気軽にZIPC！

ヤマハ株式会社
アドバンストシステム開発センター　VPグループ　主任
穴田　啓樹

1．はじめに

このレポートの目的は、ZIPC を利用した状態遷移表設計手法を経験したことのない方に、この手法のメリットを理解して頂くことです。

ZIPC WATCHERS Vol.5 では業務用音響システムのリモコンの設計・開発にZIPC を利用した例を掲載させて頂きましたが、これは大規模な例でした。今回はもっと身近に、気軽に ZIPC を利用した小規模な例を 2 つ紹介します。

私はこの手法のメリットとして、状態遷移表や自動生成コードといった成果物だけでなく、そこに至るまでのプロセスにも価値があると考えています。特に次の 3 点を中心に話しを進めます。

- 状態遷移表を作成する過程で頭の中が整理されてシステムの理解が深まる
- セルを記入していく方法により作業がはかどる
- 全体が把握できて安心できる

2．ITRON カーネル勉強会での利用例

ここでは ITRON カーネル勉強会において、サービスコール発行によるカーネル内部の状態遷移を状態遷移表に作成し、ZIPC のシミュレーション機能を使って動作を説明した例を紹介します。私は組み込み開発技術研究会 CEST という、豊橋技術科学大学の高田広章先生を中心に、中部地域の企業が集まり、毎月、組み込みシステム開発について幅広くディスカッションしているグループに参加しています。とても優秀な方が集まられて毎回たいへん有意義に勉強させて頂いています。CEST に興味を持たれた方は是非 http://www.ertl.jp/CEST（2002 年 5 月 31 日現在）にアクセスしてみてください。

2001 年の夏には高田研究室と CEST が共同で、高田研究室で開発している ITRON カーネル TOPPERS/JSP http://www.ertl.jp/TOPPERS/（2002 年 5 月 31 日現在）のソースをレビューする勉強会が開催されました。私も参加して「タスクの待ちと起床」を担当し、説明の補助ツールとして ZIPC のシミュレーション機能を利用して発表しました。表 1 がその状態遷移表です。この表は私の解釈・責任で作成したものです。不備がありましても、高田先生や研究室の皆様、及びその著作物である TOPPERS/JSP の関係者に問い合わせをしないよう、お願いします。簡単に説明しますと、サービスコール発行をイベント、ITRON カーネルの内部状態を状態として表を構成

しています。発表当日は ZIPC のシミュレーション機能を使って、各サービスコール（ZIPC のイベント）を発行しながら ITRON カーネルの内部状態が遷移する様子を説明しました。この状態遷移表を作成するにあたり、ITRON 仕様書を確認して TOPPERS/JSP のソースを追跡しました。ITRON 仕様については説明しませんので必要に応じて仕様書を確認ください。まず、状態は DORMANT, RUNNABLE,WAITING,SUSPENDED,WAITING SUSPENDED の５つです。実際には待ち状態に伴う付属状態があるのですがここでは省略しています。「RUNNABLE」を見ておや？と思われた方がいるかもしれません。ITRON 仕様書に定義されている、「READY」「RUNNING」状態は TOPPERS/JSP の内部状態としては「RUNNABLE」として実装されています。実際の動作では「RUNNABLE」状態のタスクの中からスケジューラーが優先順位に従い、どのタスクを「RUNNING」にするかを決めていて、内部状態として「RUNNING」か「READY」かを意識する必要はないからです。

既に ZIPC を使われている方の中には、ZIPC は ITRON のシミュレーション環境を持っているのに何故このようなことをするのか疑問に思われたかもしれません。今回は ITRON カーネル自身の動作を勉強・説明するために状態遷移表を作成したものなので、その点はご理解ください。ここでお伝えしたい重要なことは、状態遷移表を作成していく過程で、頭の中が整理されて理解が深まった点です。例えば、待ち状態にあるタスクを強制終了（ter_tsk）した場合の動作はどうなるでしょうか？「強制終了するだろう」と思われましたか？　実際には強制終了する場合とそうでない場合があり、待ち状態のタスクの起動要求が（act_tsk により）キューイングされている場合には、タスクが再起動され RUNNABLE 状態になります。これは一例に過ぎないのですが、状態遷移表を作成しながらサービスコールとカーネル内部の状態の関係が明確になり、ITRON カーネルの動作理解が深まりました。

この記事を読んで頂いた方の中で、まだ状態遷移表で設計されたことのない方がいらしたら、是非、身近なもの、よく知っている状態マシンについて、状態遷移表を作成してみてください。状態やイベントの抽出、遷移やアクションを決めていく過程でシステムの理解が深まり、きっとこのプロセスの価値に気づかれると思います。

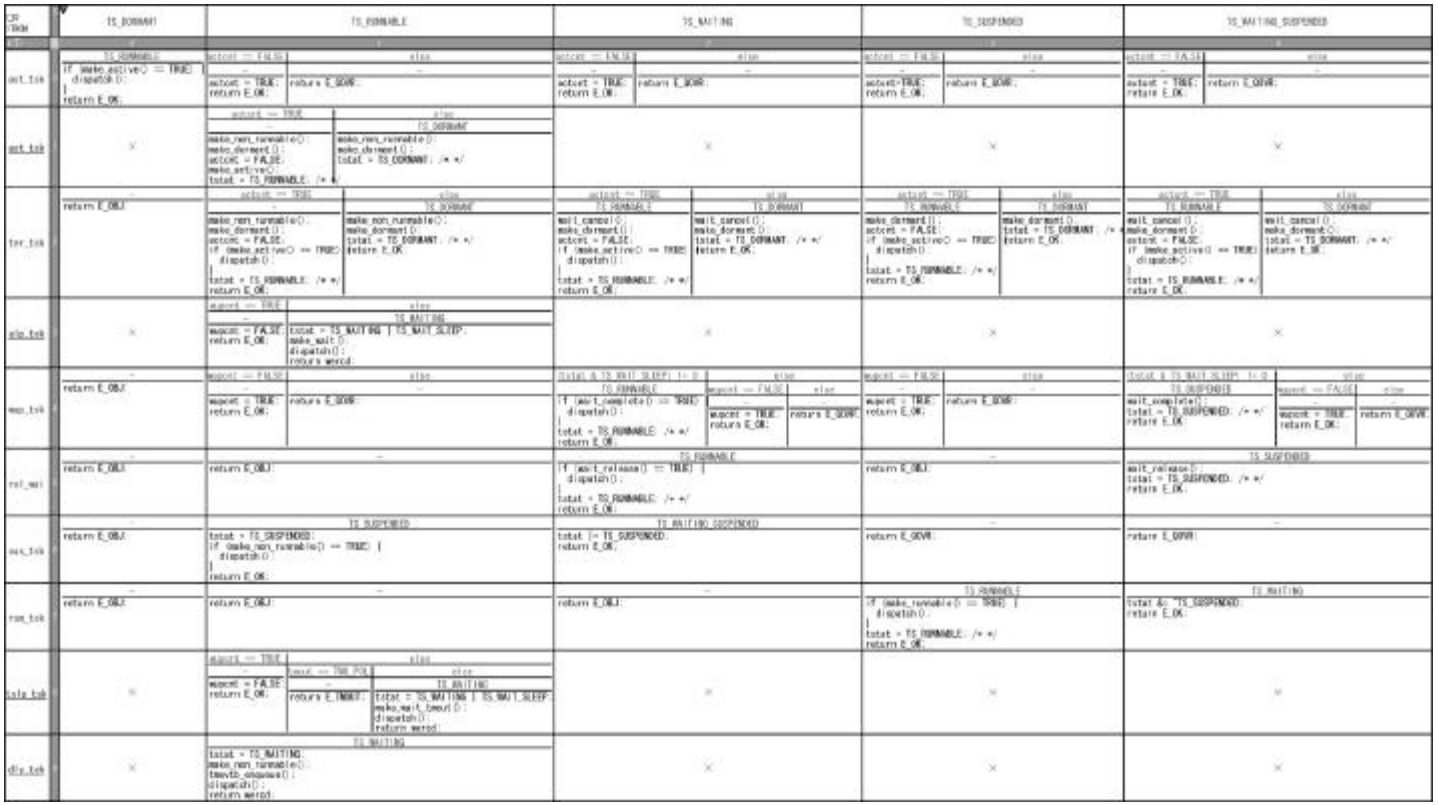

表１　ITRON サービスコール発行によるカーネル内部の状態遷移

3．デモプログラムGUI開発での利用例

ここではMIDIシーケンサーデモプログラムのGUI開発において、マウス操作によるMIDIシーケンサーの演奏状態遷移をZIPCを利用して設計した例を紹介します。このシステムは株式会社エルミックシステム様が開発されたLinux on ITRONのデモンストレーションとして、弊社がMIDIシーケンサーアプリケーションを開発したものです。MST2001のエルミックシステム様のブースで展示予定だったのですが、都合により、展示できませんでした。Linux on ITRONについて簡単に説明しますと、LinuxとITRONのそれぞれの得意な部分を組み合わせて使うHybrid Architecture OSの1つで、ITRONのタスクの1つとしてLinuxが動作します。リアルタイム性が要求される部分をITRONで処理させ、ネットワークやGUIなどをLinuxで処理させます。詳細を知りたい方は、http://www.elmic.co.jp/japanese/products/accel_linux.html（2002年5月31日現在）をアクセスしてみてください。このデモプログラムではリアルタイム性の要求されるMIDIシーケンサーをITRONで処理させ、GUIをLinuxで処理させました。

図1がそのGUI画面です。LATIN, TECHNO, CLUB, JAZZ, ROCKボタンを押して演奏曲を選択するとタマゴの表示が動物の表示に変わります。START, STOPボタンで演奏の開始・停止を制御して、火山の麓にある骨のスライダーで音量を制御します。火山が噴火しているのは音量メーター、右下の数字は演奏ポジション表示になっています。

出展の話が決まったのが開催2週間ほど前でしたので、デモでお客様の操作に耐えうるものを短期間に開発する必要がありました。このデモプログラムの要件は4つあり、5曲選択できること、演奏開始・停止できること、音量制御できること、演奏ポジションを表示できることでした。これ以外の詳細仕様は決まっていませんでした。

表2がこのデモプログラムの状態遷移表です。作業の手順ですが、まず状態を抽出しました。「停止中」「演奏中」「一時停止中」の3つの状態があります。次にイベントの抽出ですが、「ボタンが押された」「スライダーが操作された」「演奏が終了した」の3つがあります。ボタンやスライダーの仕様について詳しくみていきます。メカニカルなボタンでは、押した瞬間に動作するケースが多いと思うのですが、画面上のボタンではボタンが押され、そして、ボタンの領域内でマウスが離されたら動作して、ボタンの領域外でマウスが離されたら動作しないケースが多いと思います。更にはボタンが押された後に領域を外れると表示が変化するものやしないものもあります。スライダーについてはスライダーを押して動かした時に動作するもの、スライダ

ーを離した時に動作するものがあります。普段何気なく使っている GUI の動作も、改めて調べてみるといろいろな動作仕様があることがわかります。

ここまで読まれて「何故 toolkit を使わないのか？」と思われた方も多いと思います。Linux には多くの toolkit がありますし、ITRON にも少ないですがいくつかあります。いくつか調査してみたところ、私の考える組み込みシステムの GUI の要求を満たすものはありませんでした。ビジネスアプリケーション以上に、組み込みシステムでは GUI は製品の顔、会社の顔になる部分なので、各社デザインやユーザビリティを含めて力をいれる部分だと思います。色やボタンのデザインを多少変えられるものはあるのですが、他では使われないような独自の機能をもつ部品を追加可能であったり、動作を微妙に変えられたり、痒いところに手が届くものは、私が調査した限り見つけられませんでした。自社用に専用の toolkit を開発している会社もあると思いますが、今回はそこまではせずに、Microwindows を利用して開発しました。

もし状態遷移表を作らずにいきなりコーディングしていたらどうなったでしょうか？　短期間で作りたいのでソースを早く書きたいところですが、「演奏開始ボタンが押されたら演奏開始する」といった仕様一つをとってみても、ボタンが押されたらまず表示を押されたイメージに更新して、ボタンの領域内でマウスが離されたら演奏を開始する、という詳細仕様が隠れています。こういった細かな仕様をコーディングしていきながら気づいてまたコーディングしていくと「建て増し温泉旅館」が作られてしまうわけです。意味が通じないといけないので説明しますと、ここでは増改築の多い構造物という意味で「建て増し温泉旅館」という言葉を使いました。本物の温泉旅館は風情があってよいですが、プログラムの建て増しは困ります。

状態遷移表設計手法のメリットとして、モレ・ヌケを防ぐ、という謳い文句がありますが、セルを埋めていけばよいのではかどる点もあります。もちろんこれは状態遷移表の作成時に状態とイベントが適切に抽出されていることが前提となりますが。一般に言えることとして、人間は全体が把握できないと不安になります。先の建て増し温泉旅館の例では、全体が見えずに後から付け足していきますが、状態遷移表の設計では、先に全体を把握してからセルを埋めていくので安心感もあります。

4．おわりに

今回は 2 つの小規模な事例を通して、私が ZIPC を使った状態遷移表設計手法のメリットだと考えている点を説明しましたが理解して頂けたでしょうか？ ZIPC は CASE ツールだということを意識せずに気軽な思考ツールとして使えます。私はワープロソフトや表計算ソフト

のような当たり前な存在だと感じているくらいです。ZIPC WATCHERS Vol.5 にも書きましたが、まずは無料 ZIPC セミナーでこの手法を体験されることをお勧めします。

図1　MIDI シーケンサーの GUI

			演奏状態		
			STOPPED	PLAYING	PAUSE
ボタン	SELECT	領域内でSELECTボタンが押された	新しいSELECTボタンを押されたイメージで描画	―	―
		ポインタが領域内で離された	PLAYING 以前のSELECTボタンを離されたイメージで描画 音量スライドを最大値で描画 曲選択コマンドを送信	―	―
		ポインタが領域から外れた	新しいSELECTボタンを離されたイメージで描画	―	―
	STOP	領域内でSTOPボタンが押された	―	―	―
		ポインタが領域内で離された	STOPボタンを押されたイメージを描画	STOPボタンを押されたイメージを描画	STOPボタンを押されたイメージを描画
		ポインタが領域から外れた	―	STOPPED STOPボタンを離されたイメージで描画 演奏停止コマンドを送信 PLAYボタンを離されたイメージで描画 音量スライドを最大値で描画 シーケンスポジションを初期化 ベロシティメーターを初期化	STOPPED STOPボタンを離されたイメージで描画 演奏停止コマンドを送信 PLAYボタンを離されたイメージで描画 音量スライドを最大値で描画 シーケンスポジションを初期化 ベロシティメーターを初期化
			STOPボタンを離されたイメージを描画	STOPボタンを離されたイメージを描画	STOPボタンを離されたイメージを描画
	PLAY	領域内でPLAYボタンが押された	―	―	―
		ポインタが領域内で離された	PLAYボタンを押されたイメージで描画	―	―
		ポインタが領域から外れた	PLAYING 演奏開始コマンドを送信	PAUSE PLAYボタンをポーズのイメージで描画 演奏停止コマンドを送信	PLAYING PLAYボタンを押されたイメージで描画 演奏継続コマンドを送信
			―	―	―
			PLAYボタンを離されたイメージで描画	PLAYボタンを押されたイメージで描画	―
VOLUME		音量スライダーをドラッグされた	―	―	―
		音量スライダーのツマミを離された	音量スライダーを描画	音量スライダーを描画	音量スライダーを描画
			―	―	―
			音量コマンドを送信	音量コマンドを送信	音量コマンドを送信
曲終了を検出した			×	STOPPED PLAYボタンを離されたイメージで描画 シーケンスポジションを初期化 音量スライドを最大値で描画	×

表2 MIDI シーケンサーの状態遷移表