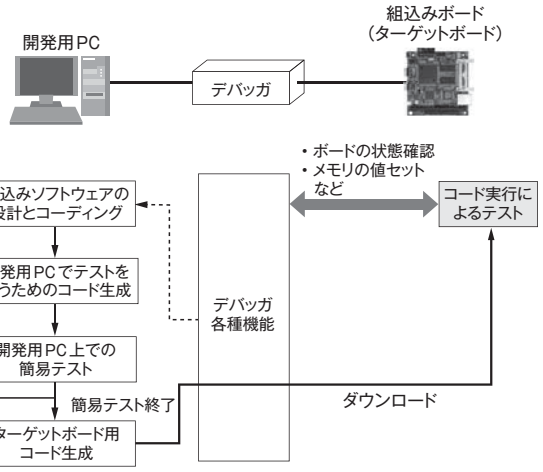

図1．組込みシステムにおけるデバッグ方法 ― デバッグは組込みボード上を中心に行われるため，ボード完成前や他の開発者が利用している間はテストの実施を待つ必要がある。

Outline of software module debugging in embedded system

法や制御対象となる機器が異なるなど様々な対応が必要である。そこで，各システムに適した専用の組込みボードを開発してそれぞれのニーズに対応している。

組込みボードは，CPUと周辺デバイスで構成される。周辺デバイスは，タイマや，メモリ，ネットワークなどであり，監視・制御対象によってタイマの周波数や入出力デバイスの種類，数などを決めている。組込みソフトウェアは，組込みボードに依存したソフトウェアで，ボード上のデバイスを使って周期的に制御対象機器の状態をチェックし，対応する制御を行う構造になっている[3]。このため，一般のPC上でできるのはプログラムの実装から実行コードの生成（ビルド）までである。コード実行によるテストは，**図1**に示すように，実ボードとJTAG（Joint Test Action Group）(注3)などのテスト仕様に準拠した組込みシステム用のデバッガを用いて行うのが一般的である。

このような開発の流れのなかで，次のような問題がある。

第1に，実際の組込みボードがないとソフトウェアの開発ができないことである。前述のとおり組込みソフトウェアはハードウェアに依存した構造になっているため，実ボードが完成するまでソフトウェアを動作させることができない。このことは，開発期間の短縮を困難にしている。

第2に，分散開発への対応が難しいことである。組込みソフトウェアの規模は海外を含めた多くの拠点で行っている。これらの拠点でソフトウェアのテストを行うために，最新の組込みボードを配布する必要がある。開発段階ではボードの仕様変更があるため，仕様変更のつど各拠点へ最新ボードを配布しているが，数日単位での遅れは避けられない。

第3に，ハードウェア故障時のテストが困難なことである。前述のとおり実ボードとJTAGなどのデバッガを用いてテストを行うが，テスト項目に従ってボードを故障させることは事実上不可能である。したがって，組込みソフトウェアのボード故障時の処理が正しく機能するかどうかをチェックすることが困難になっている。

このような状況のなか，組込みソフトウェアの開発者はそれぞれの開発とテストをPC上で実施できる簡易なテスト環境を作っている[4]。担当モジュールに関連する他のモジュールやデバイスのおおまかな動きをPC上で模擬するプログラムを作成し，これを用いて担当モジュールのテストを行う方法である。しかし，この場合のテスト用の実行コードはデバイスドライバを模擬プログラム用に変えるなどしており，組込みボード上での動きと異なるケースが多い。このため，概略テストまでをこの環境で行い，詳細テストは実際の組込みボードが利用可能なタイミングまで待って行うことになる。

当社は，ここで述べた三つの課題を解決する，組込みボードエミュレータを用いた実機レステスト環境を構築している。以下では，実機レステスト環境とそれによる組込みソフトウェア開発の概要を述べる。

3　異種CPUエミュレータ QEMU

3.1　概要

QEMUは異種CPUエミュレータであり，CPUクロックレベルでCPUやデバイスの状態をエミュレートするエミュレータに比べて高速なエミュレーション速度の実現を目指したエミュレータである。x86，ARM，PowerPCなど様々なCPUの動作を，x86やARMなどのCPU上でエミュレートすることが可能である。

QEMUは，Fabrice Bellard氏がUSENIX2005に投稿した論文“QEMU, A Fast and Portable Dynamic Translator”[2]で初めて紹介された。マシンエミュレーションにJIT（Just In Time）の手法を利用し，中間コードを介して実行環境のコードを生成することで，エミュレーションターゲットと動作環境の両方について豊富なプラットフォームへの対応を実現している。現在OSSとして，開発者のコミュニティが形成され，機能追加や性能改善と，ターゲットや動作環境のプラットフォームの拡充が行われている。2012年4月時点で約40名の開発者が参加している。

3.2　動作の仕組み

QEMUは，エミュレーション対象のCPUのコードから中間コードを生成して動的コンパイルを行い，実行環境のCPUのコードを生成し実行する。コード生成は，ジャンプや条件分岐までの短い実行コード列（基本ブロック）の単位で行われる。

（注3）　IEEE 1149.1（電気電子技術者協会規格 1149.1）として標準化された，基板の検査などのために内部状態を読み出す方式。

動作概要は，次のとおりである。

(1) エミュレーション対象の実行コードは，基本ブロックごとに実行時点で，中間コードを介して実行環境のコードへ変換して実行される。中間コードを介することで多様なエミュレーション対象や実行環境への対応が容易になっている。変換結果はキャッシュされ，次に実行される際には高速に実行される。

(2) ステップ実行などを行うための命令単位で変換を行う，特別な処理が実装されている。

(3) エミュレーション対象ハードウェアの内部状態（プログラムカウンタやレジスタなど）の再現を行っている。

(4) コードの変換及び実行とデバイスのエミュレーションを適切なタイミングで切り替えながら，交互に実行する。

QEMUはこのような仕組みで動作するため，実ボードでの動作と大きく違わない速度で動かすことが可能である。また，プログラムカウンタやレジスタの内容の確認をしたり，ステップ単位での実行を行ったり，デバッグがしやすくなっている。

QEMUではシステムエミュレーションモードとユーザーエミュレーションモードを利用することができる。システムエミュレーションモードは，CPUエミュレータとデバイスエミュレータを組み合わせて仮想マシン（ボード）を構成し，その上でOS（オペレーティングシステム）を動かすことが可能なモードである。ユーザーエミュレーションモードは，デバイスへのアクセスを含むシステムコール処理を実行環境のOSのシステムコールに変換して実行するものである。組込みボードエミュレータには，システムエミュレーションモードを利用している。

4 異種CPUエミュレータを用いた組込みソフトウェアの開発

4.1 組込みボード向け実機レステスト環境

3章で述べたように，QEMUによりCPUエミュレータとデバイスエミュレータを組み合わせて，組込みボードのエミュレータを構成することができる。

当社は，組込みシステムで使われているボードエミュレータをこの方法で実装し，実機レステスト環境を構築している（**図2**）。QEMUはシステムエミュレーションモードを用い，CPUについては提供されているエミュレータを利用し，デバイスについては組込みボードで用いている各種のデバイスのエミュレータを開発している。

デバイスエミュレータは，デバイスの動作の仕方により二つの種類が存在する。タイマのような実機レステスト環境を動かすPC上のデバイスを利用するケースと，デバイスの動作を全てエミュレータとして実装するケースである。前者のケースではホストデバイスからの割込みをイベントとしてQEMU側に通知し，QEMUではエミュレーションサイクルでイベントのチェックを行い，イベントが発生していれば対応するデバイスのエミュレーションコードを実行する（**図3**）。後者のケースでは，組込みソフトウェアからのデバイス呼出しが行われた際にエミュレーションコードを実行する。

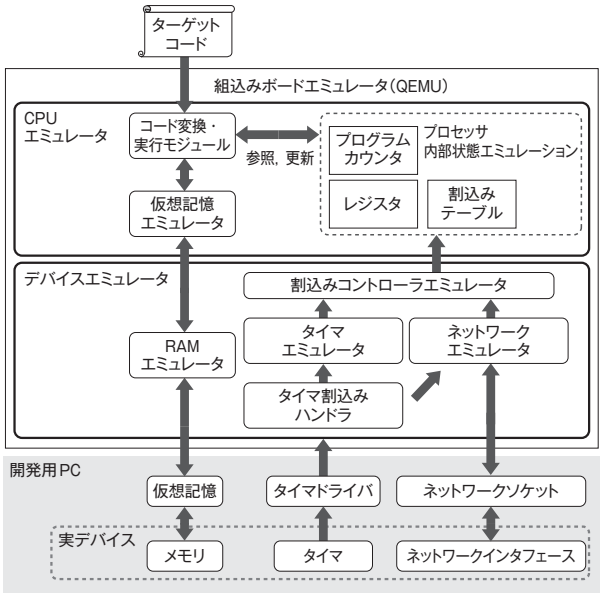

図2．組込みボードエミュレータの構成 ― 組込みボードエミュレータは，CPUとボードを構成するデバイスのエミュレータで構成する。CPUエミュレータはレジスタやメモリなど，内部状態を再現しながらエミュレートする。

Configuration and operation of embedded board emulator

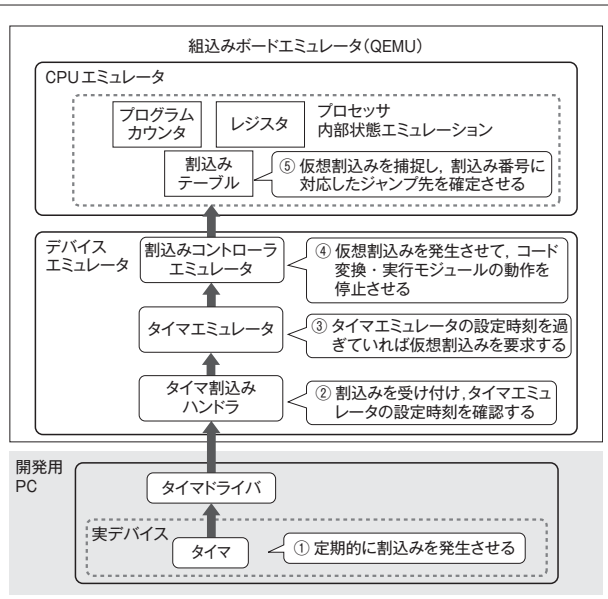

図3．組込みボードエミュレータの動作概要 ― エミュレータを動かしている開発用PCで，LANからのメッセージ受信などをトリガとして，QEMUは対応するデバイスエミュレータを起動する。この結果，仮想マシン上での割込みが生成され，割込み処理が実行される。

Operation of embedded board emulator

エミュレーションは，組込みソフトウェアの実行コードのエミュレーションとデバイスのエミュレーションを適宜切り替えて順番に実行している。

このような方法により，組込みボードエミュレータは組込みボードの状態を再現しつつ実行コードを実行するため，コード実行中のハードウェアの状態を読み出してチェックしたり，レジスタやデバイスの状態をあらかじめセットして実行コードのふるまいをチェックしたりすることが可能である。

4.2　実機レステスト環境を用いた組込みソフトウェア開発

前節で述べた組込みボードエミュレータを用いることで，組込みソフトウェアの開発をPC上で実機なしで行うことが可能になる。これにより，2章で述べた三つの問題点を解決できる。

第1に，実際の組込みボードが完成する前にPC上のエミュレータを使って組込みソフトウェアのテストをすることが可能になる。これは，開発期間を短縮できることを意味する。実機レステスト環境を用いた組込みソフトウェアの開発を**図4**に示す。

第2に，大規模開発体制への対応が容易になる。実機レステスト環境はソフトウェアであり，容易に最新の環境，つまり最新の仕様を反映した組込みボードエミュレータを，海外を含めた開発拠点に送付して共有することが可能である。

第3に，ボードの故障の模擬ができる。デバイスエミュレータに故障模擬機能を組込むことで，実ボードではテストが困難なハードウェア故障時のソフトウェアの動作テストを行うことが可能になる。

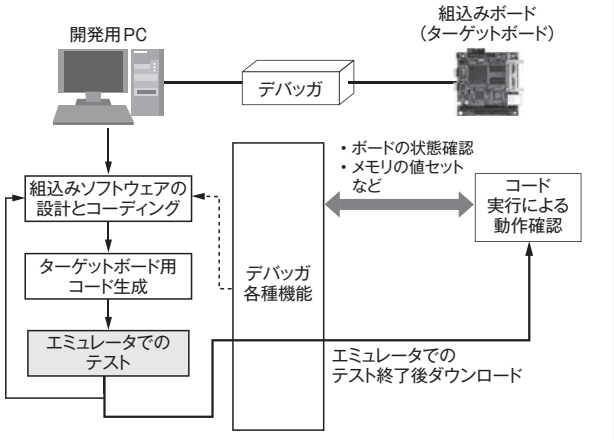

図4．組込みボードエミュレータを用いたデバッグ— デバッグはエミュレータを中心に実施するので，実際の組込みボードが完成するのを待つことなくデバッグができる。

Outline of software module debugging using embedded board emulator

5　あとがき

組込みソフトウェアが大規模になり，機能は複雑化している。信頼性の高いソフトウェアを開発するには，効率的な開発環境の利用が不可欠である。異種CPUエミュレータによる実機レステスト環境はこのような課題に対する解決策の一つである。OSSの異種CPUエミュレータであるQEMUを用いることにより，開発スケジュールに合わせて組込みボードエミュレータを開発することが可能になる。

今後，機器全体のエミュレーション方法を開発するなど，システムレベルでの実機レステスト環境の構築を行っていく。

文　献

(1) 日下部宏之 他．“計測・制御システムの動向と展開”．東芝レビュー．**66**，10，2011，p.2－6.

(2) Bellard, F. "QEMU, a Fast and Portable Dynamic Tanslator", USENIX 2005 Annual Technical Conference. Anaheim, CA, USA, 2005-04, USENIX. ＜http://static.usenix.org/event/usenix05/tech/freenix/full_papers/bellard/bellard_html/＞, (accessed 2012-06-22).

(3) Williams, R.“リアルタイム組込みシステム基礎講座”．宇野俊夫 他訳．翔泳社，2006，11，464p.

(4) 河井　淳 他．“組み込みシステム用ソフトウェア開発環境”．情報処理学会誌．**37**，9，1996，p.872－879.

原嶋　秀次　HARASHIMA Shuji, Dr.Eng.

ソフトウェア技術センター　先端ソフトウェア開発担当参事，工博。システム構築技術の研究・開発に従事。情報処理学会，電子情報通信学会，日本データベース学会，ACM会員。

Corporate Software Engineering Center

蔭山　佳輝　KAGEYAMA Yoshiteru

ソフトウェア技術センター　先端ソフトウェア開発担当主務。システム構築技術の研究・開発に従事。

Corporate Software Engineering Center

河込　和宏　KAWAGOME Kazuhiro

ソフトウェア技術センター　先端ソフトウェア開発担当。システム構築技術の研究・開発に従事。情報処理学会，電子情報通信学会会員。

Corporate Software Engineering Center