通信制御プログラムにおける部品化プログラミング方式†

木村正男†† 大林恵次††
上森明†† 山下博之††

本論文では，通信制御プログラムにおける生産性と品質の向上を目的とした部品化プログラミング方式およびその方式を用いたプログラムの試作結果を述べる．前置処理装置上で動作する通信制御プログラムの作成に当たっては，多数の通信制御手順を短期間でサポートする必要があり，多数のプログラマにより開発が進められる．したがってプログラム開発上，多様なプログラマの資質に依存せず短期間で高品質のプログラムをいかに効率よく作成するかが重要な課題となる．本論文で提案する部品化プログラミング方式の特徴は次のとおりである．①通信制御プログラムの構成を，状態と入力によりルーチンを選択し起動させる固定的な処理と，プロトコルを実現する多数のルーチン群とに分割し，ルーチンを部品として扱う．②部品化されたルーチン群をデータベース管理システム上にデータベース情報として構築し活用する．本方式により，類似した手順間および同一手順内でのルーチンの共用度を高くすることにより高生産性を達成できる．また，データベース管理システムのもつ検索機能を有効利用でき，設計・製造上の手作業を削減することによりプログラムを高品質化できる．本手法により物理レイヤ，データリンクレイヤの機能をもつ通信制御プログラムを試作した結果，生産性および品質の向上に有効であることがわかった．

1. まえがき

前置処理装置上で動作する通信制御プログラムの作成に当たっては，多数の制御手順を短期間でサポートする必要があり，多数のプログラマにより開発が進められる．したがってプログラム開発上，多様なプログラマの資質に依存せず短期間で高品質のプログラムをいかに効率よく作成するが重要な課題となる．また，保守工程における制御手順の変更・追加に対しても柔軟性のあるプログラム構成とし，効率よく対処できるようにしておく必要がある．

本論文では生産性と品質の向上を目的とした通信制御プログラムの設計手法について述べる．生産性向上の技法として，従来，仕様記述言語等の研究が行われている[1),2)]が，仕様の定義だけを正確にしても次に続く設計へ円滑につながらないという問題があり，近年はコードの再利用や部品化技術による自動プログラミングを目指した研究が行われる動向にある[3)]．部品をベースとしたプログラミング手法の研究は一部に報告されている[4)]が，実用化されているものは見受けられない．

本論文では，通信制御の機能と構造自体を整理することにより，(1)プログラムの構成要素であるルーチンの共用度を高くして部品化すること，(2)部品化されたルーチンをデータベースで管理しながらこれを用いて構造化設計すること，により生産性と品質を向上する設計手法を提案する．

通信制御機能は，交信主体間で特定の制御データを授受しながら情報データの送達確認を行うという特徴があり，制御中心型設計法が適合する．本論文では，制御中心型設計で用いられる有限状態オートマトンモデルをベースとして，プログラム機能を判断処理と制御処理に分割し，制御処理を細分化することにより，ルーチンの部品化を可能とする手法を述べる．また，これら部品をデータベースで管理しプログラムの設計・製造を支援することによりプログラムによる人手作業の削減を可能とした．この手法により，プログラム変更・追加作業を迅速化することができ，単純ミス等によるバグの混入を防止することができ，高生産性・高品質を達成することができる．

2. 通信制御プログラムの原理と部品化の可能性

2.1 通信制御プログラムの構成と動作原理

(1) 状態遷移制御の原理

状態遷移制御を行う通信制御プログラムの動作原理は有限状態オートマトンの動作として表現でき，その動作は以下の5組によって定義される[5)]．

$[X, I, O, N, M]$

† Parts-Based Programming Method in Communication Control Program by MASAO KIMURA, KEIJI OHBAYASHI, AKIRA UEMORI and HIROYUKI YAMASHITA (Communication Control System Section, Yokosuka Electrical Communication Laboratory, N. T. T.).

†† 日本電信電話公社横須賀電気通信研究所通信制御研究室

ただし，X：状態集合，I：入力集合，O：出力集合，
M：出力関数 (M; $X*\ I \rightarrow O$),
N：状態遷移関数 (N; $X*\ I \rightarrow X$).

出力関数 M は，ある状態 x のときに入力された情報 i により出力 o (そのときに実行すべき動作) が決まることを意味する．状態遷移関数 N は，ある状態 x のときに入力された情報 i により次の状態 x が決まることを意味する．このように関数 M, N はともに，そのときの状態 x と入力 i のみに依存する．このためオートマトンの動作は，状態を節点（ノード）とし，節点間の有向枝（アーク）を状態遷移として，有向枝上に入力と出力を記述した「状態遷移図」で表現することができる．**図1**に簡単な状態遷移図の例を示す．

（2） 通信制御プログラムの構成

図1のような状態遷移制御を行うプログラムを実現するためには，a. 制御対象の状態を保持する機能，b. 関数 M を定義する機能，c. 関数 M に基づき出力 o を選択する機能，d. 出力 o を実行する機能，e. 関数 N を定義する機能，f. 関数 N に基づき次の状態 x を選択する機能，を有する必要がある．本通信制御プログラムの構成は以下に示す考え方から，a を状態管理表，b を状態遷移マトリクス，c をスケジューラ，d, e, f をルーチンとして実現する．通信制御機能は階層化されているため，プロトコルの各レイヤごとにこれらの構成とした．一つのプロトコルレイヤの機能を実現する部分を総称してモジュールと呼ぶ．

（イ） 状態管理表

通信制御プログラムでは複数の回線を同時に制御するため，回線対応に管理表をもち各回線の動的な状態を保持する．また，回線ごとに制御手順が異なるため管理表上に使用する制御手順の種別を表示する．

（ロ） 状態遷移マトリクス

状態 X, 入力 I, 出力 O (ルーチン) からなる関数 M を状態遷移マトリクスとして管理する．状態遷移マトリクスは状態遷移図で省略されているすべての組合せを網羅的に規定できる．一つのプロトコルレイヤには複数の制御手順が存在しうるため，状態遷移マトリクスは制御手順ごとに設定する．

（ハ） スケジューラ

スケジューラは入力が発生した場合，状態遷移マトリクスを用いて所定のルーチンを選択するプログラムであり，その動作は次節で述べる．

（ニ） ルーチン

ルーチンは**図2**に示すように外部からの入力に対し，種々の資源を使用して所定の動作を実行する機能と考えられ，1ルーチンを1出力 o に対応づけることができる．ルーチンは状態遷移図上の遷移時の処理に該当し，一つの制御手順は状態遷移マトリクスに従った多数のルーチンの動作により実現される．

（3） 通信制御プログラムの動作

モジュールに対する入力情報としては，制御対象である回線番号と指示内容であるコマンド（以降，トリガと呼ぶ）が必要となる．入力情報が通知されてからルーチンを選択するまでの動作順序は次のとおりであり，その概要を**図3**に示す．

① スケジューラは入力情報から対象とする回線番号を得て，これを基に当該回線の状態管理表を得る．

② 状態管理表上の制御手順種別から，使用する状

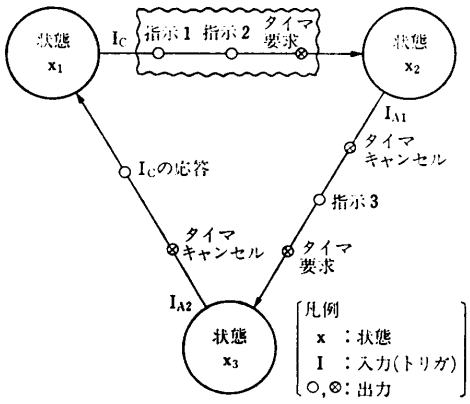


図1 状態遷移図の例
Fig. 1 Example of state transition diagram.

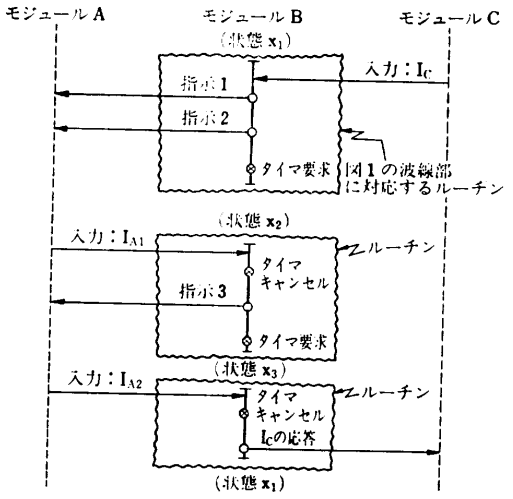


図2 ルーチンの例
Fig. 2 Example of routine.

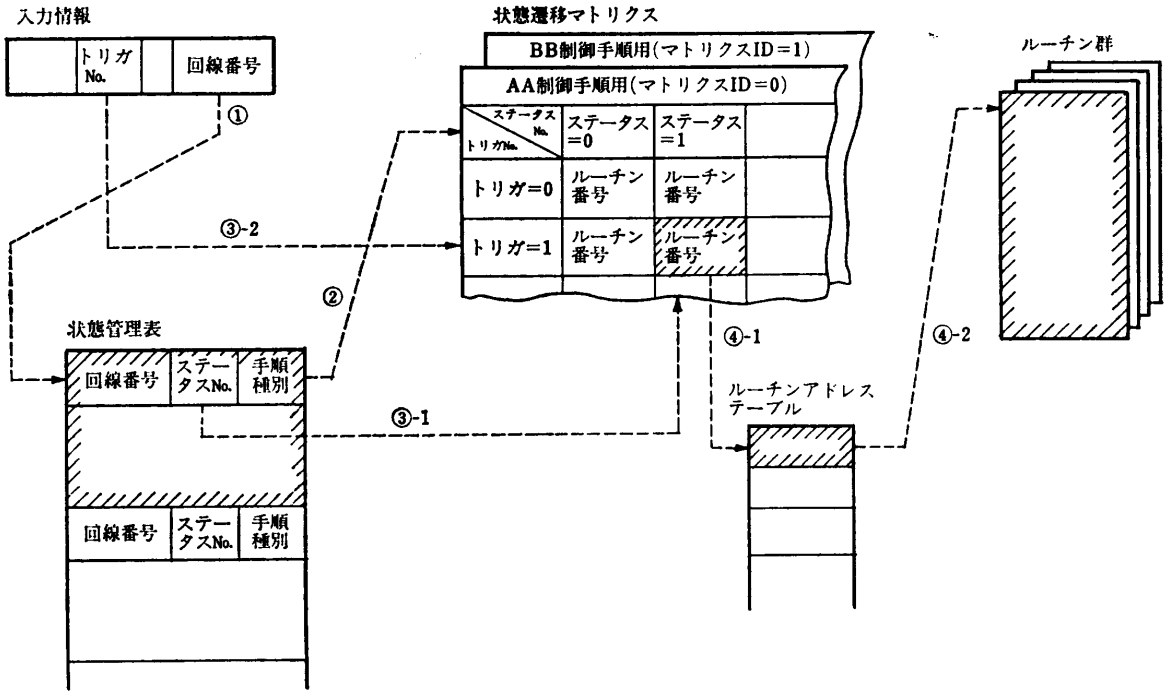


図 3 ルーチンのスケジュール過程
Fig. 3 Routine scheduling process.

態遷移マトリクスを選択する.

③ 状態管理表上の現時点の状態表示と入力情報中のトリガから状態遷移マトリクス内に規定されているルーチン番号を得る.

④ ルーチン番号からルーチンアドレスが判明し,当該ルーチンが起動される.

⑤ ルーチンは出力処理実行後,次の状態を状態管理表に設定してスケジューラに制御移行する.

2.2 部品化の可能性と限界

通信制御機能の規定として,同一制御手順内での異常処理は同一となることが多く,類似した制御手順間では正常処理も含め同一となる場合が多い.したがって,別々の状態遷移マトリクスであっても同一ルーチンを共通資源として部品的に使用できる可能性がある.

しかし通常,ルーチンの処理中には判断分岐処理があるため,ルーチンの共用度はあまり高くならない.すなわち,入力情報により処理の内容,遷移先状態が変わるためルーチン走行後でないと遷移先状態が決定できず,ルーチン内に遷移先状態を埋め込む必要が生じる.したがって,共用化するためには動作だけでなく遷移先状態までも一致していなければならず,ルーチンの共用度が高くならない.

次章では,上記の欠点を除去しルーチンの共用度を高くして,部品化するための手法を述べる.

3. 部品化プログラミング手法

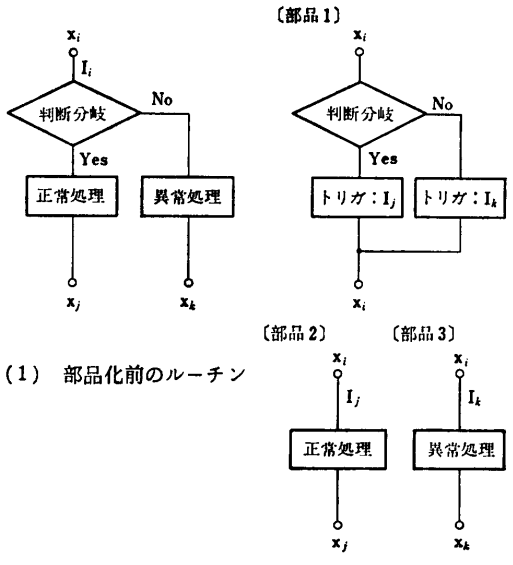
3.1 ルーチン分割による部品化

(1) ルーチン分割法

図 4 に示すようにルーチンの処理中に判断分岐がある場合には判断結果に応じて新規のトリガを発生させ,そこで処理を中断し,状態遷移は行わずにルーチンが走行する以前の状態のままにする.以降の処理は新規に発生したトリガの種類により,別ルーチンが継続して実行する.通常,ルーチンの走行ルートは状態,トリガおよびトリガに付随する入力データに依存する.ここで採用した方式は,入力データに依存して走行ルートが変わる部分を判断ルーチンおよび新規トリガにより起動される実行ルーチンに分割し,分割されたルーチン(これを部品化ルーチンと呼ぶ)の走行ルートを状態とトリガのみの関数とする方法である.このようにすればルーチン内の動作内容は状態とトリガにより一意に決まり,共用度を高くできる.

(2) 部品化ルーチンのスケジュール方式

部品化ルーチンによりルーチンと遷移先状態の関係が1対1となるため状態遷移処理をルーチンからスケ



（1） 部品化前のルーチン

（2） 部品化後のルーチン

図 4 ルーチン分割による部品化の例
Fig. 4 Example of decomposed routine parts.

ジューラに移行できる．すなわち，状態遷移マトリクスに関数 N を定義させる機能をもたせ，スケジューラがその情報により管理表上の状態を遷移先に変更する．異なる遷移先状態でも実行する機能が同じであればルーチンを共用できるため共用度を高くできる．

3.2 ルーチン分解による小部品化

フローチャートを基にアセンブラ命令を記述してルーチンを作成する作業では，通信制御機能実現上のマクロ的な処理をマシン属性（命令，レジスタ等）に依存したミクロ的な処理に変換しているため思考観点が変わり，プログラマの資質によりプログラム品質にかなり差が出てしまう．この問題点に対処するため，通信制御プログラムでは基本的な処理の組合せが多いことに着目し，フローチャート上に記述される最小機能単位をマクロ言語（以降では小部品と呼ぶこともある）化した．小部品はアセンブラ命令の集合で構成され，チーフプログラマが設計・製造する．ルーチンは小部品の集合で構成され，多数のルーチン作成プログラマがこの小部品のみを用いてコーディングする．このためバグは少なくなりプログラム品質は均一になる．マクロ言語の特徴は以下のとおりであり，通信制御向きとしマシン属性を排除した．

a．言語の機能単位は通信制御機能実現上必要となる資源対応に**表 1** に示す分類で設定する．

b．言語間での引継情報はもたせず，記述順序の制約も原則としてもたせない．

表 1 マクロ言語の分類
Table 1 Category of macro languages.

No.	分　類	機 能 概 要
1	トリガ出力制御	他モジュール/ルーチンへの要求，通知
2	タイマ制御	タイマ種別の指定とリクエスト/キャンセル要求
3	バッファ制御	バッファ種別の指定と確保/解放要求
4	カウンタ制御	カウンタ種別の指定とインクリメント/デクリメント
5	テーブル制御	テーブルへの書込み，参照
6	プログラム制御	ルーチンスケジュール処理への戻り等

3.3 部品の組合せによるプログラムの構造化設計

上記で述べてきたように本通信制御プログラムの構成は，モジュール，状態遷移マトリクス，ルーチン，マクロ言語，アセンブラ命令と多階層化されており，上の階層からみて下の階層の機能が部品的に使用できる．制御構造の特徴としては，以下が挙げられる．

a．ルーチンの選択，起動するまでの論理が固定．

b．ルーチン間の情報引継方法が固定（トリガに集約される）．

プログラムの機能変更・追加があっても，上記の各階層の部品群を改造・追加・入替えするだけで柔軟に対処することができる．

3.4 部品化の欠点

（1） 性能・メモリ量に与える影響

部品化ルーチンではルーチンの構造は単純となるが，新たなトリガを追加することによりスケジュール回数が多くなり動的ステップ数が増加し，状態遷移マトリクスのサイズも新しいトリガの分だけ大きくなる．

（2） 開発作業の操作性に与える影響

a．通信制御機能は多数のルーチンの組合せにより実現されるため，個々のルーチンの品質がよくてもルーチン間にまたがった処理矛盾があれば問題となる．本方式ではルーチン規模が小さくなり，手順上の一つの処理は複数のルーチンに分散されて実現されるため，ルーチン間の処理矛盾を見つけにくい．

b．デバグに伴いルーチンおよび状態遷移マトリクスを修正する必要があるが，トリガ数，ルーチン数が多いためその管理が煩雑となる．

c．ルーチンを手順間および同一マトリクス内の種種の箇所で共用しているため，修正に伴い他の手順/箇所への影響が把握しにくい．

4. 性能と操作性の改善策

本章では，前章で述べた性能，操作性に関する欠点の改善策について述べる．メモリ量についてはメモリの低価格化の将来動向を踏まえ対象外とする．

4.1 性能の改善策

動的ステップ数の増加を極力防ぐため，スケジュールがルーチンを選択する際，判断処理により発生するトリガの場合には短絡スケジュールを行うようにした．すなわち，スケジューラは先行する判断ルーチンを選択する際に使用した回線番号，状態管理表および状態遷移マトリクスをそのまま使用し，後続する実行ルーチンを選択する．短絡スケジュールでは，状態とトリガによりルーチン番号を選択するだけでよいため，性能劣化を抑えることができる．

4.2 操作性の改善策

開発作業上の操作性の問題に対処するため，通信制御プログラムを構成する各要素（状態遷移マトリクス，ルーチン，マクロ言語等）の内容を部品管理データベースとして構築し，データベース管理システムがもつ検索機能を活用した．また，データベース管理システムのもつ機能は汎用的であるため，本通信制御プログラムの制御方式にあわせて，データベース上のデータを入出力・解読・編集するアプリケーションプログラムを充実させることにより設計・製造・試験・保守の作業の一部を自動化する支援システムを作成した．部品管理データベースの概要は次のとおりである．

（1） 状態遷移マトリクスの管理法

状態遷移マトリクスは表形式で表現され，本通信制御プログラムでは状態，トリガ，ルーチン，遷移先状態の関係がすべて1対1となっているため，データベース化しやすい．**図5**に示すように状態番号，入力（トリガ）番号，出力（ルーチン）番号，遷移先状態番号をドメインとするデータベースで構成できる．1状態遷移マトリクスは{(ステータス数)×(トリガ数)}分のタプルによって表現される．

（2） ルーチンの管理法

1ルーチンの構成はマクロ言語の集合で表現され，入口も出口も一つであり通信制御の機能としては分岐はない．したがって，1ルーチンは相反する二つの機能をもつことがなく，タイマ，バッファ等の各資源対応にどのようなアクセスを行うのか順番に列挙すればルーチンを規定することができる．各資源はプロトコルレイヤごとに有限であり基本機能種別として位置づけ，アクセス方法をマクロ言語とそのパラメータで表現する．これはルーチンの有する機能を基本機能種別，マクロ言語，パラメータという階層的な分類で管理しているともいえる．これにルーチンを識別するためのルーチン番号とルーチン内でのアクセス順序を入れればルーチンをデータベース化できる．したがって，**図6**に示すようにルーチン番号，基本機能種別番号，ルーチン内マクロコール順序，マクロ言語およびそのパラメータをドメインとするデータベースで構成でき，1ルーチンを複数タプルで表現した．

（3） マクロ言語の管理法

マクロ言語の展開形はアセンブラ命令の集合で構成されるが，これらはデータベースとして管理しない．この理由は，データ投入工数とデータベースでの利用

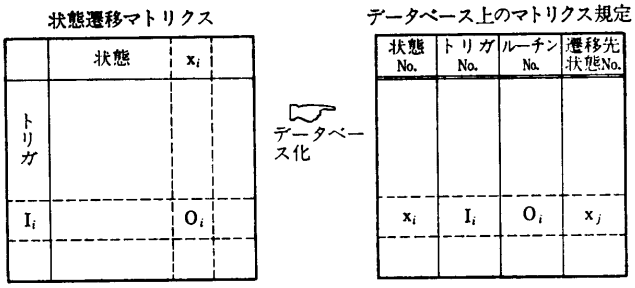
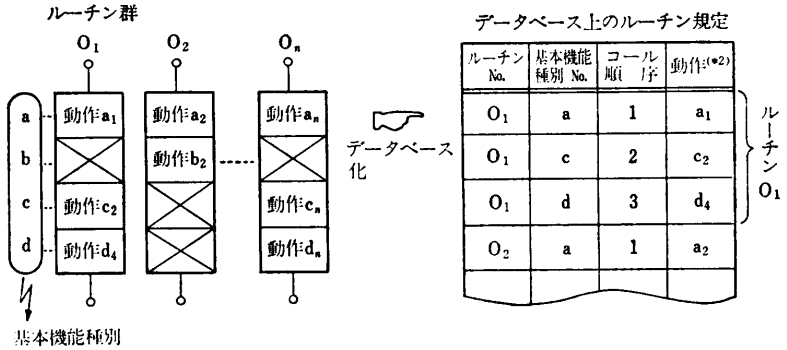


図5 状態遷移マトリクスのデータベース化
Fig. 5 Expression of state transition matrix in database.



(*1) ⊠ は該当機能を使用しないことを示す．
(*2) マクロ言語およびパラメータ

図6 ルーチン群のデータベース化
Fig. 6 Expression of routines in database.

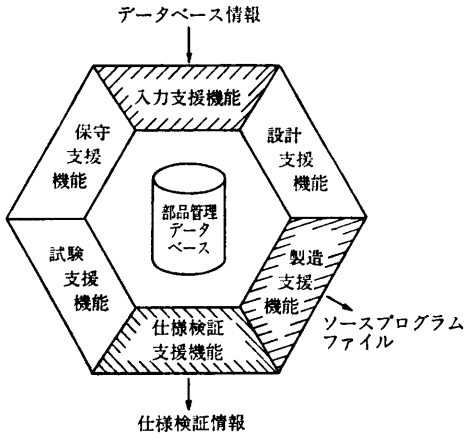


図 7 部品化プログラミング支援システムの機能
Fig. 7 Main function of parts-based-programming support system.

効果を考えた場合，通信制御処理上のものに限定したほうがよいためである．言語ごとの動的ステップ数等，設計に必要なものは管理する．すなわち，データベースを活用してマシン属性に依存する処理の検証は行わない．マクロ言語の管理内容は，ルーチン作成プログラマに対して新規ルーチンを作成する際の利便を図るため，該当するプロトコルレイヤのために存在するマクロ言語の機能情報をもてばよい．したがって，プロトコルレイヤ，基本機能種別，マクロ言語，パラメータをセット関係とし，おのおのその意味を並記したデータベースで構成した．

（4） 支援システムの機能

支援システムにおける機能としては**図 7** に示すものがある．ここでは，運用中の入力，仕様検証，プログラム製造等の支援機能について述べる．

入力支援は，データベース情報の入力作業を容易にする機能をもつ．仕様検証支援は，設計/試験時点でのレビューの際，状態遷移マトリクスに従い動作するルーチン群の起動順序を解析する機能および起動されるルーチンの内容を編集・出力する機能をもつ．プログラム製造支援は，データベース上の状態遷移マトリクス情報およびマクロ言語で記述されたルーチン情報に従って，コンパイラの入力情報であるソースファイルを作成する機能をもつ．

5. 評　価

本論文で提案した部品化プログラミング方式（部品化設計手法と部品管理機構）の有効性を把握するため，プロトコルレイヤのうち物理レイヤ，データリンクレイヤの制御手順を試作した．

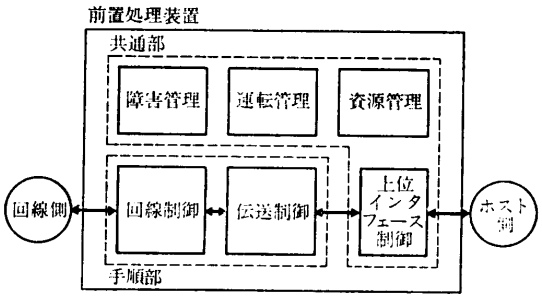


図 8 試作プログラムのモジュール構成
Fig. 8 Module structure of experimental communication control program.

5.1 試作プログラムの概要

試作プログラムは**図 8** のように共通部と手順部の 2 サブシステムから構成される．共通部は，資源管理，障害管理，運転管理，上位インタフェース制御の 4 モジュールから構成され，制御手順を動作させるための共通的な機能を分担する．手順部は，回線制御，伝送制御の 2 モジュールから構成され，おのおの，物理レイヤ，データリンクレイヤにおける複数の制御手順を実行する機能を分担する．提案手法は手順部に適用した．

5.2 生産性向上効果

本論文では，ルーチン部分に対する生産性の向上度はルーチンの共用度に比例すると考える．ルーチンを部品として共用できれば，類似機能を新規に作成する

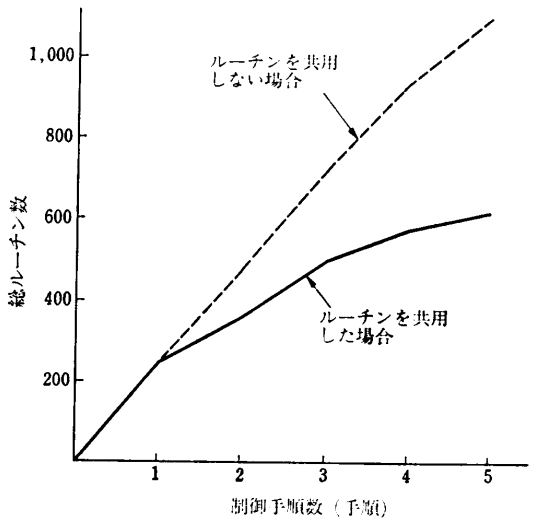


図 9 ルーチンの共用化効果の例（回線制御の場合）
Fig. 9 Example of routine-parts-reusability (a case of line control protocols).

表 2 ルーチンの共用化効果の例（ハイレベル伝送制御の場合）
Table 2 Example of routine-parts-reusability (a case of HDLC protocol).

No.	処理内容	ルーチン数		格子数		共用化率（格子数/ルーチン数）
		個数	%	個数	%	
1	回線オープン/クローズ	9	3.3	47	0.7	5.2
2	データリンクの確立	20	7.2	46	0.7	2.3
3	データリンクの解放	37	13.4	80	1.1	2.2
4	データ受信	29	10.5	64	0.9	2.2
5	データ送信	78	28.3	867	12.1	11.1
6	障害後処理	99	35.9	1310	18.3	13.2
7	ダウン処理（論理矛盾）	4	1.4	4730	66.2	1182

(注) 総ルーチン数：276個，総格子数：7144個，
平均共用化率：26格子数/ルーチン数

より設計・製造・試験に要する工数がその分不要となり，生産性が向上するという考え方である．

（1） 制御手順間でのルーチン共用効果

物理レイヤにおいてサポートした制御手順数と総ルーチン数の関係を図9に示す．制御手順間で共用できたルーチン数は約 50% であった．すなわち，ルーチン部分に対する生産性向上度が約2倍になるとみなせる．

（2） 1制御手順内でのルーチン共用効果

データリンクレイヤにおいて HDLC クラス BA 制御手順を作成した場合のルーチン数，格子数（状態遷移マトリクスでの状態とトリガの交点を格子と呼び，この格子数に対応させ制御手順でのすべての動作を規定する）および共用度を処理内容別に計上したものを表2に示す．表2よりルーチンの共用度は平均 26 カ所/ルーチンとなる（ダウン処理を除いても 9 カ所/ルーチン）．また，他のルーチンとの相互関係をもたないダウン処理が全体の格子数に対して 60～70% を占めているため，ルーチン間にまたがった処理論理の検討は状態遷移マトリクスのうち 30～40% のものに対して行えばよいことがわかる．

5.3 品質向上効果

モジュール別の検出バグ数は図 10 に示すとおりであり，共通部に比べ手順部のバグ件数が少ない．また，デバグ過程におけるバグ検出件数の時系列的な推移（図 11）からわかるように，手順部のほうが共通部よりも早期にバグが枯れやすい．これは次の理由によると考えられる．

（1） データベース管理がもつ検索機能を活用することにより，検討もれ等による設計ミスを早期に防止でき，バグ混入率が少なくなる．

（2） ルーチンが複数の手順間で共用されるため，多くのプログラマによりチェックされバグ混入率が少なくなる．

（3） プログラム構造が状態遷移マトリクス，ルーチン，マクロ言語と階層化されているため，各レベルごとに並行デバグすることができ早期バグを摘出できる．

5.4 実用性に関する考察

（1） 性能・メモリ量について

性能については，部品化前と後を比較すると約 6%

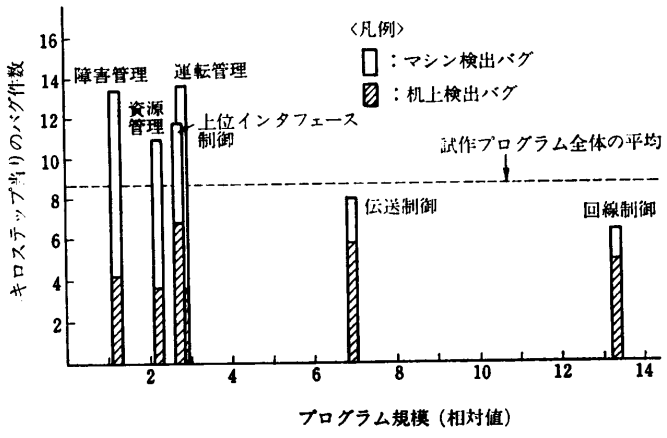


図 10 モジュールごとのバグ検出状況
Fig. 10 Detected bugs of each module.

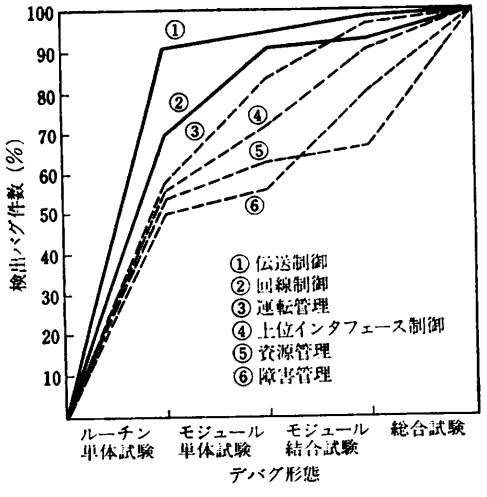


図 11 モジュールごとのバグ摘出推移
Fig. 11 Detected bugs of each module in debugging process.

程度の動的ステップ数増加であった．なお，部品化前のステップ数については，シミュレーションランおよび机上での計算により推定した．メモリ量については，部品化後のマトリクスサイズが約3倍に増加した．

（2） データベース利用により期待される効果

操作性・実作業時間に関して期待される定性的な効果は次のとおりである．

① 同一のデータベースからソースプログラム・ファイルを自動生成できるとともにプログラム設計書等のドキュメントがマクロ言語を基本とした様式で自動的に出力できる．バグ修正はデータベース上での修正を基本とするため，デバグに伴い従来のフローチャート相当の修正（手作業）は不要となる．

② 仕様検証支援機能では，ルーチンの起動順序が検索でき，実マシン上では確認しにくい通信制御機能の処理矛盾を容易にチェックできる．設計誤りはバグの 50% 以上を占めるため，この機能を有効に活用すればバグ混入を防止し，早期に検出できる．

③ データベース上での部品管理が可能となるため，ルーチンを修正する際，そのルーチンを起動する状態とトリガのすべての組合せを簡単に検索することができ，共用化ルーチンの修正を行ってもよいかが判断しやすくデグレードが事前に防止できる．

6. むすび

通信制御プログラムの生産性と品質の向上をねらいとした部品化プログラミング方式について述べた．プログラムの部品化の原則は，次のとおりである．

［原則1］ プログラムの構成要素であるルーチンを，判断処理と実行処理に分割して部品とする．

［原則2］ ルーチンにおける状態遷移処理を除去し，スケジューラが当該処理を行う．

［原則3］ ルーチンがもつ機能を細分化してマクロ言語として小部品化し，ルーチン作成時はこの言語のみで記述する．

［原則4］ 部品，小部品をデータベースで管理し，データベース機能の支援により設計・製造を行う．

本手法は，通信制御プログラムがもつ状態遷移制御論理の特長を活かしたルーチンの部品化とこれらを組み合わせた階層化モジュール構成法の採用によりルーチン共用度を向上させることおよび部品管理データベースを用いて製造・試験・保守などの作業を自動化することにより生産性の向上，バグ混入率の極小化，バグ摘出・措置の迅速化を図ったものである．本手法によりプログラムを試作した結果，生産性および品質に対して効果的であったことを確認するとともに，本手法の実用性についての見通しを得た．今後の課題としては，（1）上位のプロトコルレイヤに対する適用，（2）作成済のデータベースを活用した別マシンへのプログラム移植の検討，（3）汎用の部品化プログラミング技術への展開等が考えられる．

謝辞 最後に，本研究の機会を与えていただいた横須賀電気通信研究所データ通信研究部新井克彦部長，苗村憲司統括調査役，ならびに終始御指導いただいた日本電信電話公社研究開発本部酒井保良調査役に深く感謝いたします．

参考文献

1) Stenning, N. V.: A Data Transfer Protocol, *Comput. Networks*, Vol. 1, No. 2, pp. 99-110 (1976).
2) Brand, D. and Joyner, W. H.: Verification of Protocols Using Symbolic Execution, *Comput. Networks*, Vol. 2, No. 4, pp. 351-360 (1978).
3) 佐藤：COMPCON '81 Fall；第23回 IEEE 計算機学会国際会議報告，ソフトウェア工学研究会，21-1 (1981).
4) Denning, P. J.: Parts-Based Programming, COMPCON Fall, pp. 309 (1981).
5) Danthine, A. S.: Protocol Representation with Finite-State Models, *IEEE Trans. Comm.*, Vol. COM-28, No. 4, pp. 632-643 (1980).

（昭和59年2月3日受付）
（昭和59年6月19日採録）