マルチメディア通信と分散処理　44－1
（１９９０．１．２６）

実時間制御用メッセージの ＡＳＮ．１符号化／復号化高速処理方式

勝丸　郁子　　福澤　淳二　　寺田　松昭
（株）日立製作所システム開発研究所

実時間要求が厳しいセルレベルネットワークにおける機器制御用メッセージのＡＳＮ．１符号化／復号化処理方式を提案し、処理性能を実測により評価した。提案方式の特徴は、符号化／復号化処理高速化のため、（１）予め符号化した転送形式の情報をテーブルの形で用意しておくことによって、メッセージ毎に必要な符号化／復号化処理を削減し、（２）メッセージと転送形式間の変換をこのテーブルを用いて直接行い、データ転写回数を削減した点にある。本方式に基づく符号化／復号化処理部を作成し、性能を実測した。これにより、メッセージの要素数、要素長の増加に従い本提案方式の効果が大きくなることを示した。

An ASN.1 encoding/decoding method for the real-time control message

Ikuko Katsumaru　Junji Fukuzawa　Matsuaki Terada
Systems Development Laboratory,Hitachi Ltd.
1099,Ohzenji,Asao-ku,Kawasaki,215,Japan

An ASN.1 encoding/decoding method for the device control message at time critical networks is discussed with its experimental evaluation. To improve the encoding/decoding speed, we proposed the method in which the transferring data format table is pre-encoded to skip the encoding/decoding process and every messages is translated directly using this table to reduce the number of times of data copying. This method was evaluated by implementing the system and the result showed that the effect of this method becomes more evident when the number and the length of the message elements increased.

１．はじめに

組立ライン内のロボット、プログラマブルコントローラやＦＡパソコン等の機器間を接続するセルレベルのネットワークにおいては、指令から応答受信までの応答時間に対する要求が厳しい。そこで、このような環境を対象として、ＯＳＩ７層構造の３層から６層までの処理を省略し、応答時間短縮を狙った通信規約が提案されている[1]。

この規約では、第６層のプレゼンテーション層で行うデータ表現形式の変換処理を各応用層プロトコル要素が行う。応用層プロトコル要素の構文変換処理は、データの表現形式が異なる機器間でメッセージの交換を可能とするため、共通の転送データ表現形式（転送形式）を定め、要求・応答メッセージと転送形式との変換を行なう。この変換処理は、メッセージの長さ及び、内容の複雑さに関係するため、変換に時間がかかる。応答時間に対する要求を満たすために、この変換処理の高速化が１つの課題となる。

構文変換処理高速化に対しては、変換規則そのものを改良するアプローチ[2]もあるが、マルチベンダ環境を実現するためには、標準変換規則であるＡＳＮ．１(Abstract Syntax Notation One)符号化規則に基づいて、変換処理の高速化を行なう必要がある。本稿では、ＡＳＮ．１符号化規則に基づく変換処理の高速化方式として構文予約方式を提案し、変換処理時間を実測した。

以下、２章において構文変換処理の概要を、３章において構文変換処理実装上の課題を、４章において実時間制御用メッセージに適用する変換方式を提案し、５章において提案方式の変換処理時間実測結果を述べる。

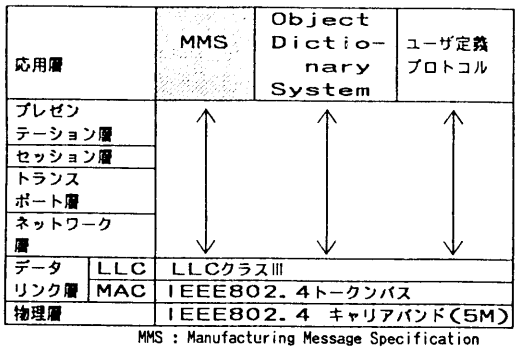


図１　実時間制御用通信規約の概要

２．構文変換処理の概要

実時間制御用通信規約の一例を図１に示す。この通信規約では応用層プロトコルとして、コントローラと機器との間の指令・応答（メッセージ）を標準化したＭＭＳ（Manufacutuaring Message Specification）[3]を採用している。本稿では、このＭＭＳをとりあげその構文変換処理方式の考察を行う。

２．１　ＭＭＳにおける構文変換処理

ＭＭＳは機器制御用に約９０種のメッセージを定義している。ＭＭＳを用いて機器を制御する場合、次の手順でメッセージの転送を行う（図２）。まず、コントローラ側のユーザプログラムから発行された要求メッセージをＡＳＮ．１符号化規則に従って、転送データ形式（転送形式）に変換し、下位層に渡す。下位層は変換された要求メッセージをデバイス制御装置に送信する。デバイス制御装置のＭＭＳは、受信した転送形式を要求メッセージに逆変換し、デバイス制御プログラムに渡す。デバイス制御プログラムはこの指令に対する処理を行ない応答メッセージを作成してＭＭＳに渡す。ＭＭＳは同様の処理を行ってコントローラ側の応用プログラムへ応答を返送する。このように、１回の指令に対して、メッセージと転送形式間の変換が合計４回必要となる。

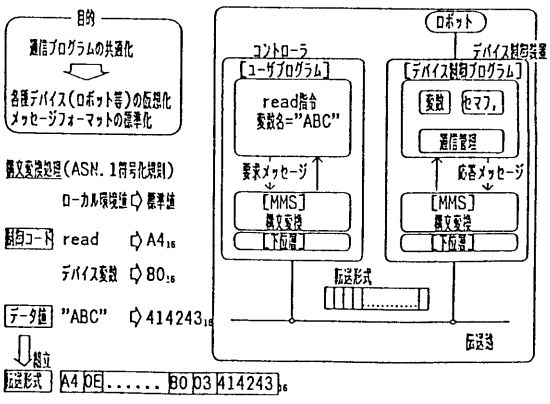
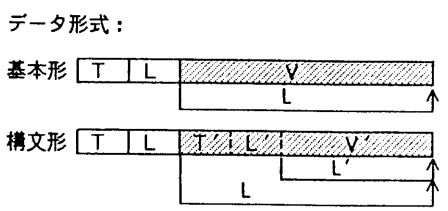


図２　ＭＭＳの概要

２．２　転送形式の構造

ＡＳＮ．１符号化規則に従って符号化された転送形式の構造は複雑である。ＡＳＮ．１では、図３に示すように、メッセージ内の各データを識別子（Ｔ）、長さ（Ｌ）、コンテンツ（Ｖ）の３要素に符号化する。識別子はデータの種類、長さは符号化されたデータの長さ、コンテンツは符号化されたデータを表す。長さフィールド（Ｌ）の内容は、コンテンツ（Ｖ）の長さによって変わる。コンテンツはデータ内容によって常に最も短いコードに変換される。このためＡＳＮ．１符号化規則を用いた転送形式の構造は、発行メッセージ毎に異なる。



T、T′：指令内データの識別子（１バイト以上）
L、L′：符号化したデータの長さ（１バイト以上）
V、V′：符号化したデータ（０バイト以上）
基本形の場合、指令内に設定されたデータ（整数値、文字列等）が設定される
構文形の場合、識別子、長さ、データが入れ子構造になる

図３　転送形式の構造

３．構文変換処理実装上の課題

３．１　要求条件

実時間制御に使用されるメッセージの構文変換処理に対しては以下の要求がある。

（１）変換処理の高速化

実時間制御を行なうためには、応答性が重要となる。例えば、組立ラインにおいて実時間制御を行なうためには、５０～１００ｍｓの応答時間が要求される[1]。指令から応答受信までの１トランザクションで、メッセージと転送形式間の変換が４回行なわれる。応答時間要求を満足するために、この変換処理の高速化が必要となる。

（２）移植性

ＭＭＳはＰＣ、ロボット等のプログラマブルデバイス上に実装される。プログラマブルデバイス上では、デバイスを制御するためメモリ等の多くの資源が必要となる。このため、プログラマブルデバイス上では、構文変換処理で使用する資源（メモリ）をできるだけ削減する必要がある。

３．２　従来方式の問題点

ＡＳＮ．１符号化規則に従った構文変換処理に関して、従来次のような報告がされている。

（１）中間形式経由方式[1][2]

この方式では、応用層プロトコルと構文変換処理部との間に中間形式を定め、構文変換処理部はこの中間形式と転送形式間の変換を行う（図４）。構文変換処理部は、ＡＳＮ．１によって記述された変換規則を木構造形式のテーブルに展開し、中間形式と転送形式間の変換をこのテーブルを参照して各メッセージ共通に行う。

（２）直接変換方式[3]

この方式では、メッセージと転送形式間の変換を構文変換処理部が直接行う（図５）。構文変換処理部は、ＡＳＮ．１によって記述された変換規則をプログラムに組込み（以下ダイレクトコーディングと呼ぶ）、メッセージ毎に転送形式との変換を行う。

上記方式を実時間制御用メッセージに適用した場合、次のような問題がある。

中間形式を設けると、メッセージの表現形式が異なる応用層プロトコルにも変換処理部を汎用的に使用できる。しかしこの方式では、中間形式への変換が符号化／復号化処理毎に必要のため、変換時間の劣化が予想される。

変換処理を、テーブルを参照して行なうことにより変換処理を共通化できる利点がある。しかしこの方式は、変換時にテーブルサーチを行なうため、メッセージ毎に最適なコーディングが行えるダイレクトコーディング方式に比べ処理時間がかかる。しかし、ダイレクトコーディング方式は、メッセージ単位に変換処理を記述するためプログ

ラム規模が大きくなる。

変換処理性能の面では、中間形式を経由せず、メッセージと転送形式間の変換をダイレクトコーディングする方式が最も効率が良い[1]。しかしダイレクトコーディングした場合、全てのメッセージについて個別に変換処理が必要なため、プログラマブルデバイス上に実装する場合プログラム規模が問題となる。

4．実時間制御用メッセージに適用する構文変換処理

4．1　構文予約方式

（1）高速化の検討

3．2で述べた従来方式ではメッセージ、あるいは中間形式から転送形式へ変換する場合、コンテンツの長さを求めるために、符号化したデータを一旦ワークエリアに退避しておき、先頭要素より順次転送形式へ変換してゆく。この方式では、符号化時にデータ転写が2回発生し変換処理の劣化を招く。しかし、転送形式の最終要素から符号化してゆけば、符号化したデータの長さを求めながら転送形式への変換が可能となり、データ転写の回数を削減できる。すなわち、転送形式を逆方向から変換することにより、変換処理の高速化が図れる。

（2）プログラム規模削減の検討

ＡＳＮ．1で記述されたＭＭＳのメッセージは、図6に示すように、メッセージに対して1種類の変換規則しか持たないメッセージ（以下、固定形メッセージと呼ぶ）と、複数の変換規則を持つメッセージ（以下、不定形メッセージと呼ぶ）とに分類できる。

固定形の場合、変換規則が一定のため、メッセージ内容の解析を行なわず、メッセージと転送形式を1対1に対応付けることができる。従って、固定形メッセージについては、転送形式の構造とメッセージ内データの対応付けを行なったテーブル（以下、構文定義テーブルと呼ぶ）を用いることにより、メッセージ固有の変換処理を行なわず転送形式の組立／分解処理を共通化できる。

これに対し不定形の場合は、メッセージ内容によって変換規則が選択される。従って、不定形メッセージについては、転送形式の組立／分解の際、メッセージ固有の変換処理が必要となる。

上記検討に基づき、実時間制御用メッセージに適用する構文変換処理として構文予約方式を提案する。本方式では、転送形式へ逆方向から変換し、符号化処理を高速化する。固定形メッセージについて、構文定義テーブルを用い符号化／復号化処理を共通化し、プログラム規模の削減を図る。以下、本方式において使用する構文定義テーブルの構造と、これを用いた符号化／復号化方式を述べる。

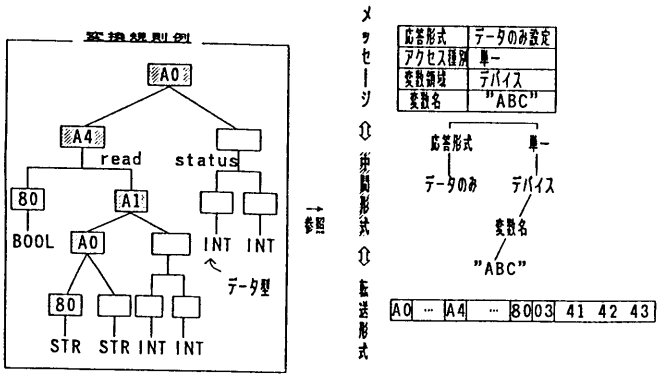


図4　中間形式経由方式

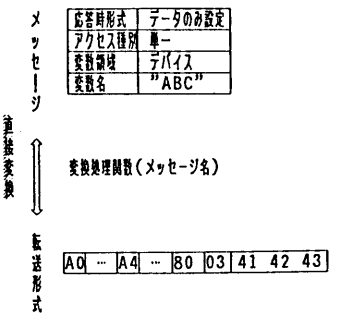


図5　直接変換方式

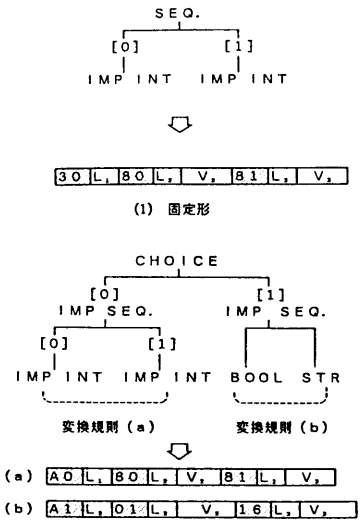
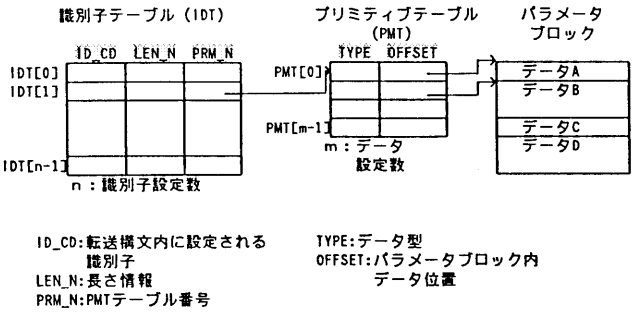


図6 固定形／不定形メッセージの変換規則



ID_CD:転送構文内に設定される識別子
LEN_N:長さ情報
PRM_N:PMTテーブル番号
TYPE:データ型
OFFSET:パラメータブロック内データ位置

図7 構文定義テーブルの構造

4．2 構文定義テーブルの構造

構文予約方式で使用する構文定義テーブルの構造を図7に示す。このテーブルは、識別子と転送形式の構造を定義した識別子テーブル（ＩＤＴ）と、メッセージ内のデータの位置、及び型情報を設定したプリミティブテーブル（ＰＭＴ）からなる。

ＩＤＴには、転送形式で使用する識別子コード（ＩＤ＿ＣＤ）と、コンテンツの範囲を示す情報（ＬＥＮ＿Ｎ）を設ける。ＬＥＮ＿Ｎを参照することにより、コンテンツの長さを逆方向から求めることができる。ＰＭＴには、メッセージ内のデータ形式を記述した型情報（ＴＹＰＥ）と、メッセージ内のデータ設定位置を示したフィールド（ＯＦＦＳＥＴ）を設ける。ＯＦＦＳＥＴを参照することにより、メッセージと転送形式間のデータ授受を直接行うことができる。

4．3 符号化／復号化処理方法

構文予約方式の概要を図８ａ、ｂに示す。

（１）固定形メッセージの変換処理

固定形メッセージの場合は、メッセージに対応した構文定義テーブルが予め作成されている。符号化の際、メッセージを受け取った構文変換処理部は、メッセージに対応する構文定義テーブルを選択する。構文定義テーブルから転送形式へ変換を行う場合、ＩＤＴの最終情報より順に符号化を行なう。まず、ＰＭＴ内の情報に従ってデータを符号化し転送形式設定用送信バッファの末尾に符号化データを設定する。次に、ＩＤＴ内の範囲情報に従ってコンテンツの長さを符号化し、識別子コードとともに送信バッファに設定する。

復号化の場合は、転送形式の先頭要素を復号化し、該当する構文定義テーブルを選択する。転送形式の復号化は、ＩＤＴの先頭要素より行なう。転送形式の情報が、ＩＤＴ内の識別子コード、範囲情報と一致した場合、ＰＭＴ内の情報に従ってデータを復号化し、メッセージ内に設定する。

（２）不定形メッセージの変換処理

不定形メッセージの符号化は、高速化のため変

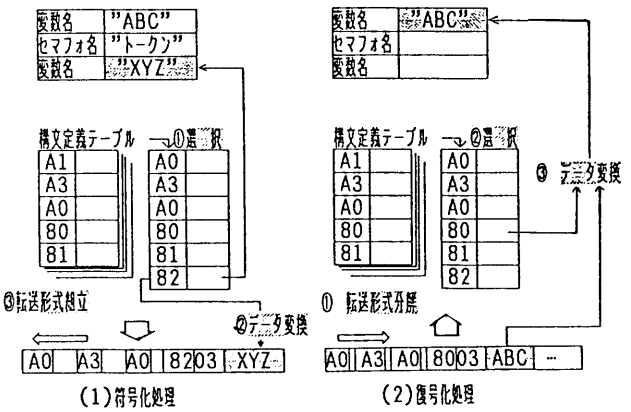


図8a 構文予約方式(固定形メッセージ)

換規則をダイレクトコーディングする方式とする。この時、データ転写の回数を削減するために、メッセージから構文定義テーブルを作成する。構文定義テーブルから転送形式への変換時には、固定形メッセージの場合と同様の手順で行なう。

復号化の場合、まず転送形式より構文定義テーブルへ変換して、固定形メッセージと同様の手順で復号化することもできる。しかし、この方式は中間形式を経由する方式と同様、変換処理の劣化が予想される。このため、復号化時には、直接変換方式と同様、変換規則をダイレクトコーディングして、転送形式を分解してゆく。

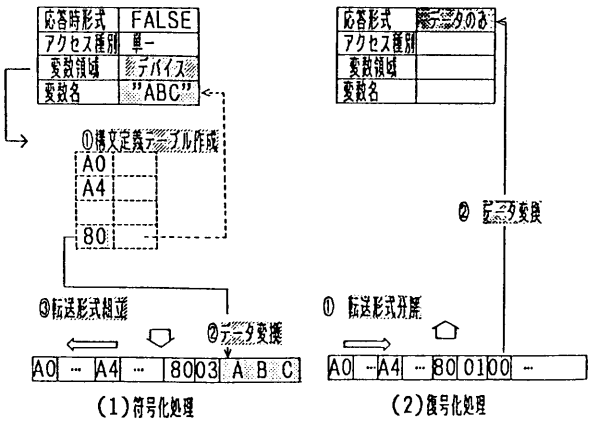


図8b 構文予約方式(不定形メッセージ)

5．変換処理時間実測結果

5．1　測定内容

本方式の性能を次の２点につき直接変換方式と比較する。

（１）構文定義テーブルを使用した変換処理の性能

直接変換方式では、メッセージ毎に内容解析を行なって個別に変換処理を行なっている。提案方式では、構文定義テーブルを使用することにより、メッセージ内容の解析処理を省略している。この効果を、固定形メッセージを使用して測定する。

（２）データ転写回数削減の効果

直接変換方式では、符号化データを一旦ワークエリアに退避している。提案方式では、符号化時

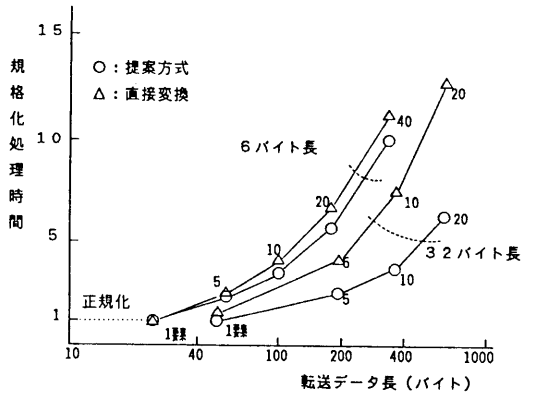


図９　固定形メッセージ符号化処理時間

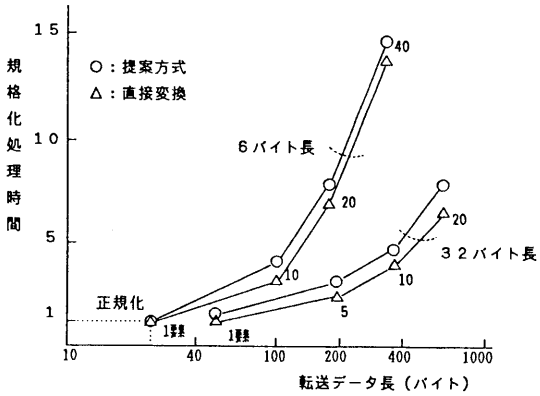


図１０　固定形メッセージ復号化処理時間

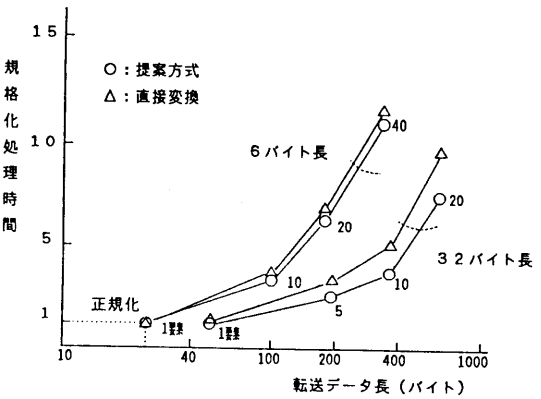


図１１　固定形メッセージ符号化／復号化処理時間

に転送形式へ逆方向がら変換してゆきデータ転写回数を削減している。この効果を、不定形メッセージの符号化処理について測定する。

５．２　測定方法

提案方式と直接変換方式の双方の構文変換処理部をＣ言語で作成し、１ＭＩＰＳのミニコン上で符号化／復号化処理性能を測定する。測定には以下のメッセージを使用する。

固定形メッセージとして、ＭＭＳのgetNameList応答を、不定形メッセージとして、read応答をそれぞれ使用する。getNameListでは、変数名の長さと要素を、readにおいては変数のデータ長と要素を変えて性能を測定する。

５．３　実測結果

固定形メッセージ、不定形メッセージの符号化／復号化処理時間を、要素数１、要素長６バイトにおける提案方式の変換時間で規格化し、以下の図に示す。

図９、図１０、図１１に固定形メッセージについての符号化復号化処理時間を示す。符号化処理については、１要素のデータ長、および要素数が小さい場合、直接変換方式と提案方式の性能はほぼ同等となる。しかし、データ長、および要素数の増加に従って提案方式の効果が大きくなる。復号化処理の場合、提案方式では構文定義テーブルを参照して従来方式よりも複雑な処理を行なうが、従来方式とほぼ同等の性能となった。符号化／復号化処理全体については、変数名の長さ３２バイト、要素数２０の場合で約２０％の性能向上となった。

図１２、図１３に不定形メッセージについての符号化処理時間を示す。図に示すように、データ転写を削減した場合、１要素のデータ長が大きいほどその効果が大きくなる。復号化処理については、直接変換方式、提案方式共同じ方式を使用している。図１４に示すように不定形メッセージの符号化／復号化処理全体については、データ長３２バイト、要素数２０の場合で約３０％の性能向上となった。

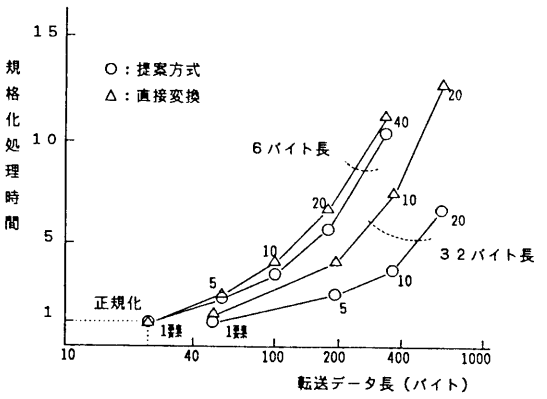


図１２　不定形メッセージ時間
（要素数を変えた場合）

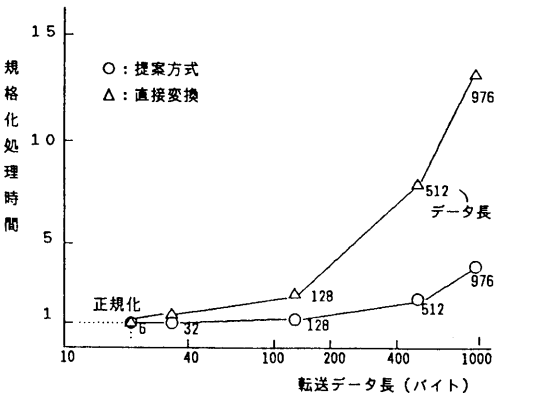


図１３　不定形メッセージ符号化処理時間
（データ長を変えた場合）

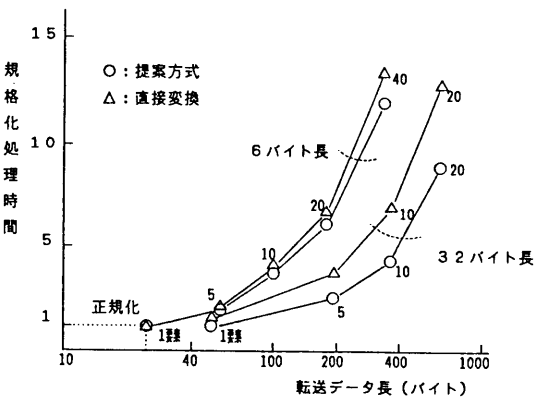


図１４　不定形メッセージ符号化／復号化処理時間
（要素数を変えた場合）

5．4　考察

提案方式と3．2で述べた従来方式を、変換処理性能、移植性について比較した結果を表1に示す。

（1）変換処理性能

直接変換方式と、中間形式を経由する方式と比較した場合、前者は後者の約2～3倍の性能が得られる[1]。直接変換方式と提案方式を比較した場合、固定形、不定形メッセージ双方で直接変換方式と同等以上の性能が得られた。この傾向は要素数、及びデータ長が大きな場合に特に顕著となる。

（2）移植性

移植性についてはプログラム規模が問題となる。中間形式を経由する方式では、応用層内においてメッセージと中間形式間の変換処理が必要となる。また、同様にダイレクトコーディング方式においても、メッセージ単位に変換処理を記述する必要がある。提案方式では、固定形メッセージについては個別の符号化／復号化処理が省略できる。実時間制御用に実装されるメッセージの内、約半数が固定形となる。提案方式はこのメッセージ分、プログラム規模の削減が行なえる。

6．おわりに

実時間制御環境に適したＡＳＮ．1符号化復号化方式を提案した。本方式では、中間形式を介さず直接転送形式の変換を行って、変換回数及び変換時のデータ転写回数を削減し変換処理の高速化を図った。また、変換規則が固定のメッセージについては構文定義テーブルを予め作成しておくことにより、プログラム規模を削減することができた。本方式の性能を測定した結果、直接変換方式と同等以上の性能が得られた。

謝辞

本研究の機会を与えて下さった（株）日立製作所システム開発研究所所長堂免信義氏、同4部部長大町一彦氏、並びに本研究を行うにあたり御指導、御協力を頂いた関係者各位に深謝いたします。

参考文献

[1]国際ロボットＦＡ技術センタ、"ＭＡＰ　ＦＡ実現へのかぎ"

[2]J.R.Pimentel,"EFICIENT ENCODING OF APPLICATION LAYER PDU's FOR FIELDBUS NETWORKs",ACM Computer Communication Review 18,'88.3,14-44

[3]ISO,MMS 9506

[4]中川路、勝山、水野、"ASN.1ツールAPLICOTの設計"情報処理学会第35回全国大会、5U-9,'87.10

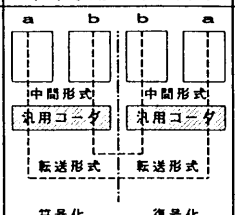
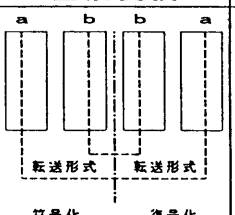
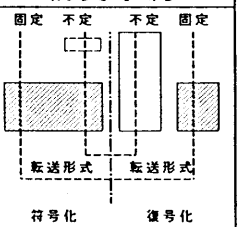
[5]姉崎、"ASN.1ハンドラの設計と実装"、情報処理学会マルチメディア通信と分散処理研究会、36-1,'88.2

[6]長谷川、野村、堀内、"ASN.1支援ツールの開発"－コンパイラ及びエディター"、情報処理学会マルチメディア通信と分散処理研究会、39-4、'88.9

[7]C.Partridge,"A Comparison of External Data Formats",Proc.IFIP TC6 Working Symp.,'88.10,39-59

[8]長谷川、野村、"ASN.1からCへのコンパイラの評価"情報処理学会第38回全国大会、7M-7,'88.3

表１　方式比較

方式		中間形式経由	直接変換	構文予約
（メッセージ形式）プログラム構造 メッセージ追加時 □：可変処理 ▨：共通処理				
比較項目	変換速度	× （1）	○ （2~3倍）	○ （同左以上）
	移植性	△ メッセージ／中間形式間変換要	× メッセージ毎に変換要	△ 不定形の場合変換要