3D-1

制御プログラムへのＡｄａ適用法の検討

一條俊幸　、　田中寛
ＮＴＴ電気通信研究所

１．はじめに

Ａｄａ*[1]は、ソフトウェア工学の最新の成果を取り入れた流通性・信頼性・汎用性・生産性等に優れた言語である。近年の情報処理サービスの拡大・多様化に効率的に対処し、信頼性・保守性に優れた制御プログラム（ＯＳ）を実現するためには、Ａｄａで記述するのも有効な方法である。しかし現状では、一部の事例[2]を除いてＯＳへのＡｄａ適用事例は少なくＡｄａ適用時の設計・記述に関するガイドラインは明らかになっていない。

本論文は、ＯＳの一部をＡｄａで実際に記述し、設計・記述法等の観点から考察したＡｄａ適用時に考慮すべき事項を報告するものである。

２．記述対象

ＯＳのＡｄａ記述上の考慮事項を抽出するため、ハードウェア依存度が高く、制御表の参照／更新等について制御が複雑で、かつ適当な大きさのものとして、入出力要求実行処理を記述した。

３．設計・記述上の方針

（１）基本的には、既存のアセンブラ記述のロジックを踏襲するが、制御移行の単位で従来の処理を適当に分割し、記述性・読解性の良いモジュール構成とする。

（２）ハードウェア依存部分については、Ａｄａ機械語記述機能を用い、局所化する。

（３）Ａｄａ記述対象以外との整合をとるため、外部に共通な制御表の構成は変更しない。

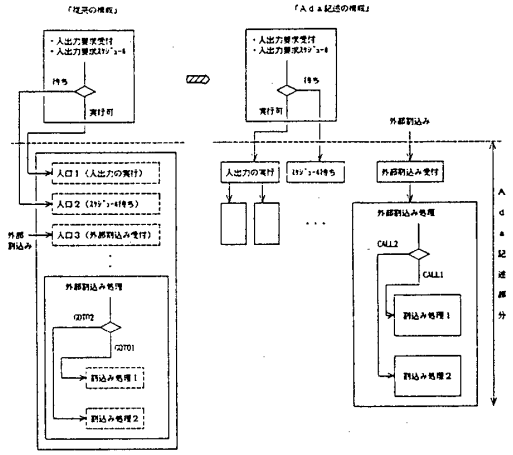

図１．入出力制御モジュールの再構成

４．入出力制御モジュールの構成

今回、読解性の向上のため、モジュール構成を図１．の様に改めた。主な変更点は、構造化の観点から、全体の流れの制御のみを行うメインコントローラなるモジュールを新たに作成したこと、アセンブラのブランチ命令（高級言語のgo to文相当）を全て手続き呼び出しに変えたこと、およびそれらに合わせて全体のロジックも変更したこと等である。その結果、制御の流れが明確になり、読解性は向上した。

５．記述上の問題点と対処

主要な問題点について、設計・記述性の観点から考察する。

（１）モジュール構成

モジュール構成の見直しにより読解性は向上したが、プログラムの呼び出し処理

＊Adaは米国政府AJPOの登録商標である

Examination on Ada Coding of the Control Program
Toshiyuki ICHIJHO , Hiroshi TANAKA
NTT Electrical Communications Laboratories

が増加した。即ち、各モジュールを呼ぶ毎に、レジスタの退避／回復処理、パラメータの設定処理等が生成される。性能重視の場合、特に次を考慮する必要がある。

①設計段階で、モジュール構成を十分に考え、外部に見せる必要のない手続きについては、内部手続きとする。

②読解性を損なわない範囲で適宜go to文も使用し、処理の効率化を図る。

（２）モジュール間インタフェース

（パラメータの授受）

Ａｄａでは、サブプログラム呼び出し時に、スカラ型のパラメータに更新可能指定をすると、呼び出され側の誤りを呼び出し側に波及させないために、パラメータの内容を一度コピーしてから渡し、リターン時に再度コピーし直す。これにより信頼性の向上にはつながるが、性能的には劣化する。これを避けるには、共通変数として定義する、あるいはレコード型のパラメータにするといった方法がある。その選択に当たっては、モジュール間のインタフェースに応じて、信頼性と性能のトレードオフで判断すべきものである。

（３）制御表の記述

ＯＳ特有の処理として制御表に関する処理がある。今回制御表の構成は変更しなかったので、フラグの参照／更新処理において、４ビット－１バイト間の変換が必要となり、非チェック型変換関数を多用することとなった。また、制御表の各変数の型の変更は、プログラムロジックの広範囲に影響を及ぼした。Ａｄａの特長を生かし効率的な制御を行おうとするならば、制御表もＡｄａに合った構成を考えるべきである。即ち、各モジュールから制御表の任意の箇所、長さのフラグ参照を許すことを避け、制御表を参照する各モジュールの機能を十分に把握し、その参照の単位を整理し、それに応じたフラグの定義を行う必要がある。

（４）ビットの定義

制御表の操作を“０”か“１”かの表現で行う場合には、型定義機能を用いて、ビット列に対応する型を自ら定義したい場合も生じる。その場合、整数型を制限（システム依存機能の内部表現節をもちいる）して“ビット列型”を表現（即ち取る値は０か１）しようとすると、オブジェクトは、一度整数型で取ったデータから数ビットだけ切り出す形となり、性能上好ましくない。従って、ビットを意識した記述を行い、高性能な記述を実現する場合には、論理型（取る値はtrueかfalse）を用いて定義すべきである。

（５）名標の記述

Ａｄａでは、データの宣言と参照について記述量が多くなりがちである。ＯＳ記述でもレコード型を多く用いるが、変数の参照に当たっては、変数をピリオドで区切って何段にもレコード名で修飾することになる。これは保守時における読解性を確保するのに有効であるが、記述量が増大し、また書いたプログラムも長さによっては見にくくなるので、参照が頻繁で短い名前を使いたい場合には、新たな１つの変数として記述する方法もある。

６．むすび

本Ａｄａ記述では、既存ＯＳの一部をコーディングし、制御表の構成等については既存の設計をそのまま引き継いだ。そのため、手続き呼び出し、制御表の処理について、記述量の増大、記述の複雑さや性能劣化を招いた。Ａｄａの持つ信頼性等の特徴を生かし、高性能な記述を実現するためには、他の高級言語を使用する以上に、設計段階で機能全体について十分な設計及び記述法の検討が必要である。今後、性能を意識した記述法に加えて、ＯＳの流通性、機能の拡張性を確保するためのガイドラインの作成が必要である。

７．謝辞

本研究に当たり、御指導頂いた情報処理方式研究室大橋室長に感謝する。

参考文献

１）ＵＳ　ＤＯＤ：Reference Manual for the Ada Programming Languag ,ANSI/MIL-STD-1815(1983)

２）細川　馨：仮想計算機システムの再設計、情報処理、vol.27. NO.3