FOMA端末の外部制御装置におけるASN.1ツールとZIPCの適用事例

（株）アイ・エス・ビー
モバイルソリューション事業部　第７システム部　技師
吉田　昌平

１．はじめに

当社（株）アイ・エス・ビーは、主に移動体通信関連におけるソフトウェア開発、エンジニアリングサービス等をビジネスの中心としているソフトウェアハウスです。

昨今の移動体通信分野では特に3G～次世代にかけてのソフトウェア開発ビジネスが大半を占め、ターゲットは端末機からインフラ装置、製品検査、各種開発・評価ツール、モバイルソリューションなど広範にわたる業務を行っております。

また、そこで培った経験やノウハウを活かした応用的な開発や、自社で保有するコアの確立、移動体通信以外の事業ドメインとの融合、海外へのリソース展開などにも積極的に取り組んでおります。

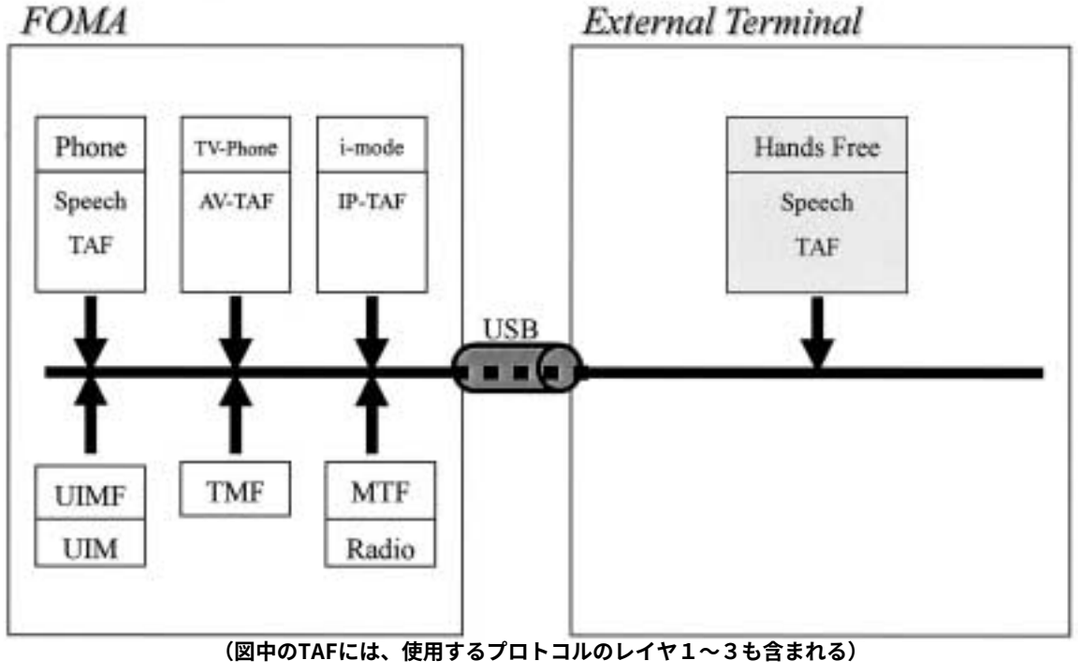
弊社のプロジェクトにおきまして、ZIPCとASN.1ツールを適用したシステムの開発を行いましたので、そのご紹介をさせていただきます。

２．ターゲットシステムの概要

ZIPCとASN.1ツールを適用したシステムは、FOMA端末とマイコンで制御している外部装置とをUSB経由で接続し、ハンズフリーでFOMA端末のベアラサービスを利用するシステムです。

FOMA端末と外部ユニットをUSBで接続し、外部ユニットからハンズフリーで音声の発着信を行うシステムで、通信をATコマンドでなくFOMA端末内部からの論理バスが延長されてきたイメージで通信制御することを目的としています。（図参照）

論理バスの規格は、ARIB TR-T12-27.A02 V3.4.0です。



（図中のTAFには、使用するプロトコルのレイヤ１～３も含まれる）

３．ツールの適用部分および目的

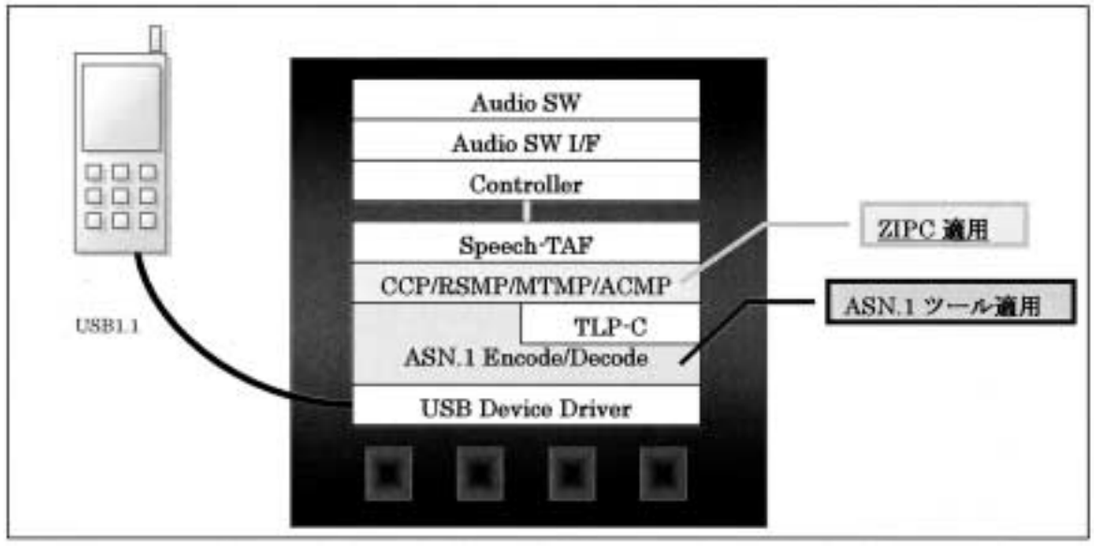
（１）適用部分

ツールを適用したのは、FOMA端末への無線部などを通信制御するための論理バス部分の開発です。

論理バス上はASN.1ツールのBasic-PER（Aligned）でエンコードされたデータを使用し、外部装置はデータをFOMA端末へ送信時にエンコードを行い、FOMA端末からデータを受信した時にデコードを行う仕様です。その送受信機能部にASN.1ツールで生成されたソースコードを組み込みました。

また、FOMA端末の無線部を制御するためのレイヤ３の仕様がZIPCで提供されていたので、その仕様からZIPCで生成されたソースコードを組み込みました。（次図参照）

（２）適用理由・目的



プロジェクトにおいて、ツールを適用した経緯は以下の通りです。

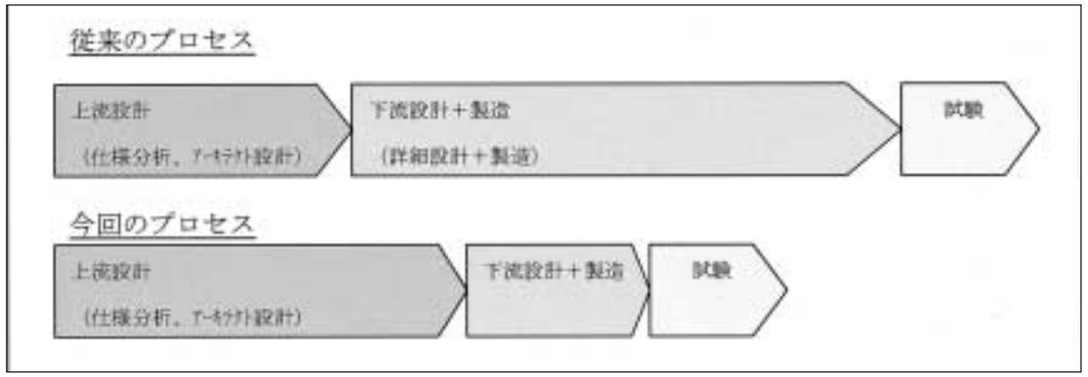
1．元となるメッセージ仕様がASN.1規格で記述されており、また、元となるFOMA端末仕様の状態遷移表がZIPCで作成されている点から、ASN.1ツールとZIPCを適用することは自然な選択でもあった。
2．４つのプロトコルとその制御部を開発するにあたっての納期スケジュールが厳しく、短期間で設計から結合試験まで実施しなければならない状況であったため、ツールを使用して仕様書からコードの生成までのプロセスを効率化する必要があった。

４．開発手順

通常こういった通信層の開発を行う場合、規格に基づいて動作シーケンスや状態遷移表を作成し、イベントや状態ごとの詳細な設計を行ってからソースコードの設計を実施するというプロセスが一般的であり、どちらかというと下流設計に位置付けられる詳細設計と製造工程に時間的コストがかかる傾向があるといえます。

今回は、ZIPC及びASN.1ツールでソースコードを作成する機能を使用することによりその時間的コストを削減し、その分上流設計レベルでのバグ（仕様矛盾など）への対策をしっかり検討するなど、上流工程にウェイトをおいて開発することができました。（次図参照）

５．メリット・効果



（1）メリット

ツールを適用したことによる最大のメリットは、先述したとおり作業・時間コストの軽減が大きいと考えられます。

ASN.1ツールではエンコード／デコードの試験環境がサンプルプログラムを流用することで作成でき、作業期間短縮とソフトウェアの品質向上に大きく寄与しました。

ZIPCでは規格の状態遷移表を利用してソースコード生成できたことが大きかったと考えられます。

また、イベント振り分けやマトリクスコール部も規格の状態遷移表をC言語風に書き換えることでほぼ動作可能なソースコードを生成でき、多くの工数を費やすプロトコル部の設計と製造の作業効率を大きく改善できました。

（2）効果

前述した工程でこれら２つのツールを使用し、この他にエンジニア自身が行う作業としては、

・プロトコル間のイベント配送制御
・タイマ処理の作成
・共通関数の作成

と、比較的容易な作業のみとなり、プロトコルの状態遷移やASN.1のエンコード／デコード部の生成とそれぞれの結合作業を短期間で終了させることができました。

ZIPCでは、状態遷移が作られていることでヌケ・モレがないこと、準正常・異常系に関しては統一した処理で関数化したので、その関数を追記することでほぼ全ルートのコーディングができてしまっていたことなどが大きな作業工数減の要因でした。

同様にASN.1ツールは、ASN.1で記述されたファイルからソースコードを生成する作業と一部重複するシンボルの解決（後述）を実施しただけであり、またツールで生成されるCSVファイルを用いてエンコード／デコードの試験環境が作れたため、短期間でプロトコルすべてのメッセージエンコード／デコードの製造、確認までを行うことができました。

また、ASN.1ツールもZIPCも自動生成した部分のソフトウェアの品質が高く、大きな問題は出ないので、結合試験などにおいて結果的にバグの発生率が下がり試験フェーズにおける生産性を上げられることになりました。

これらの効果を次の表にまとめました。

６．評価・要望など

作業フェーズ	従来:今回	効果
上流設計（仕様分析、ｱｰｷﾃｸﾄ設計）	3 : 4	上流設計がより充実
下流設計（詳細設計）＋製造	6 : 1	大幅な生産性 Up
結合試験	3 : 2	1Kstep のバグ発生率の減少による生産性 Up (50 bug/Kstep->20bug/KStep)
プロジェクト全体　総合評価	5 : 3	

いずれもツールとして大変優秀であり十分に有効性を評価しておりますが、今回の開発プロセスの中で気づいた点や問題点などをいくつか挙げてみたいと思います。

１．ASN.1ツール
・２重定義問題
（問題）
１つのシステムで異なるASNファイル上に同じ名前のラベルが存在すると、生成したコードをリンクしたときにシンボルが2重定義になる
（対策）
生成したコードで重複している部分はプロトコル名を追加して回避した。

２．ZIPC
・生成コードの処理性能
（問題）
ZIPCで生成したコードで、イベント受信からマトリクス起動し処理終了までに時間がかかり負荷が高いときにイベントが処理できない。
（対策）
マトリクス処理と状態遷移処理を１つに統合した。

３．共通の問題
・シンボル名字制限
（問題）
本業務で使用したコンパイラ、リンカーのシンボル名が16文字までの制限があったが、ツールで生成したコードのほとんどのシンボルが16文字を越していたため、16文字以下にシンボル名を修正しなければならなかった。
（対策）
生成コードを一定のルールに基づき16文字以下になるように修正した。
（ex:Message->Msg）

今後の改善要望

前段で述べたような問題点を解決するために、あるいは弊社のビジネスと照らし合わせた中で、大変僭越ではございますが、今後ご検討いただけるとうれしい要望点を挙げてみたいと思います。

１．ASN.1ツールについては、シンボル名が重複しないように（ファイル名を付加するなど）していただきたい。
２．生成したコード容量の効率化・最適化オプションなどを検討していただきたい。
３．組み込み開発の世界では、まだまだ開発環境によって定義できるシンボル等の文字数に制限が存在するシステムも多く、シンボルの文字数を制限してコードを生成する機能があればより望ましい。
４．オブジェクト指向開発への対応
組み込み開発においても構造化からオブジェクト指向への開発環境の移行がだいぶ進んできており（例えば携帯電話端末においてSymbianOS、LinuxOS、Brew等のプラットホームが採用されていることでも顕著）、生成コードのクラス化、C++対応をぜひご検討いただきたい。
５．ツールの連携
キャッツ様のツールには「Drawrial」というUI設計ツールがありますが、これを連携させ、UI設計するだけでアプリケーションのフレームワークのコードまでを生成する機能の実現をご検討願いたい。
そうすることで、組み込みシステムのUI～ミドル、ドライバ層の近くまで上流設計後ソースコードを自動生成し生産性を大幅にUPさせる“統合開発ツール”としてより一層強力になると思われる。

以上