ZIPC 導入の利点

旭光電機株式会社 技術部 開発課　和田 貴志

和田＠旭光電機と申します。電子機器のハード・ソフト開発部隊の管理職を務めています。管理職とは名ばかりで、中小企業によくあるパターンの、実務もクレーム出張もある毎日を過ごしています。

この度、CATS さんから ZIPC に関する原稿依頼を受けたのですが、私どもはまだ導入したばかりのよちよち歩きのヒヨコです。そんなヒヨコですが、導入のいきさつやすぐに効果を得られる利点の説明は、まだ導入されていない方々へのメッセージになるかな？と思い、筆を執りました。

ZIPC は開発規模や使用環境に左右されにくい、汎用的なツールだと思います。今まで使ってきてみた実感として、ソフトハウスだけでなく、むしろ実験やハード設計の傍らでソフト作成するような方々に、最適な開発ツールだと私は思います。

■ 今までの開発方法

今まで私どもがソフト開発してきたスタイルは、最少の資料を基に、頭でフローを考えながらガリガリとアセンブラで書き上げるものでした。80x86 とTLCS-90 というT社製の Z80 ライクなCPU を喜んで使ってきました。通の方のお好みは 68 系らしいですが、フローチャートや定義表を書かずに組んでいくような仕事に、80 系は結構あっていると思います。「えーっと、今 DE には前のエンコーダ値が入ってるから、これを HL に足せば....」と頭を使っている時、80 系の固有名のレジスタはいいですね。「R3 を R5 に持ってきて・・・えーっと R4 てなんだっけ？」。H8 ではとかく混乱しています。アセンブラを移植する際に、移植先の CPU のニーモニックが苦手だから作業がなかなか進まない、ってのも結構ありました。

そんな仕事ぶりですから、客先へ提出するソフト仕様書の作成は、いつもデバッグの見通しがたったぐらいから始まります。RAM マップ等の各種定義表などはよいとして、悩みの種はフローチャート作成です。作成に時間はかかるし、ソフト修正したときに同期を合わせるのも大変です。アセンブラからフローチャートを作成するツ

ールも試しましたが、省略していいような部分まで出力するために編集に手間取り、すぐに使用をあきらめました。

なぜ、フローチャートを先に書かないのでしょうか。書いてみるとすぐにわかるのですが、初期段階でフローを作成すると、客先要求の変更等で、すぐに図や線が予定していた用紙幅に収まらないようになり、あっちへ移動、こっちで削除、となります。本来のシーケンスを考えるよりも、図の編集作業の方がずっと時間を食います（関数は除く）。

こんなことなら、プログラムがまとまった最後でフローを書き上げた方が仕事が早い、ってことになりますよね。ただ、状態遷移図は最初に書いていました。フローチャートと違って遷移図はよほど状態が増えない限り、手書きの書き込みでだいたい最後まで使えるからです。結局、開発中の自分に役立つ状態遷移図は先に作成し、どちらかというと業務の引継に役立つフローチャートは後回しでした。

■ ZIPC との出会い

こんな私どもが ZIPC を導入するきっかけとなったのは、ある客先の方の声でした。

「初品立ち上げの開発までそちらにお願いするとして、その後の細かな元請けの注文に応えられるように、簡単にソフトを修正できるような構成で組めないの？」

ソフト開発はそんな簡単なものではありません、と言いたいところをぐっとこらえ、「C で開発すれば、ソースも見やすいですよ」と提案しても、「いや、そんなんじゃなくてブロックをちょちょいと動かせばなんとかなるとか」と要求されるだけでした。

フローチャートから自動生成されるソフトの存在は知っていましたが、きちんとソフトを考えずにフローの変更を行うと混乱してしまうだけなため、最初から対象にはなりませんでした。
そんな折、ネットワーク OS の開発資料をパラパラっと見ていて、間に挟まった ZIPC の資料を見つけたのです。最初は「えー？CASE？うちには関係ない。」と軽く流しました。当時は CASE の意味すらよく理解出来ていませんでした。

2 週間後ぐらいにもう一度資料を見直しているうちに、ノリのいいこれからの ZIPC 計画を記述した文章（サターンを予言しているあれです）を気に入って読んでいる自分がいました。読めば読むほど STM＋ジェネレータは求

めている内容にピッタリだと思いました。早速 CATS の営業さんに電話してVer.5.0の発売日を確認し、発売までの間に ZIPC 導入をスムーズに運ぶ計画を開始しました。

■ ZIPC 導入の段取り

マイコンに初めて ZIPC を導入した時のステップを、実体験から参考までに記述します。

① 開発ツールの社内説得

すでにマイコンの開発ツールが揃っていればZIPCだけで済みます。無ければ、ZIPC フルセット＋マイコン開発ツール一式の費用が必要です。今の不景気な世の中で費用を会社に捻出してもらうためには、導入することで得られるメリットをきちんと説明する必要があります。幸いなことに ZIPC は導入するメリットがたくさんあります。以下は、私が導入する際にプレゼンした資料の抜粋です。

＜CASE は何のために必要か＞

1）現状の問題点

（1）十分に資料を作成することなくプログラミングを行うため、論理的なバグや打ちミスなどのデバッグに時間が取られ、全体の開発工程が長くなる。

→ 設計速度×、設計コスト×

（2）出荷後に、考えていなかった条件でのバグが判明したり、論理的な矛盾が見つかってROM 交換に至ることが多々ある。

→ 設計品質×

（3）開発終了後にフローチャート等のドキュメントを作成しているが、かなりの時間が必要となる。

（4）仕様書とソフトの中身を常に完全一致させるのは難しい。

→ 設計資料×

（5）ソースしかない場合、開発担当本人以外に理解できる人間はいない。フローチャートでも、全パターンが漏れなく検討できているかどうかの判断は極めて困難。

→ 設計審査×

2）導入による改善・利点内容

（1）ソフト開発工程の短縮化

→ 設計納期○、設計コスト○

（2）出荷製品のソフトバグ・モレの発生を無くす

→ 設計品質○

（3）ドキュメントの自動生成

(4)ドキュメントとソフトの不一致を無くす
→ 設計資料○
(5)第三者でもソフトの詳細検討(DR 等)を可能にする
→ 設計審査○

品質・コスト・納期の三大要素をすべて改善出来る説明は、開発以外の方々にもわかりやすかったようですぐに承認を得ることが出来ました。意外なことに(5)への賛同が結構大きかったです。縁の無かったソフトの世界に、一歩近づけるような気になったのかな？と思いました。

② ZIP、C マイコン開発ツール一式（ICE、C 等）を入手

最初に述べたとおり、私どもはアセンブラだけで仕事をこなしてきました。そんな私どもが、C をスキップして CASE＋C 自動生成の開発スタイルへ2段飛びです。急いでZIPCの発注と同時に、C が使えるＴ社とＭ社のマイコン開発ツールを入手しました。それぞれのマイコンへの移植が本当に自動で出来るのか、不安と期待が入り交じっていました。

③マイコンのハード学習、試験基板作成

このあたりはハードを開発していれば問題ない領域です。余談ですが試験基板は、メーカ品を買わずに設計を自ら行って作成するのが一番です。次に試作基板を作るときにその経験が役立ちます。

④ ターゲット基板に ICE をつなぎ、アセンブラで簡単な作動チェック

まずはやっぱりアセンブラ。ハードとソフトを同時にデバッグするには欠かせません。スタートアドレスからニーモニック単位でブレークできる強みはありがたいものです。

⑤ C で簡単な作動チェック

マイコン用の C 言語が初体験の私どもにとって、ここはちょっと苦労した所です。C と言えば VC++しか触っておらず、マイコンでの C って、スタートはどうするんだろうとか、考えれば考えるほど謎ばかりです。メーカー推奨のサンプルリストをじっくり読むのが早道だと思います。

⑥ ZIPC の STM＋ジェネレータで作動チェック

いよいよです。4 セルぐらいの簡単な状態遷移表を作成し、チェッカをかけて、シミュレーション。ジェネレータも通って、C 言語を生成。そのC言語が・・・通りません。M 社製のコンパイラではエラー続出でした。サポートとメールのやりとりを繰り返した結果、ジェネレータをカスタマイズして頂き、問題なく通るようになりました。今はリビジョンアップしていますので、どなたでも問題なくコンパイル出来ることでしょう。ここでもちょっと苦労しましたが、おかげでジェネレートがどうやって C 言語を生成しているか、また生成された C 言語の構成の理解やアセンブラへの置き換え方などを学習する良い機会を持つことができました（教えていただきました）。

上記のように、CATS さんのサポート体制は実にしっかりしていて、私どものような初心者でも安心して作業を進めることが出来ました。（夜の9時過ぎにメールを打つと、10 時半に返事が返ってきたりして、驚いたこともありました）

■ 分けるとわかる

どんどん導入が進むと、オールアセンブラの仕事にも状態遷移表を活用するようになってきました。当然ジェネレータは使えませんが、それでも使うメリットは十分すぎるほどあります。

交通関係などクリティカルな分野の仕事で、込み入った制御の部分を開発するときに、もやもやした部分を残していると、なんとなくその仕事全体に対してプレッシャーのような、重たい物を感じてしまいます。しかし状態遷移表のすべてのマトリクスにきちんと記入し終えると、プログラムが全然出来ていないにもかかわらず、「もう悩む部分は無い」と感じることが出来ます。この気持ちは体験しないとわかりにくい、すがすがしくて新鮮なものです。設計審査での突っ込みにも明快な回答をすることが出来ます。

一人で組んでいても、表の作成前までがシーケンスを考える部分、表の作成後からコーディングする部分、ときっちり分けることが出来ます。チームで開発すればなおその利点はあることでしょう（まだ ZIPC を使用したチーム開発の経験はありません）。

この感覚を一旦つかむと、状態遷移図でアバウトに流れを決めて、コーディングしながらそのシーケンスを固めていく今までのやり方なんて、能率が悪くてやってられません。自然に消滅しました。

■ 最後に

昔、Y 社のシンセサイザーで、DX7 という名器がありました。FM 変調だけでいい音を出す、しかし制限も多い機種でした。やがて、「こう出来ないか、ああ出来ないか」といった声をほぼすべて取り込んだ SY77(99)という、音を作る要素の接続が自由自在に操れる機種が出ました。でも、結局ユーザーはその無限といっていい音づくりの手法の中から、ごく一部の機能を使って満足していたと思います。

ZIPC の Ver.5.0 の状態遷移表も、まさに自由自在です。状態やイベント記述の部分も何層にも階層化・並列化出来るし、各セルにもやろうと思えば色んな事をとことん記入できます。だけど、色々試していると自分のスタイルが出来上がってくるというか、好んで使用する機能が限定されてきます。恐らく ZIPC を使っている 10 人が同じ内容のシーケンスをそれぞれ状態遷移表で記述しようとすると、10 人皆バラバラのスタイルになると思います(ごく単純なのは除いて)。

だったらフローチャートはもっと自由だ、という声がありそうですが、状態遷移表を書いている時に感じる、品質への安心感と、押着せのない自由な感覚とは、まるで異なるものです。

ツールは、機能と使いごこちが重要です。使うだけで、開発のスタイルを自然に変えてくれた ZIPC ってのは、ちょっとすごいと思います。他にもネットワーク OS の分散処理への応用方法、STM の FTA 的使用法など試しておもしろいことがありました。きっと、使う人ごとに発見があると思います。最後までお読みいただきありがとうございました。

P.S.
CATS 殿へ一つお願いがあります。
STM を縮小印刷可能にして下さい。

(わだ たかし)