次世代組込みソフトウェア開発環境
Development Environment for Next-Generation Embedded Software

植木　克彦 UEKI Katsuhiko
長尾　圭浩 NAGAO Yoshihiro
松下　正樹 MATSUSHITA Masaki

組込みマイコンを利用した制御用ソフトウェアの需要は，この数年飛躍的に増大している。これに伴い，マイコンソフトウェアの開発が製品開発の重要な要素になっている。ここでは，次世代のマイコンソフトウェアの動向を踏まえ，当社が提案するマイコンソフトウェア開発環境 IDE(Integrated Development Environment)について紹介する。IDE の特長は，開発ツール群の統合によるマイコンソフトウェアのスパイラル開発に対応したシームレス開発環境を提供することであり，開発の加速と品質のよりいっそうの向上を可能とすることである。

With control software being increasingly embedded in microcomputers, the development of microcomputer control software is becoming an important factor in system development. This paper introduces the integrated development environment (IDE) proposed by Toshiba for the development of software for the next decade. IDE enables seamless development of control software embedded in microcomputers through the integration of development support tools, thus accelerating the speed of development and enhancing quality.

1 まえがき

近年のマイコン CPU パワーの増大に伴い，マイコンに搭載される制御用ソフトウェアは，種類，量ともに飛躍的に増大している。また，これらのマイコン制御用ソフトウェアの開発作業は多岐にわたり，煩雑になりつつある。しかし，マイコンソフトウェア開発環境はこれまでは開発のサイドサポートとして考えられる傾向があり，マイコンソフトウェア開発者のニーズに十分合致しているとは言い難いのが現状である。このため CPU メーカーには，メーカーが提供する CPU と親和性の高い開発環境を求める声が多く寄せられている。

当社も多種多様な高性能 CPU をユーザーに提供してきており，これらに対応したマイコンソフトウェア開発環境の早期提供を求められている。このため，当社の高性能マイコンに対応した次世代組込みソフトウェア開発環境の研究・開発を進めている。多様化，複雑化するマイコンソフトウェアの短期間での開発を可能とするものである。ここでは，これまでのマイコンソフトウェア開発の課題を整理し，これらの解決策として次世代組込みソフトウェア開発環境の基本コンセプトならびにその実現系としてのプログラム実装，テストの各サイクルの支援機能について述べる。

2 マイコンソフトウェア開発の課題

次世代組込みソフトウェア開発環境の開発にあたって，当社ではユーザー各社にインタビュー調査を実施し，その結果から次世代のマイコンソフトウェア開発スタイルを次のように描いている。

(1) マイコンソフトウェアの規模や複雑さは引き続き増大し，分散・分業による開発が必須(す)となる。
(2) 仕様定義，設計，実装を繰り返し行い，徐々に機能を実現していくスパイラル形の開発プロセスが主流となる。
(3) ソフトウェア IP(Intellectual Property)などの活用により，コーディングレスの範囲が増大する。また，開発規模の増加の影響もあり，結果としてマイコンソフトウェア開発に不慣れな者も開発に参加する傾向が強まる。
(4) 不慣れな開発者へのケアとして，可能な限りの自動化によってスキルの不足を補完する必要が生ずる。
(5) 製品サイクルの短縮により開発期間が現在以上に短縮され，短期集中開発となる。

当社では，このような次世代組込みソフトウェア開発のスタイルを想定しており，現在さまざまなツールベンダーなどより提供されているマイコンソフトウェア開発環境には，次のような課題があると分析している。

2.1 ツール操作性の課題

マイコンソフトウェア開発に不慣れな者が開発に参加することを想定して，現在のパソコンアプリケーション開発環境と同程度の，よりユーザーに親切な操作系を実現する開発支援環境を提供する必要がある。現状の開発ツールは，一連の開発作業の流れを意識した操作の一貫性などが必ずしも考慮されていない。また，不慣れな開発者をケアするような自動化機能，例えば I/O(入出力)や RTOS(Real

Time OS)の初期設定の自動化やテスト自動化なども十分に整備されているとは言い難い。

2.2 テスト・デバッグ作業の変化に対する課題

スパイラル型開発プロセスが主流になることにより，テスト・デバッグ作業を繰り返すある種のサイクル(テスト&デバッグサイクル)が形成され，さらにシミュレータを使ったテスト手法の一般化に伴いサイクル自体が変化する。また，ソフトウェアIPの利用によりブラックボックスを組み合わせたソフトウェアのテストが重要になる。しかしながら現状のデバッガなどは，このような開発プロセスやソフトウェアIP対応を考慮していないため，将来のテスト&デバッグサイクルに必ずしもフィットしたものにはなっていない。

2.3 分散・分業化に対応する課題

現在提案されている各種の開発環境は，分散開発で求められるさまざまなグループ開発対応機能を十分にサポートしていない。マネージャと開発者の双方を支援するために，例えば構成管理や設計情報共有化などが必要である。

3 次世代組込みソフトウェア開発環境IDE

ここでは，上述した課題に対する解決策として，当社で開発を進めている次世代組込みソフトウェア統合開発環境IDEの特長と支援コンセプトについて述べる。

3.1 IDEの基本コンセプト

IDEでは，図1に示すように，プログラム実装サイクル支援，テストサイクル支援，グループ開発対応支援の三つの大きな機能群を用意し，第2章で述べたマイコンソフトウェア開発の課題への対応を可能とするようくふうしている。

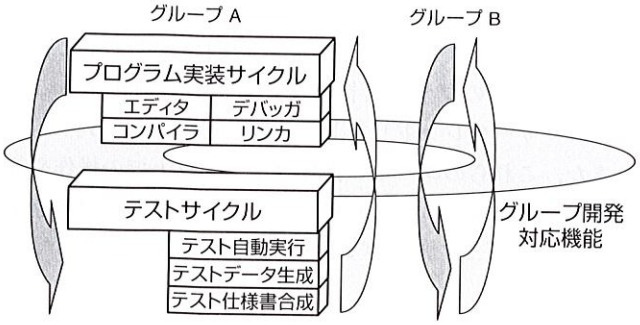

図1．IDEのソフトウェア開発支援コンセプト　プログラム実装サイクル支援，テストサイクル支援，グループ開発対応支援の三つの大きな機能群を示す。

Concept of software development support with IDE

3.1.1 プログラム実装サイクル支援機能

マイコンソフトウェア開発においては，設計仕様を実装に変換し，その動作を確認するプログラム実装作業のできいかんが，開発期間やマイコンソフトウェアの品質レベルを左右すると言っても過言ではない。不慣れな開発者や開発分業化，ソフトウェアIP利用などの要因を考えると，効率良く作業できかつ作業ミスが入り込まないようなプログラム実装サイクルを実現することは重要である。IDEにおけるプログラム実装サイクルでは，ソフトウェアIP利用を前提にしたコーディングレスを視野に入れている。コーディングからコンパイル，リンクといった一連の作業を中心にシームレスな作業環境を提供することで，快適なプログラム実装サイクルを実現することをねらっている。

3.1.2 テストサイクル支援機能

スパイラル化した開発プロセスにおけるテスト作業ではテスト作業自身も繰り返し行われることになり，結果として開発工数の増加を招きかねない。IDEでは，テストデータのターゲットマイコンソフトウェアへの投入から動作判定，不具合検出時のデバッグまでの一連のテストサイクルを連携し，テストデータの再利用などを可能とすることでテストサイクルの負荷軽減を実現する。

3.1.3 グループ開発対応支援機能

対象マイコンソフトウェア規模が増大し，複数開発者が分散・分業開発する場合，プロジェクトマネージメントが重要になる。特に，プログラム実装サイクルやテストサイクルにおける各開発担当者の進捗(ちょく)情報などはマネージャから把握が難しく，「ソフトウェア開発が見えない」といわれる原因になっている。コードやオブジェクトを対象とした構成管理機能や設計書などの中間成果物や情報の管理機能，バグや不具合などの管理機能，開発進捗管理などの機能の充実が求められる。各分散サイトは，すでにさまざまなプロジェクトマネージメントのしくみ(ルール)や支援ツールを利用しているが，IDEではこれらを考慮し，ユーザーが利用している外部ツールとの連携を容易に実現できるように配慮するとともに，アドオン機能としての提供も可能となるようアーキテクチャをくふうしている。

3.2 IDE利用による効果

IDEは上述したような三つの機能群を実現することで，次世代のマイコンソフトウェア開発に対し次のようなインパクトを与えることを期待している。

(1) 導入容易性の確保－初心者でも簡単　既存の開発環境では，導入まであるいは習熟するまでの負荷は初心者の場合かなりのものになり，結果として開発遅延を引き起こす大きな要因になっている。IDEを利用した場合，こうした負荷は従来に比べ相当量軽減され，結果として開発者が開発作業に集中することを可能とする。

(2) マイコンソフトウェア開発期間短縮と品質の向上　プログラム実装サイクルやテストサイクルのシーム

レス化により，単純な作業ミスが入り込む余地を減らすことができ，開発期間短縮に直結する。さらに，テストサイクルで提供するテスト自動化機構により，テストフェーズでの信頼性の大幅な向上が期待できる。

(3) 開発の可視化—マネージャも安心　グループ開発対応機能は，開発者の開発実務作業とマネージャの管理作業を共通のプラットフォームで行うことを可能とし，情報共有が加速される。これによりマネージャにとって悩ましかった開発可視化の課題が解決できる。

以下に，プログラム実装サイクル，テストサイクルについてIDEの概要を示す。

4 プログラム実装サイクル

IDEで実現するプログラム実装サイクルでは“シームレス プログラミング”，“ユーザーフレンドリ UI(User Interface)”などの機能的な優位性をもたせることで，プログラム実装作業の効果的な改善をねらっている。

4.1 シームレス プログラミング環境

マイコンソフトウェアのプログラム実装作業を詳細に見ると，I/OやRTOSの初期設定，マイコンソフトウェアプログラムコードの作成，これらのコンパイル，リンクといったビルド作業や，動作確認のためのデバッグ作業が順次あるいは繰り返し進められる作業であることがわかる。

従来のマイコンソフトウェア開発環境では，これらの一連の作業を行おうとした場合，OSコンフィグレータ，コードエディタ，コンパイラ・リンカ，デバッガなどの各種ツールを順次起動しなければならなかった。このような作業の進めかたは作業効率が低いことはいうまでもないが，作業の過程でデバッグ対象プログラムのバージョン取違えなどの単純な作業ミスが入り込むケースが少なくなかった。

IDEでは，こうしたプログラム実装作業を一連のサイクルとしてとらえた“シームレス プログラミング環境”を提供する。“シームレス プログラミング”とは，マイコンソフトウェアのプログラム実装における一連の作業を，同一プラットフォーム内で連続的に流していくもので，これにより上述の不具合要因の除去を図る。

図2は，従来の開発環境を利用した場合のプログラム実装フェーズの作業の流れと，IDEを利用した場合のプログラム実装サイクルを比較したものである。従来方式(図2(a))では外部エディタでソースファイルを選択してエディットするといった作業をはじめとして，デバッガによる不具合検出まで7ステップの作業が必要とされ，そのつど，必要なツールの起動やファイル設定などが要求された。これに対しIDEによるプログラム実装サイクルでは，OSコンフィグレータ，エディタ，コンパイラ・リンカ，

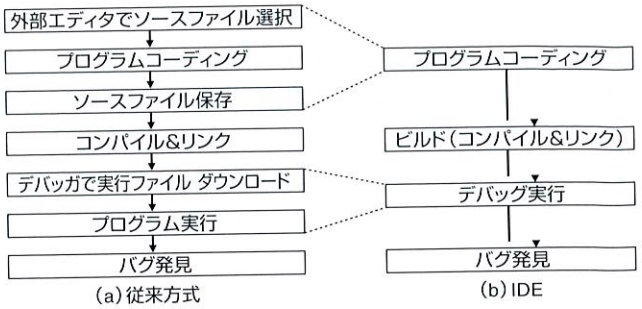

図2．プログラム実装プロセスの比較　IDEのプログラム実装サイクルは7ステップから4ステップに集約される。

Comparison of program making processes

デバッガなどのツールはIDE内部の動作設定シナリオによってユーザー作業の状況に応じて逐次自動起動される。これによりプログラム実装サイクルは(図2(b))に示す4ステップの作業に集約される。

4.2 ユーザーフレンドリ UI

現在のWindows®(注1)アプリケーションの普及はそのUIが視覚的でわかりやすいことが大きく影響していると言われている。マイコンソフトウェア開発環境については，これまでどちらかというと開発のプロ向けのUIを指向してきた。しかしながら背景でも述べたように，不慣れな開発者などが大量に開発に参加するようになった場合，マイコンソフトウェア開発環境といえども視覚的にわかりやすいインタフェースで開発をナビゲートすることが求められる。IDEプログラム実装サイクルではこれらを考慮し，よりわかりやすいUIをくふうしている。

例えば，従来，デバッガとコードエディタではツールが別なためコードの表示領域もそれぞれ独自に用意していたが，IDEではシームレスプログラミングを実現したことで，共通のコード表示領域をデバッガ，コードエディタの両方から利用することを可能としている。また，これ以外にもソフトウェアIPなどを利用した開発スタイルに合わせた各種情報のブラウズ機能なども極力共通化することで，ユーザーフレンドリなUIの確保を図っている。

また，これらのユーザー操作ウィンドウ上での操作や詳細設定は各種アイコンの利用やドラッグ＆ドロップ機能を充実し，ビジュアルなプログラミング環境を提供している。

5 IDEテストサイクル

開発期間の短縮傾向の加速によりマイコンソフトウェアのテストに充当できる期間もより短くなり，短期間で精度の良いテストを実施することが求められている。IDEテ

(注1) Windowsは，米国Microsoft社の商標。

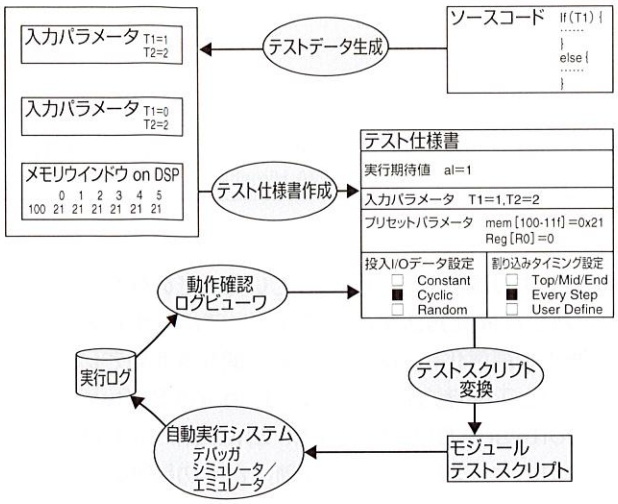

図3．IDEによるテスト自動化の例　　テスト効率向上とテストの質の向上を図る。

Example of testing automation

ストサイクルでは，こうした要求の解決を目的に“自動テスト”,“テストアイテム評価”などの技術を採用している。

5.1 自動テスト

テスト作業はテストデータ投入からテスト動作とその判定，不具合の検出と原因特定，修正といった一連の作業から構成される。IDEでは，図3に示すようにテストデータ(Test Vector)生成，テスト仕様書作成，テストスクリプト変換，テスト自動実行やテスト動作確認などの機能を実現し連携動作させることでテスト自動化をねらっている。

(1) テストデータ生成機能　　対象マイコンソフトウェアの入力パラメータを設定し，そのバリエーションを生成する。

(2) テスト仕様書作成機能　　入力パラメータ，投入I/Oデータなどを設定し，テスト時の動作シナリオを定義する。

(3) テストスクリプト変換機能　　テスト仕様書の情報を基に，テスト対象が受け取れる動作スクリプトの形に変換する。

(4) テスト自動実行システム　　デバッガ，外部エミュレータと連携しテスト対象をスクリプトに従って動作させる。

(5) テスト動作確認機能　　動作ログなどを自動記録し，仕様で設定された期待動作との対比によりテスト合否判定を行う。

IDEではこれらの機能を実現することで，テストの自動化によるテスト効率向上とテストの質の向上をねらっている。

5.2 テストアイテム評価

マイコンソフトウェアの規模や複雑さの増大に伴い，テストアイテムは指数関数的に増加している。IDEでは限られたテスト期間の有効利用をねらい，テストアイテムの順序付けと優先度付けの方式を提案している。これは通常，テスト戦略とも言われるもので開発経験の豊富な熟練者は無意識のうちに実践している。

IDEでは，これをより広範囲のユーザーが利用できるようにするため，対象マイコンソフトウェアの信頼性・ハザード分析法を活用して実現する。ハザード分析はシステムにとって好ましくない事象を体系的に分析する手法である。IDEではこれを利用して，マイコンソフトウェアにとって不具合を誘発するようなシステムコンディションを網羅的に抽出し，これに発生確率，マイコンソフトウェア動作順序などの要因を考慮してシステム信頼性側面からのテストアイテムのテスト順序や優先度付けの評価を行う。これにより，短いテスト期間の中でむだなく重要なテストアイテム，テストしやすいアイテムの順でテストし，テスト作業の効率を向上することをねらっている。またこれにより，マイコンソフトウェアにとっての信頼性側面から重要なテストを抜けなく実施することが可能となり，さらなる品質向上も期待できる。

6 あとがき

ここでは，当社で研究開発を進めているマイコンソフトウェア統合開発環境IDEを中心として，次世代のマイコンソフトウェア開発のあるべき姿の一端を紹介した。ここで示したIDEに含まれる一部機能については，ユーザー各位にすでに提供しており，好評をいただいている。今後とも当社マイコンユーザーの声を真摯(し)に受け止め，より使い勝手の良い効果的な開発環境を提供していきたい。

文　献

(1) K. Tamura, et. al.“A software Testing method based on Hazard Analysis and Planning”. Proc. of ISSRE '98. 1998, p.103-110.

植木　克彦　UEKI Katsuhiko
研究開発センター システム技術ラボラトリー。
テスト・デバッグ手法の研究・開発に従事。情報処理学会会員。
System Engineering Lab.

長尾　圭浩　NAGAO Yoshihiro
デジタルメディア機器社 COS開発センター 開発第二部。組込みマイコンのソフトウェア開発環境の開発・設計に従事。
COS Development Center

松下　正樹　MATSUSHITA Masaki
デジタルメディア機器社 COS開発センター 開発第二部。組込みマイコンのソフトウェア開発環境の開発・設計に従事。
COS Development Center