# マイクロプログラムソースジェネレータＧＳＧＥＮ

## 4M-9 == フローチャートからマイクロプログラムソースコードへの変換手法 ==

岩島耕二、尾藤龍茂、舟山正憲、内野雅央
日本電気ソフトウェア（株）

### １．　はじめに

我々は、マイクロプログラム（以後μＰとする）用の汎用ソースジェネレータGSGEN(General Source Generator)を開発した。

近年、ソフトウェアの分野では、各種のプログラム構造図（ＳＰチャート）が提案されており、それを基にしたソースジェネレータが実現されている。しかし、μＰの分野において、このようなプログラム構造図を用いたソースジェネレータを使用するには、次のような問題がある。

① μＰの分野は、分岐制御がハードウェアに強く依存しているもの（特に水平型μＰ）があるため、ＳＰチャートを用いて構造化することは困難である。

② ＳＰチャートのソースジェネレータは、一般に特定高級言語に対応している。しかし、μＰ分野では、μＰの性能を重視するため、通常、アセンブラ言語が使用されている。また、アーキテクチャ毎にアセンブラ言語仕様が異なっているため、複数のアセンブラ言語を取り扱うソースジェネレータを開発する必要がある。

以上の問題を解決するため、ＳＰチャートの代りにフローチャートを採用した。また、ルールベース方式を採用して、フローチャートからμＰソースへの変換方法を容易に定義できる汎用ソースジェネレータ（GSGEN）を、PC-9800シリーズ上に開発した。これにより、GSGENは、特定言語に限定されない汎用ジェネレータとしてμＰ開発に多く使用されている。

本稿では、GSGENにおけるフローチャートからμＰソースコードへの変換方法を報告する。

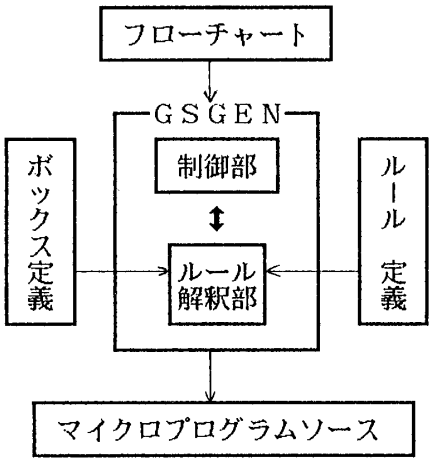



図１．GSGEN構成図

### ２．　ソース変換

GSGENは、図１で示すように、フローチャート、ボックス定義、ルール定義を入力とし、制御部でルール解釈部を制御し、フローチャートからμＰソースを発生する。

### ２．１　概要

ボックス定義は、フローチャートを構成するボックスの種類毎にその形状や属性等を定義し、ルール定義は、ボックスの種類毎にソースへの変換方法を定義する。

GSGENでは、次の手順でμＰソースを発生する。

① まず、フローチャート全体を入力し、ボックス間の結線情報等の抽出、及びボックス定義の範囲内でのチェックを行なう。

② 次に、制御部が、１ボックス毎にルール解釈部を起動する。

③ ルール解釈部は、ボックスの種類に応じたボックス定義と、フローチャート内に記述されたボックスの内容と、事前に抽出された結線情報を参照しながら、ボックスの種類毎に定義されているルール定義を解釈実行し、μＰソースを発生する。

### ２．２　ボックス定義

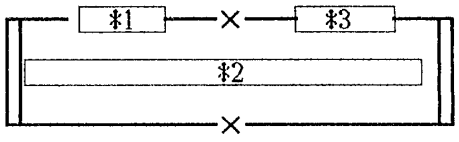
ボックス定義は、以下のものを含む。

① ボックス１個の全体定義、つまり、ボックスの名前、属性、形状など。

② その１ボックスに記述する項目の名前、属性、表示位置、入力文字数など。

③ ボックス間の接続線の入口あるいは出口となる接続点の名前、属性、位置など。

たとえば、図２のサブルーチンボックスのボックス定義では、サブルーチンボックスという事と形状を上記①で定義し、＊1～＊3の項目を②で定義し、×の２ヵ所の接続点を③で定義する。




＊1 ： セルフラベル
＊2 ： アドレス指定
＊3 ： サブルーチン先名
× ： ボックス間を結ぶ点

図２．　サブルーチンボックス

---

Microprogram source generator GSGEN
A microprogram source-code generation method from flowchart
Kohji Iwashima, Tatsushige Bitoh, Masanori Funayama, Masao Uchino
NEC Software,Ltd.

## ２．３　ルール定義

フローチャートからアセンブラ言語仕様に基づくμＰソースへの変換方法を、ルール定義言語を使用して、ボックス毎に定義する。ルール定義言語は、手続型言語であり、代入文とIF文等の実行制御文から構成されている。これらの文中には、所定のμＰソースへの変換処理を容易に記述できるように、以下のオペランドを指定することができる。

(1)　ボックス定義中の項目名

ボックスの項目名により、フローチャート上のその項目への入力テキストを直接参照することができる。

(2)　変数

作業用の数値型及び文字型変数や、ソースイメージ格納変数など予めその使用法が決められている予約変数がある。

数値型・文字型変数は、繰り返し処理に対するカウンタや、テキストの一時待避等に使用する。ソースイメージ格納変数は、ボックス内容から作成したμＰソースを格納する。１ボックスの処理終了時、この内容がファイルへ出力される。

(3)　関数

ソース発生時に必要な情報を容易に取り出せるように、次の関数を用意している。

① ボックス情報獲得関数。
② 結線情報獲得関数。
③ 文字列編集関数。

これらの関数で、前後のボックスの情報や、そのボックスと接続している線上の条件式、条件値などを取り出して、出力すべきμＰソースの内容を変更することができる。このように、自分自身のボックス以外のデータでμＰソースを発生することができるため、フローチャートの形式の自由度を高めることができる。

(4)　定数

算術計算用に数値型定数、文字列編集用に文字型定数がある。

## ２．４　ルール例

ルール定義の作成例を図３に示す。

ここでは、ボックス、項目名は、図２のボックスを使用する。本ルール例は、『サブルーチン命令（CALL）を出力し、かつ、*3が入力されていれば、アドレス指定のアセンブラ制御文を出力する。また、*2中に';'があれば、それ以降のものをコメントとする。』という動作を記述している。

#NUM0・#NUM1は数値型変数、#CHR0・#CHR1は文字型変数、#SRCはソースイメージ格納変数である。また、SCANは';'を*2から検索する関数、LENGTHは*2の長さを獲得する関数、COPYは文字列の移送をする関数、CONCATは文字列データを連結する関数である。

## ３．　GSGENの効果

GSGENは、特定言語に限定されることなく、また、ポストプロセスの種類に限定されることなく、数多くの種類の言語に対処できる。また、それらの言語の仕様変更時にも、ルール定義を変更するのみで、迅速な対処ができる。

実際に、数種類のフローチャートに対して、GSGENを適用した結果、フローチャートの種類毎に専用のソースジェネレータを作成するのに比べ、開発期間及び工数ともに1/5 ～1/10へ短縮することができた。

## ４．　今後の課題

μＰの分野においても、すべてがアセンブラレベルで開発されるのではなく、高級言語とアセンブラ言語を併用するケースが増えつつある。このような状況に対処するため、高級言語に対するＳＰチャートと、アセンブラ言語に対するフローチャートを混在処理できるシステムを検討する必要があろう。

<参考文献>

１）　森末秀雄、発田弘、他：汎用計算機におけるマイクロプログラム技術
情報処理　Vol.28 NO.12,1987

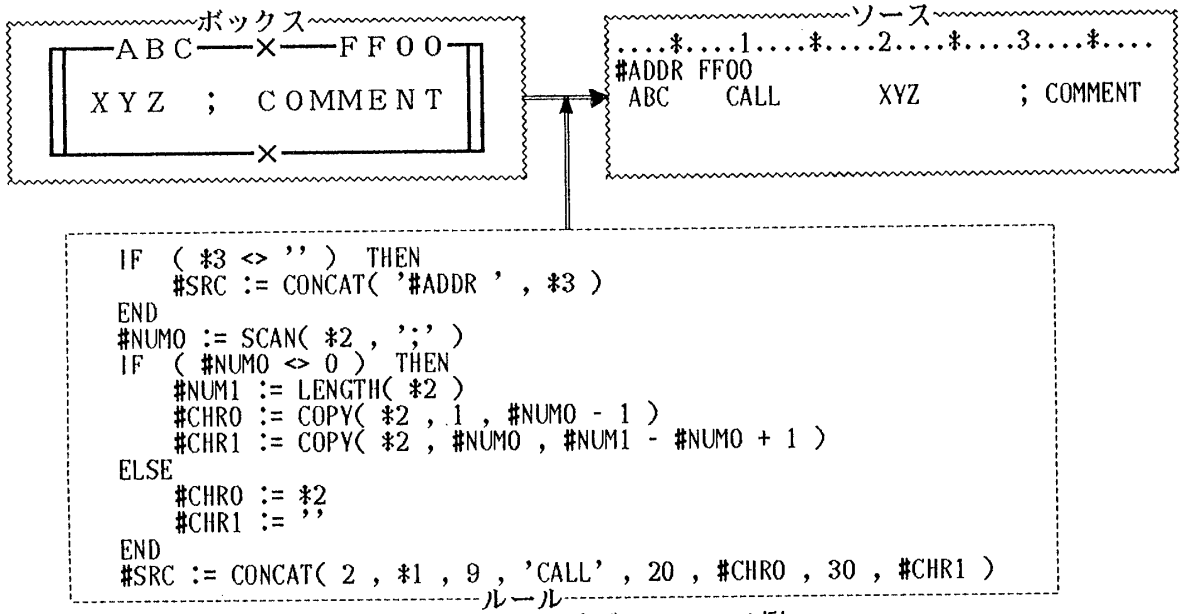



図３．　ボックス、ルール及びμＰソース例