

組込みコンパイラによる開発と制御

田村 修†

マイクロコントローラ自身に軽量の言語処理系を内蔵させることにより、開発効率を高めるとともに長期的な保守を容易にする技術を紹介する。運用時の動作シーケンスのカスタマイズも迅速に行えることから、分散プロセッサシステムのコンフィギュレーションに応用した例を紹介する。

System Development and Control with the Built-in Compiler

Osamu Tamura†

In order to maintain embedded system in the long term, a lightweight language processor is built inside of a microcontroller. With the embedded compiler, the control sequence is easily customized during the operation. The example demonstrates the flexible configuration for a distributed multiprocessor system.

1. はじめに

組込み機器の開発、なかでも多品種少量生産の現場では、案件ごとに異なる CPU の採用や、同一システムでの異種 CPU の混在などによって、開発負荷が増大する傾向がある。また、互換チップの入手や開発環境の維持も含めた長期的な保守が難しくなることが懸念されている。ここでは、CPU メーカーごとに用意される統合環境でクロス開発する現行方式に対し、

- (1) アーキテクチャの仮想化
- (2) 基本的な開発環境の機器内蔵

により開発保守負荷の軽減を目指した組込み開発方式を提案する。

2. 基本構成

本方式では、アセンブラ記述した仮想スタックマシン上で軽量のコンパイラを動作させる(図1)。インタプリタ部分の移植のみでの異種 CPU への切り替えや、運用時のプログラム変更が容易となる。プログラムソースも内蔵すれば、開発環境の更新やリソースの散逸によって製作当時のバイナリを再構築できないなどのトラブルも回避できる。ここでは搭載メモリの少ないローエンドのマイコンをも対象としたため独自の言語を実装したが、処理能力に余裕があれば軽量化した Java による構成も考えられる。

3. 軽量コンパイラ

ローエンドマイコンに要する処理の多くは C 言語レベルで数十～数百行程度であることから、この規模の記述に耐え得る範囲で仕様を縮小したプログラミング言語を設計した。実装コストが小さく拡張性が高い FORTH を範に、フロー記述の見難さを改善しポインタ的操作を禁止するなど改変した。データとして変数、配列、文字列を扱える。C 言語の制御構文と演算子の多くに対応し、PostScript に似た記述をする言語とした。

コンパイラは、ソースコード中の単語を ID 化する処理がほとんどで、単語と生成される中間コードとが同順でほぼ一対一に対応する。インタプリタ部分は仮想 16 bit スタックマシンとして、中間コードに対応する処理を順次呼び出し、データスタックとリターンスタックを操作する。システム定義ワードの追加による高速化など、移植と拡張が容易な構造となっている。自己記述によるコンパイラの中間コードサイズは約 1.7K バイト、0.5K バイトの RAM で 30 程度のワードを定義できる。

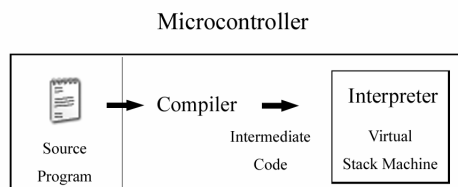


図 1 組込みコンパイル

Fig. 1 Embedded Compilation

† 有限会社リカージョン
Recursion Co., Ltd.

```

"\n\tHello, World.\n" %s // 文字列表示
:hello
;x ;y // 九九表
1 =y
{
  y 9 <= while // 1~9まで繰返し
  1 =x
  {
    x 9 <= while // 1~9まで繰返し
    x y * %3d // x * y を表示
    ++ =x
  } do .
  '\r' %c '\n' %c // 改行
  ++ =y
} do .
:multiply
1 > { dup -- self * } if // 階乗
:factorial

```

図2 プログラム例

Fig. 2 Examples of the ForCy Program

4. 実装例

4.1. ローエンドマイコンへの組み込み

マイクロチップ社の PIC 16F88 (ROM 4K 語, RAM 368 バイト) に実装した. RS-232C 回線より入力されるプログラムテキストを, 逐次コンパイルしてフラッシュメモリに書き込む. PIC16 アーキテクチャでは, 寸断された RAM 領域の連結, フラッシュ領域へのアクセス, 間接アドレッシングなどの手間により, 他のプロセッサに比べスタックマシン動作のオーバーヘッドが大きい. 20MHz クロックでは, 円周率 100 桁の計算に 8 秒を要した. 8 クイーン問題やクイックソートなどの実行により, 基本的なスタック動作を確認した.

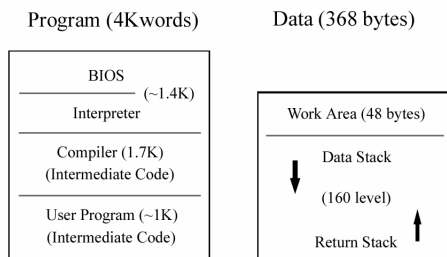


図3 メモリマップ (PIC16F88)

Fig. 3 Memory Usage on PIC16F88

Communication	2,400 bps xon/xoff
Compile	25 words/sec
Execution	12,000 empty loops/sec
User Program	~ 300 lines (PIC16F88 - 20MHz)

図4 プロセッサ処理性能

Fig. 4 Performance of the Processor

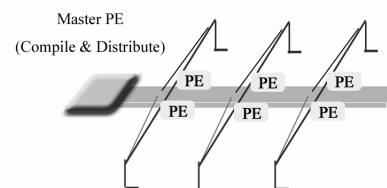


図5 分散プロセッサへのコンパイル配信

Fig. 5 A built-in Compile for Multiprocessor

4.2. 分散プロセッサへの応用

分散マルチプロセッサ環境において, 多数のプロセッサへの動作設定に応用した. ホストPC上で作成したプログラムを一括して管理プロセッサ (R8C/Tiny) に送り込み, 機器内部でそれぞれのプロセッサ (PIC12F683) 向けにコンパイル, 配信する例を開発ツールとともに紹介する.

5. まとめ

本方式は, 組み込み開発現場の負荷を軽減すること, なかでも長期的な保守を確実にを行うことに主眼を置いて考案し実装した. 有効性が期待される利用状況としてはつぎが考えられる.

- (1) 長期的な維持管理が必要とされるFA機器でのマイコン制御
- (2) 運用現場での動作シーケンスのカスタマイズを要する実験装置, シーケンサ
- (3) 多種多様なプロセッサを統合管理するスタンドアローンシステム (ロボットなど)
- (4) マイコン応用の学習教材 (テキストエディタとターミナルのみでプログラム可)
- (5) FPGA 内蔵カスタムCPUのプログラミング環境

なお, 本開発は 2005 年度 IPA 未踏ソフトウェア創造事業で支援されている.