制御システムにおけるシステム高速起動方式の実現

6D－7

馬場 儀之　摂津 敦　菅井 尚人
三菱電機（株）情報技術総合研究所

1. はじめに

近年、制御システムの形態が分散構成になり、個々のコントローラの多機能化が求められている。これに対して、応用プログラム（以下、ＡＰＰ）の継承性から上位のリアルタイムコンピュータを下位のコントローラ領域に適用する解があるが、電源断を含む障害からの復旧を迅速に行うために初期立ち上げ時間を最小化する必要がある。

本稿では、多機能なコントローラの、高速な初期立ち上げを行うための起動方式について述べる。

2. 現状の問題点と設計方針

コントローラの多機能化に伴って問題となる高速起動の阻害要因と現状の問題点は、以下の通りである。

（1）電源投入後、コントローラにて用いる大容量プログラム及びデータを格納する記憶装置へのアクセスが可能になる迄に時間を要する。このため、コントローラのブートデバイスは、初期化時間が少なく且つデータ利用時の手続きが少ない装置を選んでいる。しかし、この種の記憶装置は容量が少ない。

（2）多くの機能を利用することで、使用資源の初期化処理に時間を要する。このため、使用ハードウェア及びソフトウェアの性能向上を図っているが、開発コストがかかる上に、システム固有の解決策なので再利用性は低い。

（3）オペレーティングシステム（以下、ＯＳ）の初期化が終了してからでないとＡＰＰを起動できない。このため、ＯＳをＡＰＰが用いる機能だけで構成したり、ＯＳの資源確保量を最適に見積もって余分な初期化処理を省いている。しかし、これだけでは、逐次的な初期化処理におけるＩ／Ｏ待ち時間やコントローラが使用しない機能に関わる初期化処理時間によるＡＰＰ起動遅延がある。

そこで、本方式では、ＲＯＭ化可能且つ機能モジュール着脱可能なＯＳ構成を実現することで（1）を解決し、起動の基本フェーズでコントローラ基本機能を早期に立ち上げて高速起動が必要なＡＰＰを起動することで（3）を解決し、更に拡張フェーズで残りの拡張機能を立ち上げてコントローラ基本機能と結合させることで（2）を解決する方針とする。

3. 実現方法

3．1. 基本フェーズの実現

基本フェーズでは、コントローラ基本機能を高速に立ち上げ、高速起動を行うＡＰＰを起動する。

ＯＳは、最小構成と機能モジュールの着脱を容易にするために、マイクロカーネル技術を適用する。コントローラ基本機能は、高速起動のＡＰＰと、この動作に必要な拡張機能モジュール及びマイクロカーネルと、拡張機能を扱うファイルシステムモジュールと、拡張機能ローダで構成し（図1）、利用時に初期化時間が少なく且つ直接的に命令実行が可能な Flash-EEPROM 等の不揮発性メモリに格納する。多機能コントローラ実現時の、残りのソフトウェアのモジュール群は拡張機能部としてまとめ、大容量の外部記憶装置等に格納する。

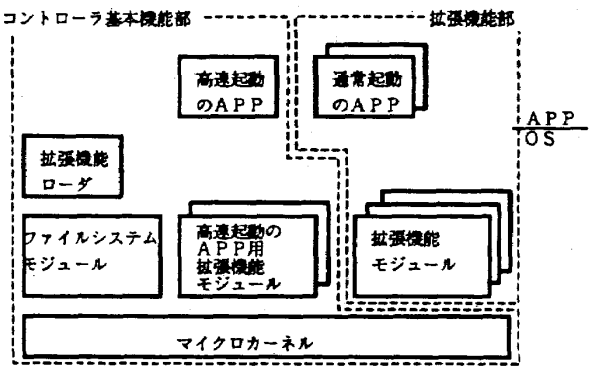

図1．ソフトウェア構成

A method of fast booting mechanism for control system.
Noriyuki BABA, Atsushi SETTSU, Naoto SUGAI
Mitsubishi Electric Corp.

3．2．拡張フェーズの実現

拡張フェーズでは、コントローラ基本機能と拡張機能を結合する。まず、拡張機能部のロード処理と初期化処理を行うが、ロード処理はコントローラ基本機能部に含めた拡張機能部ローダによって、初期化処理は各機能モジュール内に含めた初期化関数によって、いずれもマイクロカーネルが実現する並列処理環境で行う。

両機能の結合にあたって、概して不揮発性メモリのアクセスは主メモリより遅く、コントローラ基本機能が不揮発性メモリ上で動作することは好ましくない。従って、不揮発性メモリ上のコントローラ基本機能も主メモリ上に展開して、プログラム実行速度を向上させることとする。

このために、コントローラ基本機能部内の関数は、各起動フェーズでインストールするスタブ経由で実行する（図2）。このスタブでは、拡張フェーズ終了以前は不揮発性メモリ上に存在するコントローラ基本機能部内関数を呼び出す（①）。拡張フェーズでコントローラ基本機能を主メモリへ展開（②）した後、スタブの内容を主メモリ上のコントローラ基本機能部内関数を呼び出すように（③）書き換える。

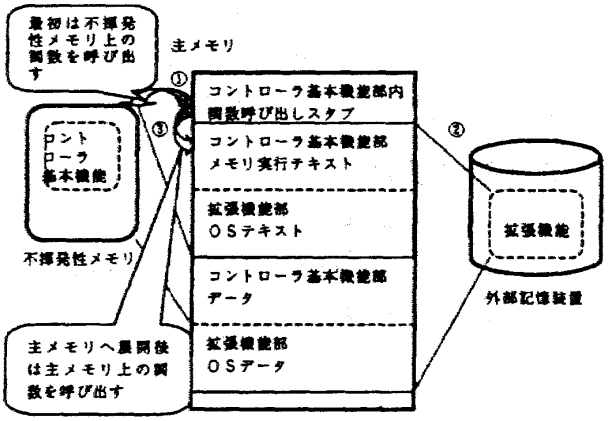

図2．主メモリの使用形態

両機能を結合させるために、両機能の実行プログラムを作成する過程を図3に示す。以下が、実行プログラム作成過程である。

（1）実行ファイル作成

ＯＳの生成手順で、コントローラ基本機能と拡張機能を含んだ実行ファイルを作成する。コントローラ基本機能のテキストは不揮発性メモリ上実行用と主メモリ上実行用を用意し、主メモリ上に展開するコントローラ基本機能のテキストは、リンカが両機能間でアドレス解決した主メモリ実行用を用いる。

（2）　不揮発性メモリ上実行プログラム部抽出

実行ファイルから、不揮発性メモリ上で実行するコントローラ基本機能を抽出する。不揮発性メモリ上のコントローラ基本機能を利用するためのスタブをインストールする関数を含む。

（3）　主メモリ上実行プログラム部抽出

実行ファイルから、主メモリ上実行用のコントローラ基本機能のテキストと拡張機能を抽出する。主メモリ上のコントローラ基本機能を利用するためのスタブをインストールする関数を含む。

（4）各機能部を配置

コントローラ基本機能部をパワーオンリセット後に実行する不揮発性メモリに、拡張機能部を大容量の外部記憶装置に配置する。

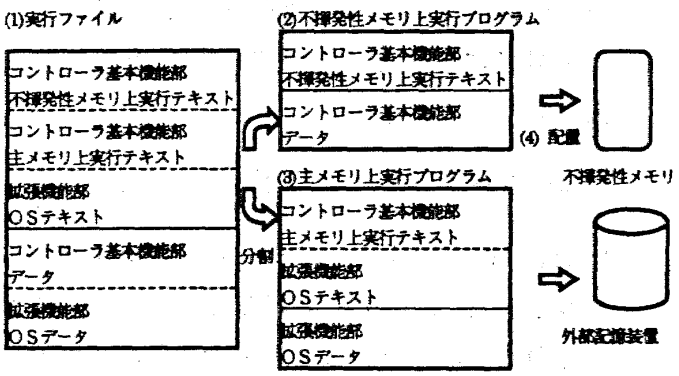

図3．実行プログラムの作成過程

4．おわりに

本稿では、多機能コントローラに要求される、初期立ち上げ時間の最小化に対する解決策について述べた。本起動方式により、コントローラ基本機能を高速に立ち上げて所望のＡＰＰを高速起動することが可能になり、その後に拡張機能を立ち上げることによって多くの機能を利用することが可能になっている。

今後、本起動方式の実装と評価を経て、製品適用を図って行く。