ソフトウェア工学 122－13
（1999．3．19）

要求仕様に基づくソフトウェアソースコードの自動生成法

山中 弘† 田村 直樹†
†三菱電機株式会社 情報技術総合研究所

家電製品などに組込まれるマイコン組込みソフトウェアの開発では，ソフトウェアの品質改善と開発期間短縮が最も重要な問題とされている．これら問題を解決するためのアプローチの 1 つとして，要求仕様に基づいたソースコード自動生成法がある．マイコン組込みソフトウェアのソースコード自動生成法としては，従来から状態遷移表あるいは状態遷移図を基にしたソースコード自動生成への取組みがさかんに行われてきた．しかし，マイコン組込みソフトウェアによく見られる時間制約付き周期処理による入出力装置制御ロジックを実装するソースコード生成法は存在しなかった．我々は状態遷移表に時間制約付き周期処理を記述できるよう拡張を行い，そのロジックをタイマ割り込みにより実装する方式について検討を行った．本報告でその内容を説明する．

A mechanism for source code generation based on software's requirements specifications

Hiroshi Yamanaka† Naoki Tamura†
†Information Technology R & D Center, Mitsubishi Electric Corporation

Software quality improvement and reduction of time-to-market are most siginificant issues in development of embedded softwares for consumer products and so on. Source code generation based on the requirements specifications is one of the approaches for this issue. There have been proposed many tools that generates source code based on state transition tables, however, these tools are unable to generate time-constraint source code, which is widely used in micro-computer software. In this paper, we present another approach to generate source code based on state-transition table. We applied extended state-transition table to describe time-constrained requirements, and automatically generate source code based on timer-interupt mechanisms of micro-processors.

1. はじめに

家電製品など，価格競争や新製品開発競争の激しい分野を市場とするマイコン組込みソフトウェアの開発では，短納期，高品質，かつ低コストが求められる．これらの特徴を持つマイコン組込みソフトウェア開発の問題点として，仕様変更の頻繁に伴い開発工程が圧迫される傾向があること，試験仕様の作成や手入力/目視確認といった労働集約的な試験作業に充分な時間がかけられていないことなどが挙げられる．これ

ら問題点を解決する手段として，従来からいくつかのアプローチが提案されている．

我々は，そのアプローチの 1 つであるテスト支援によるマイコン組み込みソフトウェア開発支援を進めてきた[1], [2]．そして今回，支援範囲の拡大を狙って，ソースコード自動生成による開発支援を進めている．

家電製品などでよく見られる周期的に行われる入出力装置制御ロジックは，ハードウェア実機上に組込んでチューニングを行う必要がある．我々が検討を進めているソースコード自動生成法は，このチューニングに伴う修正/変更作業を誤りなく/迅速に行えるようにすることを狙いとし，マイコン組込みソフトウェアの状態遷移表(または状態遷移図)からイベントふるまいロジックを自動生成するだけでなく，周期的に行われる入出力装置制御ロジックを自動生成することを目標とした．

本報告において，まず 2 章で従来から我々が進めてきたマイコン組込みソフトウェア開発効率化のアプローチとソースコード自動生成に取組んだ背景について述べる．3 章で従来のソースコード自動生成法について説明しその問題点を考察する．4 章で我々の提案する新しいソースコード自動生成法について説明する．そして 5 章で提案したソースコード自動生成法の課題点について考察し，6 章でまとめを述べる．

2. 課題と対策

2.1. マイコン組込みソフトウェア開発の課題

マイコン組込みソフトウェア開発では，以下に述べることが課題としてあげられる．まず，マンマシン仕様に対するユーザ嗜好の変化や激しい製品の機能競争によって仕様変更が頻繁に発生する．この結果，仕様変更に伴う修正作業により開発工程が圧迫される傾向がある．また，ソフトウェアの開発期間が短く，その結果試験仕様の作成や手入力/目視確認といった労働集約的な試験作業に充分な時間がかけられていないのが現状である．これらの要因により，製品出荷後の不具合が報告されるケースが増えている．

これら問題点を解決する手段として，従来から次の3つのアプローチが提案されている(図1)．

①プロトタイピング

要求仕様を早期に検証し，開発の後工程での変更を極小化することを狙って，実際にソフトウェアの設計/製造に着手する前に GUI プロトタイプを用いたシミュレーションを行う．

②ソースコード自動生成

要求仕様/設計仕様を基にソースコードを自動生成することで，製造にかかるコストを低減すると共に，開発者の要求仕様の解釈間違いを防ぎ仕様とコードの等価性を保証する．

③テスト支援

作成されたソフトウェアが，要求仕様通りに動作するかを検証するシステム試験工程を効率化する．ユーザの手入力/結果確認作業をなくすためのブラックボックス試験法(ハードウェアに組込んだ試験)，プロトタイピングの流れを受けたシミュレーション試験法(ソフトウェアのみの論理試験)などの方法がある[1]．

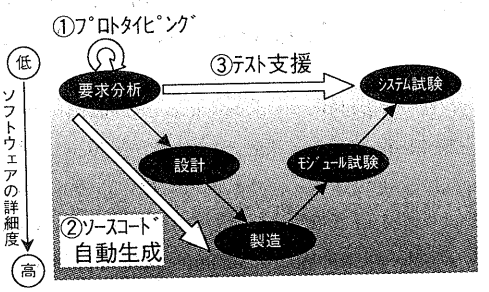

図 1. マイコン組込みソフトウェアの開発支援

いずれのアプローチでも要求仕様の記述が必要となるが，要求仕様の記述法としては状態遷移表または状態遷移図が用いられることが多い．我々の調査でも，マイコン組込みソフトウェアの要求仕様の 50%以上が，表 1 に示すようにイベントとそれに呼応するシステムのふるまいで占められること，状態遷移表または状態遷移図がマイコン組込みソフトウェア開発現場で広く受

け入れられていることが確認されていることからである[1].

表1. 各製品ソフトウェア仕様種別と割合

	PHS	TV	VTR	ファンヒータ	炊飯器	コードレス電話	PPC	FAX	合計(割合)
イベント・アクション	6	7	7	3	4	7	5	4	43.0(53.8%)
安全性	1	1	0.5	3	3	1	1	2	12.5(15.6%)
タイミング	2	1	1	2	1	1	1	3	12.0(15.0%)
快適性	1	1	0.5	1	1	1	3	1	9.5(11.9%)
フィードバック	0	0	1	0	1	0	0	0	2.0(2.5%)
性能	0	0	0	1	0	0	0	0	1.0(1.3%)

2.2. これまで我々の取組み

我々は，先述のアプローチの一つであるテスト支援によるマイコン組込みソフトウェア開発効率化を進めている．そして，従来のマイコン入出力ポートレベルのブラックボックステスト法を改良した新しい開発支援方式とその支援ツール環境 testCASE の開発を行っている[1], [2], [3], [4], [5]．testCASE は，状態遷移表によって記述された要求仕様を基にテスト仕様の作成作業/テスト実行作業を自動化する(図2)．

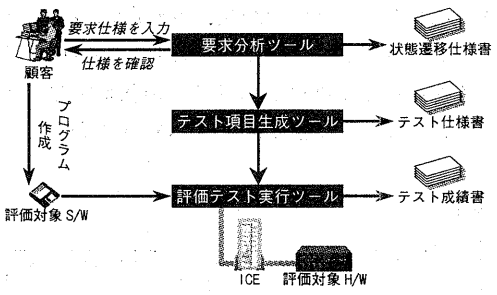

図2. testCASE を用いた開発の流れ

testCASE は，状態遷移表を基にテスト仕様の作成作業とテスト実行作業を自動化するが，ここで試験自動化の仕組みを簡単に説明する．一般的なマイコン組込みプログラムの実装を図3に示す．図3において，ユーザが入力デバイスに与えた操作は，ドライバを介しイベント判別を行うための変数に設定される．マイコン組込みソフトウェアはこの変数の値に従って応答する処理を行う．処理結果は出力値を格納する変数に設定され，ドライバを介し出力デバイスに出力される．testCASE は，この入出力デバイス制御用の変数をシステム試験の界面とし，ブラックボックス試験を行う．イベント判別変数に値を設定することによって自動的に擬似イベントを発生させ，処理結果の出力値格納変数を期待値と照合することにより自動的に擬似出力確認を行う．

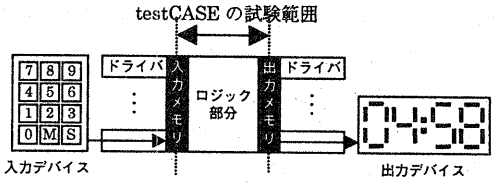

図3. マイコン組込みソフトウェアの典型的実装

この testCASE によるテスト支援法を，従来の入出力ポートを界面としたブラックボックス試験法/シミュレーション試験法と比較した場合，次の特徴がある．

入出力ポートブラックボックス試験法との比較

・イベントの入力値/照合する期待値の設定が楽．
・ICE 以外の特殊な試験装置を必要としない．
・入出力デバイスを含めた試験は実施できない．

シミュレーション試験法との比較

・ハードウェア実機上で試験を行える．

2.3. ソースコード自動生成への取組み

ソースコード自動生成法は，製造工程を効率化すること/要求仕様と生成コードの等価性を保証すること，などの特徴を持つ．仕様変更を即ソースコードに結びつけるソースコード自動生成機能があれば，マイコン組込みソフトウェアの分野で頻発しがちな要求仕様の変更にも対処し易い．我々は，testCASE の支援範囲の拡大を狙って，ソースコード自動生成による開発支援に取り組んだ．後述するように我々の提案するソースコード自動生成法は，マイコン組込みソフトウェア開発で多く行われているチューニング作業の効率化/testCASE の自動試験対象外モジュールの信頼性向上，という特長を持つ．

3. 従来のソースコード自動生成法

従来から，状態遷移表(図)を基にソースコード自動生成を行うツールの発表/市販がなされている[6]．ここでは，従来ツールのソースコード自動生成法を紹介し，その問題点について考察する．

3.1. 従来のソースコード自動生成ロジック

従来の状態遷移表(図)からソースコード自動生成を行うツールは，基本的に以下に説明するロジックでソースコード生成を行う．

①アクションの性質分類

状態遷移表(図)内に，次の 4 つの種類のアクションを記述する(図 4)．

(a) Event アクション

システムがあるイベントを受け取った時に，瞬間的に実行されるアクション．

(b) Entry アクション

システムがある状態に遷移した時に，瞬間的に実行されるアクション．

(c) During アクション

システムがある状態に留まる間，継続的に実行されるアクション．

(d) Exit アクション

システムがある状態から抜ける時に，瞬間的に実行されるアクション．

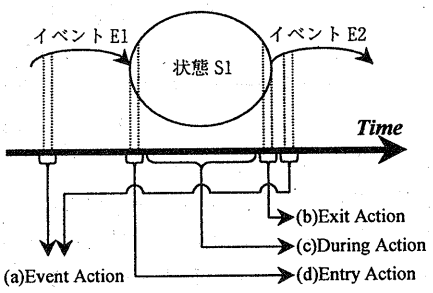

図 4. 状態遷移図のアクション

②アクションディスパッチ関数の生成

状態遷移表(図)の記述を基に，(a)状態の判定，(b)イベントの判定，(c)状態遷移に伴う状態変数の書き換え，を実行し判定された状態/イベントに従って，(d)必要な Action 呼び出しを実行する，アクションディスパッチ関数を自動生成する(図 5)．なお，呼び出される各アクション関数，並びにイベントの判定ロジックの実装はユーザに委ねられる．

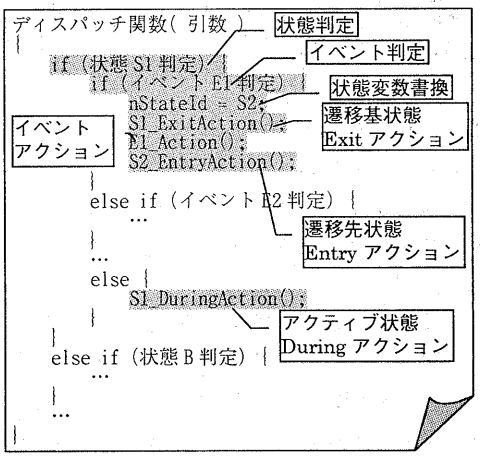

図 5. 状態遷移図からの生成コード例

③生成したディスパッチ関数の呼び出し方

生成されたディスパッチ関数は，基本的にごく短い周期で継続的にメイン関数から呼び出されることを想定し実装している(図 6)．

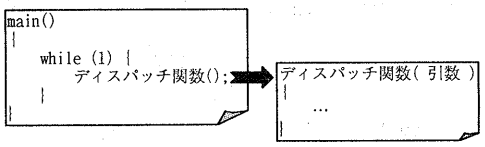

図 6. ディスパッチ関数呼び出し例

3.2. 従来のソースコード自動生成法の課題点

マイコン組込みソフトウェア開発を対象とした場合，以上のソースコード生成法には次の課題点が考えられる．

①時間制約ロジック生成

タイミングチャート，イベントトレース図の時間制約，状態遷移図のタイムアウト，などマイコン組込みシステムの時間制約の記述をサポートするツールはいくつか存在する．し

かし，その記述をソースコード生成にまで結びつけたものはない．これは，時間計測のロジックがマイコンチップやリアルタイム OS に依存し異なることが原因と考えられる．

②入出力装置制御処理のロジック生成

①と関連があるが，図 3 の入出力ドライバロジックを自動生成するものはない．つまり，入力装置から信号としてあがった入力操作をディスパッチ関数内でイベント判定可能なレベルにまでおとし込むロジックや，処理結果として格納されている変数値を出力装置に表示するロジックは自動生成できていない．これはこれらロジックが入出力ハードウェアやマイコンチップ依存であることが根本的要因であるが，他に入出力装置制御仕様に時間制約が含まれることが要因として考えられる(例：物理的に「数字 1 キー」が 1 度押され 0.02 秒にも押されていたときに初めて「数字 1 キー」が発生したとみなす(チャタリング防止)，0.02 秒毎に LED の各桁をダイナミック点灯する，など)．

③During アクションの実行ロジック

状態の During アクションの意味付けは，何通りか考えられる(表 2)．しかし，従来のソースコード自動生成法による実装では，表中の(a)または(b)のものしかサポートできない．

表 2. 状態の During アクション

	処理の構造	処理終了
(a)	ごく短いアクションを何度も繰り返し実行	外部イベント発生により状態を遷移
(b)	ごく短いアクションを何度も繰り返し実行	アクション終了と共に状態を遷移
(c)	時間のかかるアクションを実行	外部イベント発生により状態を遷移
(d)	時間のかかるアクションを実行	アクション終了と共に状態を遷移

4. testCASE のアプローチ

4.1. 基本的アプローチ

我々は，状態遷移表を基にイベント/アクションのふるまいを実装したプログラムを自動生成することに加えて，先述の従来方法の課題点②を解決することを狙って，時間制約付き周期処理の自動生成法を検討している．時間制約付き周期処理とは，決められた時間スケジュールで実行される周期処理をいう．また，この時間制約付き周期処理内で実行される個々の処理は，実行時間を無視可能なものと仮定する．

主に家電製品向けのマイコン組込みソフトウェアでは，入力キーマトリクスのキースキャン/LED 表示パネルのダイナミック点灯/通信処理などの処理を行うために，ある決められた時間間隔による周期的な処理を実装する．我々の目標は，時間制約付き周期処理に対するソースコードを自動生成することにより，周期的な処理を行うマイコン組込みソフトウェアの入出力ドライバの一部を自動生成することにある．

この自動生成法を実現することにより，次の効果を期待できる．

①入出力ドライバチューニング作業の効率化

マイコン組込みソフトウェアの開発では，目測やプロトタイピングなどによって定めていた周期の時間間隔を実機上でチューニングする作業が行われる．この自動生成法により，マイコン組込みソフトウェア開発で多く行われているチューニング作業を効率化できる．

②試験対象外モジュールの信頼性向上

図 3 に示したように，testCASE によるマイコン組込みソフトウェアの自動試験では，入出力装置の制御モジュールまで試験を行っていなかった．しかし，時間制約付き周期処理のソースコード自動生成によって，入出力装置制御モジュールの基本構造が実装され，testCASE でテストできていなかったモジュールの信頼性を向上することが可能となる(図7)．

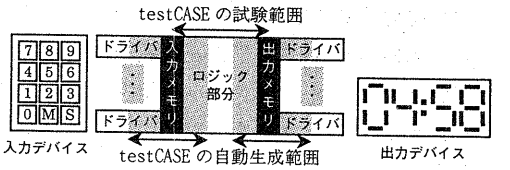

図 7. testCASE の支援目標

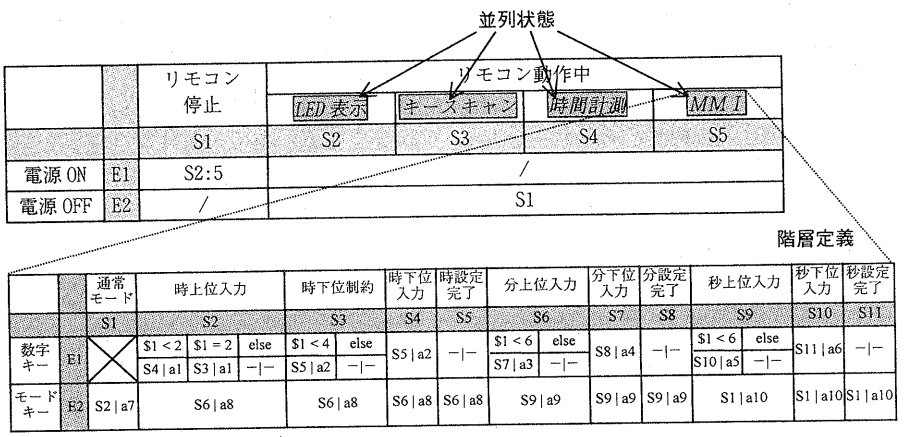

図 8. 拡張された状態遷移表の記述

このような時間制約付き周期処理のソースコード自動生成を実現するため，我々はソフトウェアで実行される各時間制約付き周期処理を記述する仕様記述法と，各時間制約付き周期処理をグループ分けして実行する割り込みスケルトン関数自動生成機能を検討した．

4.2. 時間制約付き周期処理の仕様記述法

我々は，従来の状態遷移表記述法を拡張し，次のように時間制約付き周期処理を含めた要求仕様の記述を可能とした．

①階層記述/並列状態記述のサポート

状態遷移表の階層定義/および並列状態の記述を可能とする．図 8 中，状態名がイタリック文字で網掛け表示されているものが並列状態を表す．また図 8 の"MMI"状態のように，状態に対してその下位階層を表す状態遷移表を定義可能である．

②周期処理

時間制約を持つ周期処理を，各タイミングで実行される単位アクションと単位アクション間の時間間隔をもって記述する．図 9 の"LED 点灯周期処理"内の"LED1 上位桁点灯"，"LED1 下位桁点灯"，…が単位アクションを表す．また，単位アクション間の"25msec"が時間間隔を表す．

③周期処理の状態遷移表への対応付け

定義した周期処理を，状態遷移表の During アクションとして関連付ける．例えば，図 9 の LED 点灯周期処理は図 8 の LED 表示状態に，時刻カウント周期処理は時刻計測状態に，などの決定を行う．

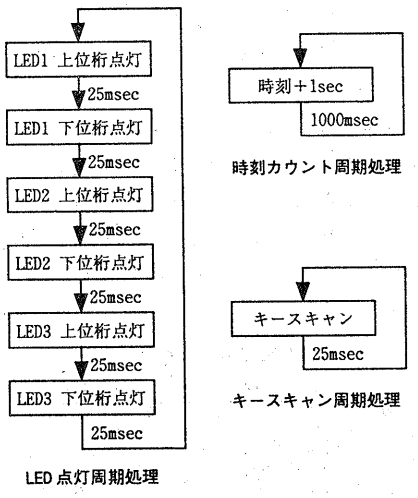

図 9. 時間制約付き周期処理の記述

4.3. 時間制約付き周期処理の自動生成

以上に述べた時間制約付き周期処理仕様を基に，その実装ソースコードの自動生成を行う．実装方法にはいくつかのやり方が考えられる(表 3)．今回の報告では，これら候補の中からハードウェア/OS 依存性が少なく正確な時間の計測が可能なタイマ割り込みを選択している．

表 3. 時間制約付き周期処理の実装候補

	長所	短所
タイマ割り込み	・正確な時間間隔	・処理のオーバヘッド
タイマカウンタ	・処理のオーバヘッドがない	・正確な時間間隔を測れない ・状態遷移ロジックとの処理の一本化が必要
リアルタイムOS	・正確な時間間隔 ・実装が楽	・OS依存 ・処理のオーバヘッド

図 10 に自動生成される時間制約付き周期処理を実装したタイマ割り込み関数の例を示し，以降その生成ロジックについて説明する．

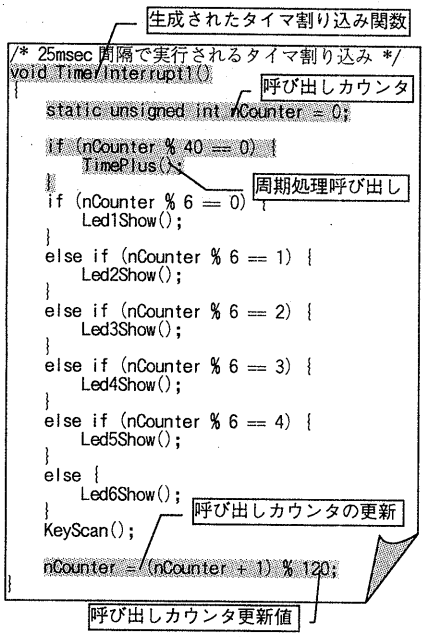

図 10. 時間制約付き周期処理の自動生成例

①同期周期処理を収集

一つのタイマに同期して動作が可能な周期処理を収集する．状態遷移表の定義から同じ時点でアクティブになると判断される状態について，その During アクションを収集する．

②共通周期の導出

複数の周期処理を一つのタイマにスケジュールするための共通周期を導出する．共通周期は，すべての単位アクション間時間間隔の最大公約数である．最大公約数があまりにも小さい場合(例えば最小の時間間隔未満)，ユーザに対して警告を発する．

③呼び出しカウンタ更新値の導出

生成される割り込み関数毎に，割り込み関数の呼び出し回数を記憶するための変数を生成する．これを呼び出しカウンタと呼ぶこととする．呼び出しカウンタがオーバフローしないように，一定回数毎(=呼び出しカウンタ更新値)呼び出しカウンタを初期化する必要がある．この回数は，各周期処理の周期時間の最小公倍数を共通周期時間で割ったものである．この値があまりにも大きい場合，ユーザに対して警告を発する．

④個々の単位アクションの呼び出し

周期処理において実行される各単位アクションを実行するタイミングは，呼び出しカウンタの剰余を求めることにより知ることができる．

5. 議論

4 章で時間制約付き周期処理の自動生成法について説明したが，この方法ではいくつかの条件/仮定をつけてコード生成を実現している．ここでは，本報告で設けた条件/仮定に伴う問題点，および本方法の拡張の可能性について述べる．

①タイマカウンタによる実装

本報告で述べた自動生成法では，タイマ割り込みによって時間制約ロジックを実装したが，これをタイマカウンタを用いて実装するやり方も考えられる．

⇒それぞれのアーキテクチャの向き/不向きのカテゴライズが必要．

②タイマリソースの制約

本報告では，タイマリソースが無限に存在することを仮定し，特にその制約について考慮していない．しかし，マイコンチップによって，利用可能なタイマ数が制限ざれる可能性がある．自動生成したソースコードが必要とするタイマ数が，利用可能なタイマ数を超えた場合にどう処理するか．

⇒いくつかの周期処理を統合し，リソースの範囲内に抑える．

③性能要求仕様の導出/フィードバック

本報告では，周期処理の時間制約に対する実行可能性を論じていない．

⇒時間制約付き周期処理仕様から，その時間間隔の間で実行されるべきアクションの性能要求が導かれるが，このアクション性能要求の導出機能/またはアクションの性能を基に周期処理の時間間隔を自動計算することができないか．

④入出力装置制御モジュールの完全自動生成

本報告の自動生成法により，時間制約付き周期処理仕様から入出力装置制御モジュールの基本構造が自動生成可能となった．しかし，ハードウェアに直接アクセスを行う処理は依然ユーザに製造してもらう必要がある．

⇒ハードウェアに依存する処理をハードウェア設計仕様に基づいて自動生成することができないか．

6. まとめ

マイコン組込みソフトウェア開発支援ツール環境 testCASE の一機能として考案した，状態遷移表とその拡張仕様記述に基づいたソフトウェアソースコードの生成法について報告した．

本報告では，多くの家電製品に見られる時間制約付き周期処理の実装に焦点をおいた．時間制約付き周期処理実装を自動的に行うために，その仕様記述法を状態遷移表記法に拡張追加し，従来のソースコード自動生成法になかった時間制約ロジックの自動生成を可能とした．この自動生成法を家電製品などのマイコン組込みソフトウェア入出力装置制御モジュールに適用することにより，従来の入出力ドライバのチューニング作業を効率化し，また testCASE の試験実行対象外であったモジュールの信頼性を向上し，マイコン組込みソフトウェアの品質向上/開発効率改善につながることが期待される．

今後は，本自動生成法を実プロジェクトに適用してその効果を確かめるとともに，ハードウェア設計仕様を組入れた入出力デバイスドライバの自動生成に取組んでいきたい．

参考文献

[1] 中島，他，「シングルチップマイコン用 S/W 開発における問題点と一解決法」，情報処理学会研究報告，Vol. 97，No. 74，pp. 17-24，1997.

[2] 山中，他，「ソフトウェア機能試験手順の状態遷移表に基づいた生成法」，情報処理学会研究報告，Vol. 98，No. 20，pp. 119-126，1998.

[3] 萩原，他，"マイコン組込み S/W 開発を支援する CASE ツール：testCASE(1)～全体構成～"，情報処理学会第 57 回全国大会，6J-07，1998.

[4] 山中，他，"マイコン組込み S/W の開発を支援する CASE ツール：testCASE(2)～状態遷移表とテスト手順生成～"，情報処理学会第 57 回全国大会，6J-08，1998.

[5] 別所，他，"マイコン組込み S/W 開発を支援する CASE ツール：testCASE(3)～testCASE を用いた開発の流れ～"，情報処理学会第 57 回全国大会，6J-09，1998.

[6] "ZIPC ユーザのための ZIPC ガイド 入門編"，ZIPC サポートセンター，1998.