ZIPC を用いたコード生成手法とその評価考察

三菱電機コントロールソフトウェア株式会社
三田事業所　技術第１課

森下　修

１．はじめに

近年カーオーディオ製品の S/W は、製品の急激な高機能化に伴ってプログラムの「大規模複雑化」が進んできた。このような状況のもと、より一層充実したソフトウェアの設計ドキュメントの必要性が高まる一方で、新技術も次々と登場し、製品の開発期間は短縮の一途をたどってきている。そのためどうしてもプログラム開発が先行してしまい、設計ドキュメントの立ち遅れが目立つようになってきた。ソフトウェアに対する変更修正は、設計ドキュメント上で広い範囲に渡って正当性や影響範囲を検証したうえで、その内容に基づきプログラムを修正するのが本来の形である。しかしプログラムに対して直接的に変更修正個所を検討すると、局所的な対処になりやすく、ソフトウェア仕様の矛盾や欠落など、品質劣化へ直結する問題が発生する危険性を含んでいる。

そういった問題を防止するための取り組みとして、「カーオーディオ S/W」の中で、メカ機構が複雑な為に多くの状態や事象をもつ「インダッシュ型 CD チェンジャーメカ」（図１）のソフトウェア開発において、状態遷移表設計手法を導入することにした。ソフトウェア仕様を状態遷移表レベルで視覚的に表現する事で、「漏れ抜け」や「矛盾点」を浮き彫りにし、状態遷移表自体をソフトウェアの設計ドキュメントと位置付け、状態遷移表からＣプログラムを自動生成する事で、ソフトウェアの設計ドキュメントのバージョンとプログラムのバージョンが絶えず等価関係であることを目指し、「ZIPC」の導入に踏み切った。

図１　インダッシュ型ＣＤ チェンジャーメカ概観図

2．ソフトウェア構成

インダッシュ型 CD チェンジャーメカ（以下メカと略す）の制御ソフトウェアの構成について簡単に記述する（図２）。ソフトウェアの機能は「再生制御」「通信制御」「メカ機構制御」「スイッチ・センサ取込制御」の 4 つの制御機能に大別される。再生制御機能はCD-DSPLSI ※1 を介して、CD の再生制御を行う。通信制御機能はカーオーディオ本体からのコマンド受信及び応答ステータス送信、連続的に受信したコマンドの実行順序の解析を行う。メカ機構制御機能はモータードライバを介してモータ制御を行いメカ機構を駆動する。またメカを駆動することによって変化する各種スイッチやセンサの状態を、スイッチ・センサ監視制御機能で取り込んでいる。

※1　CD から読み出したデジタル信号を復調してオーディオ信号として出力するための LSI

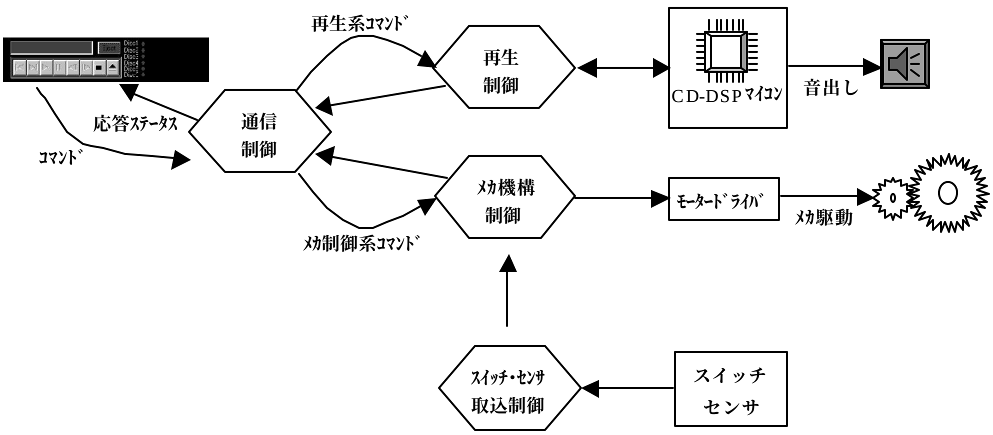

図２　ソフトウェア概略構成図

3．従来の設計手法の問題点

制御対象であるメカの機能が複雑になればなるほど、メカがとり得るすべての「状態」と、すべての「事象」を洗い出す作業が大変になる。正常動作時の状態と事象の組み合わせは取りこぼすことなく抽出することが可能だが、通常の制御シーケンスでは考えられにくいような異常ケースは、設計段階で見つけ出すことが難しい。

従来は、メカ機構設計のエンジニアによって動作の概略が検討され、決定した動作仕様をもとにソフトウェアエンジニアが、ソフトウェアでの実現方法と制御シーケンスを検討していた。正常な制御シーケンスの検討は、従来の設計方法でも大きな問題が発生することは無かった。しかし異常ケースについては、制御対象の複雑化にともない設計段階では想定出来なかったものが、システム検証段階で発生することもあった。そうした場合は、その都度設計フェーズまでフィードバックして、異常ケースに対応したソフトウェアの設計を再検討する必要があった。

しかし製品化間近になってくると時間的制約などで設計フェーズまでフィードバックする余裕が無くなり、プログラムを直接手直しして対策を講じるという局面が多々ある。そういった事の積み重ねが、プログラム構造の複雑化や、設計ドキュメントとプログラムのバージョンの不一致を引き起こす原因になっていた。

4．ZIPC 導入の実際

こういった問題点を効率良く解決するためには、ソフトウェアの設計段階で、正常ケース・異常ケースすべてを網羅した制御シーケンスを決定しておくことが必要不可欠である。また不具合や仕様変更などによって生じた設計の手直しは、必ず設計ドキュメント上で行うという開発フローを取り決める必要があった。そこで、種々手法検討の結果 ZIPC を用いた状態遷移表設計を採用することにした。

従来も設計の初期段階では、基本動作を示すドキュメントとして一般的な表計算ソフトなどのツールを使用して状態遷移表を作成していたが、プログラム製作を開始した後は、あまり参照される設計資料ではなかった。しかし ZIPC がもつ C コード生成機能を活用することで、設計資料として作成した状態遷移表がそのままプログラムコードへと直結することになり、プログラム製作にかけていた時間のほとんどを、設計時間に割り当てることが可能となった。ZIPC 導入後の開発プロセス（ZIPC 適用プロセス）は、**図 3** のようになる。

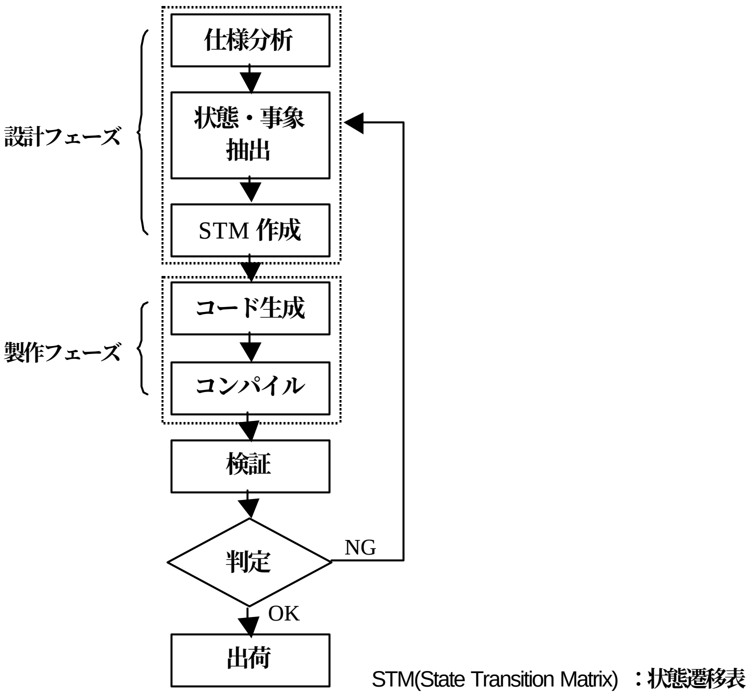

図3　ZIPC 導入後のソフトウェア開発プロセス

5．評価と考察

5．1　工数評価

ZIPC 適用プロセスを評価するために、従来の開発プロセスで開発したものと同じ機種を、ZIPC 適用プロセスに沿ってリメイクし、工数に対する評価を行った。その結果が図４である。開発フェーズは主にプロトタイプレベルである「設計試作」と、量産に向けた「量産試作」の２つのフェーズに分けられる。

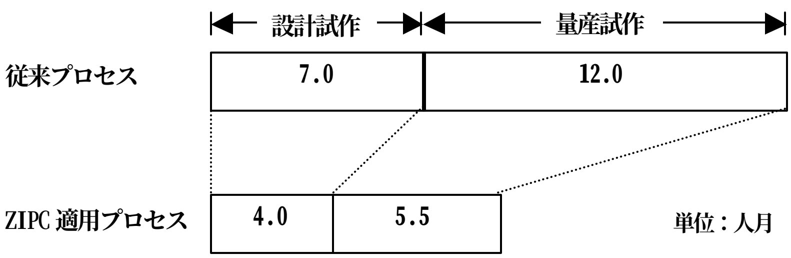

図４　工数比較

比較した結果､設計試作において 3 人月、量産試作において 6.5 人月の工数削減が行えており、それぞれにおいて 50%前後の工数を削減出来た。特に量産試作フェーズでは、制御対象であるメカの動作仕様や機構の変更などに伴い、ソフトウェアの制御シーケンスの追加や変更が発生し､従来プロセスでは仕様変更に対して、ソフトウェアでの対処方法を、主に C ソースコード上で検討することが多く、ソフト変更による影響範囲や検証範囲の見極めは、プログラマー本人の経験による推測の域を脱していなかった。そのため目に見えないところで重大な影響を与え、新たな不具合を埋め込んでしまうおそれがあったため、たとえ小さな変更であっても確認のためにソフトウェアの全機能を検証しなおす必要があった。しかしZIPC 適用プロセスでは、状態遷移表上でソフトウェアの変更に対処するため、変更された個所はすべて「セル」で視覚的に表現されることになり、理論的にソフトウェアの変更および変更箇所の検証を行うことが出来た。これらのことからZIPC を用いた状態遷移表設計は、仕様変更に対して非常に有効であると判断している。

5．2　ROM サイズ評価

次に ROM/RAM サイズの評価を行った。制御対象となるメカには ROM 容量が 64kbyte､RAM 容量が 4kbyte の CPU を実装している。**図 5** は従来プロセスと ZIPC 適用プロセスのプログラムの ROM/RAM サイズを比較した結果である。

		従来プロセス	ZIPC 適用プロセス
ROM	サイズ	51.4kbyte	87.7kbyte
	使用率	80%	137%
RAM	サイズ	2.63kbyte	2.91kbyte
	使用率	65%	72.8%

図 5　ROM/RAM サイズ比較結果

まずROMについてZIPC適用プロセスは使用率 137%と大きくオーバーし、従来プロセスのものと比較して約 1.7 倍という数字になった。RAMに関しては、従来プロセスのものと比較して約 1.1 倍であり、C コード生成時に自動生成されるSTM専用のRAMに関しては46byteであった。RAM サイズに関してはまったく問題無い結果であるが、ROM サイズに関してはCPUのROMサイズをオーバーしており、問題が残る結果となった。

ROM サイズが CPU 容量を大きくオーバーした理由としていくつか考えられる原因がある。まず従来プロセスで設計したソフトウェアは、異常状態の処理という視点では、「漏れ・抜け」が 100%ないとは言いきれない。それに対し ZIPC適用プロセスで設計したソフトウェアは、仕様の「漏れ・抜け」に関して 100%に近いレベルまで実装されていると言える。また、今回はじめてZIPCを用いて状態遷移設計を行ったということでツールや手法に対して不慣れな部分もあり、冗長設計があらゆる個所に散見されたため、STM を最適化することによって、スリム化が図れるという分析結果を得た。

そういった分析結果を考慮すると最終的に、従来プロセスと比較して 1.4 倍程度にまでサイズを抑えることが出来るという結論が得られた。ZIPC 開発元であるキャッツ株式会社によると機能等価の場合は STM から自動生成したプログラムの ROM サイズは理論上 1.2 倍になるとの事であるが、そこまで最適化をはかるにはかなりの分析時間を要することが予想され、今回の評価によると 1.4 倍という数字が現実的であると判断している。すなわち現在CPUの ROM容量の使用率が 70%程度の機種であれば、ZIPC を適用する事が可能と考えられる。

5．3　ソースコード評価

次に状態遷移表から自動生成されるソースコードについて評価を行った。対象としたプログラムの全ライン数は約42000行である。その内訳は図**6**のようになっている。全体の60%がベクタテーブルや遷移先情報など、状態遷移表から自動生成されたコードである。残りの40%は状態遷移表のセルに記述した処理に対して、設計者がC言語に近い形で記述した「置換情報」とよばれる変換データを用いてZIPCが変換したコードで、内訳を示すと 35%が共通関数的なアクション関数部分、残りの 5%がドライバ部分のコードである。

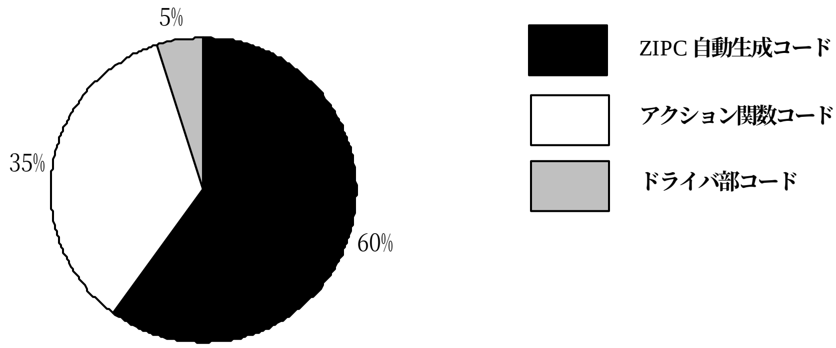

図６　ソースコードの割合

ZIPC の C コード生成機能によって自動生成された 60%部分には、生成ミスは発見されず、安定した品質のコードが出力されていた。また処理系に全く依存しない形で出力されているため､他の CPU へ移植する場合も、制御対象のメカ機構と制御シーケンスが等価である限り修正を加える必要は無用と考えられる。更に 35%にあたるアクション関数部分も、処理系に非依存の形で設計することが可能である。今回の仕様では、このプログラムを他の処理系へ移植する場合、全体の 5%にあたるドライバ部分のみ修正を加えればいいという事になる。これらの事実から、ZIPC 適用プロセスでは、非常に移植性の高いプログラムが作成可能だと判断している。

6．まとめ

6．1　ZIPC 総合評価

以上のような観点から ZIPC を用いた状態遷移表設計を評価してきた結果、ZIPC はカーオーディオソフトウェアの開発に十分適用可能であるとの結論に至った。特に C コード生成機能は非常に有効な機能である。今回は約 42000 行のソースコードを、約 1～2 分程度で生成しており、これは十分に許容出来る時間である。従って製品化間近になっても、直接プログラムを手直しして対策を講じるという処置を取らず、必ず設計ドキュメント（状態遷移表）を修正してプログラムを自動生成するという事を徹底することが出来た。その結果当初の一番の目的である、

- **プログラムと設計ドキュメントが絶えず等価関係である**

という目標はクリア出来たと考える。また状態遷移表作成作業に関しても、ツール自体が非常にシンプルな操作性で実現されている。状態やイベント、アクションセルの追加や削除といった操作も非常にシンプルであるため、変更修正作業も簡単に行える。

しかしメリットばかりではなく次のような改善希望、使用にあたっての注意点がある。状態遷移表から C コードを生成する為には置換情報が必要だが、状態遷移表の規模が大きくなるにつれ置換情報も非常に多くなる。しかしすでに入力した置換情報の一覧を参照することが出来ないため、開発が進むにつれ似たような置換情報があちらこちらに散見されるというようなこともある。またデバッグ時に全状態遷移表から一括して置換情報を検索するというような事が出来ないため、状態遷移表上から、修正したい部分を見つけ出すのに時間がかかる。さらに、状態遷移表設計という手法も知識とテクニックを要するために設計者の育成にやや時間がかかる。そのため使用者側での教育も重要である。

6．2　今後の課題

ZIPC には STM シミュレーション機能が用意されており実機が完成していない段階でも密度の濃い検証が行えるようになっている。しかし今回は手元に実機としてメカを用意することが出来たため、STM シミュレーションは実施しなかった。だが開発段階では異常な制御シーケンスの検証として、メカに対して過度のストレスを与えることがある。その結果メカが故障してしまうケースも多々あり、修理を行っている間は一時的に実機レスになることもあった。また人間の手では発生させにくい異常シーケンスの組み合わせもあり、その方法を模索することに多大な時間がかかるというケースもあった。そこで仮に STM シミュレーションを行ったと仮定した場合の工数を、予測してみた結果が図 7 である。

図 7　STM シミュレーションを行った場合の予測工数比較

キャッツ株式会社より情報を収集した結果、我々が作成した状態遷移表の内容であれば、状態遷移表1枚辺りの平均シミュレーション時間は理論的に約2時間であるというデータを出していただき、これをもとに見積もった結果、設計試作検証において約24%、量産試作検証において約 27%もの工数が削減出来るという推測値を算出する事が出来た。以上のような結果から、STM シミュレーションを行うことの意味は非常に大きいと言える。更にZIPCでは3D-CADやエミュレータ-と連動する機能も持っている。それらを総合的に導入することによって、より一層効率良く、密度の濃い検証を行うことが可能であり、今後はこれらの予測結果の実証を行う事が第一の課題となっている。

6.3　今後の方向性

最後に今後の方向性について述べる。今回の評価自体は全体的に非常にいい結果が得られたが、ROM サイズがネックとなり、評価対象となった機種に関してはZIPC適用が非常に難しいという結論になった。しかし、今後の動向としてCPU におけるメモリのコスト費は年々低下傾向にあり、今までは ROM が 64kbyte標準であったものが、128kbyte もしくは 256kbyte 標準になっていきつつある。このことを考慮すると、今回評価対象となった機種と同等機能のものを、今後は 128kbyte や 256kbyte のメモリを実装した CPU で開発していくことも視野に入れることが出来る。そうなると ROM サイズアップが大きな問題になることは考えにくい。

従って今後のカーオーディオソフトウェアの各機種展開に対して、順次 ZIPC の適用を検討・展開していく価値があるという結論に至った。