Journal of the Faculty of Engineering, Shinshu University, No.63, 1988
信州大学工学部紀要　第63号

フレキシブルな構成が可能なシーケンサネットワーク

中村　八束*　不破　泰**　西山　隆也***

（昭和62年10月31日受理）

A Flexibly Organizable Sequencer Network

Yatsuka NAKAMURA, Yasushi FUWA,
and Takaya NISHIYAMA

A new sequencer network system is proposed which has an organic network connection effective for controlling automatic machines, etc. The system executes its sequence programming by the "Method of State Transition Diagrams," in contrast to the conventional programming which is conducted by reference to the sequence circuit diagram. This sequence programming enables the system to be provided with the functions of parallel processing and communication controlling. A specific internal language is made available which describes state transition diagrams. The structure of this language is independent of any sequencer hardwares, and accordingly, programs written in this language have such a high independence that they may be put under program library management among different sequencers. In addition, CPUs may be exchanged easily, when necessary. Each sequencer in the system is allowed to reference control programs and variables via the network, which characteristically makes the network system itself regarded as a single, large sequencer.

1. はじめに

古くは自動機などのシーケンス制御は，カムシャフトやドラムなどメカニカルなもので単純な制御を行ってきた．しかし近年のコンピュータとメカトロニクス技術の発達にともない，シーケンス制御はライン制御から集中制御へ，さらに統括制御へと大規模・複雑化してきた[1]．本論文では，それらのシーケンサを有機的に接続した新しいシーケンサ・ネットワークについて提案する．

シーケンサ制御にコンピュータを用いる利点としては，シーケンスの変更が従来の機械

* 情報工学教室　教授
** 情報工学教室　助手
*** 大学院修士課程

式のものに比べ容易であり，信頼性が高いなどが挙げられる．

本システムでは従来の記述方式とは異なり，以前我々が提案した並行処理実現の手法である状態遷移法[2]の考え方による制御を行う．このことにより，細かい制御の記述が可能であり，複数の制御を1台のコンピュータで並行に行えるなどの特徴があらわれてくる．状態遷移法によるこの記述方式は従来の記述方法による制御を完全に記述できる．さらに，複数の制御を変数を介してハンドシェイクが行えるため，複数の制御を統合したより大きな制御も記述できる．

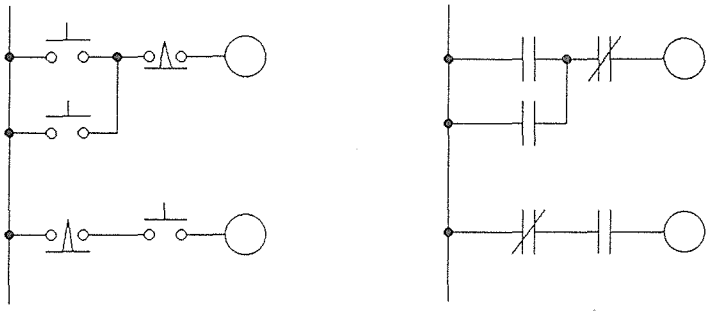
さらにこれらのシーケンサをネットワークを介して接続し，シーケンサ間で制御の手順を転送したり変数を共有する機能を持たせる．これによって，このネットワークシステム自体を大きな一つのシーケンサと見なすことができる．

2. シーケンス制御の記述法

現在最もよく利用されている記述方法は，リレー制御の回路図をもとにしたリレー回路図と呼ばれるものである[3]．この方法は，従来のリレー制御盤をコンピュータ上で実現し，その置き換えで発達してきた事によりリレー回路設計技術を持っている現場の使用者にも理解しやすいという特徴がある．

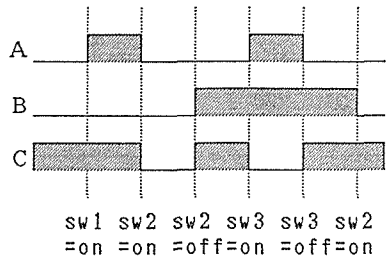
リレー制御回路は図1—aで示すようなリレーシンボルで表されるが，コンピュータ上での表現のしやすさなどから図1—bで示すラダー方式とよばれる方法で表されることも多い．ラダー方式でのプログラミングでは，画面上でのこのシーケンス回路図を入力することで制御プログラムを生成するツールも用意されている．

これとは別に，機械的なカム式コントローラから発達した方法がある[4]．これは制御を工程に分け各工程ごとの仕事と外部のスイッチやタイマなどからの信号により工程を進める条件を設けることで，各工程をあらかじめ決められた順序に従って処理するものである．この記述方法としては図2に示すように，タイムチャート記述方式（図2—a）とか，パターンテーブル記述方式（図2—b）などが挙げられる．しかしこれらの方法では外部からの入力などにより分岐したり繰り返したりという表現は難しく，表現したとしても分かりにくいものになってしまう．



（a） （b）

Fig. 1 Example relay symbols (a) and ladder symbols (b).



	input			output		
stat	1	2	3	A	B	C
1	ON			1	0	1
2		ON		0	0	1
3		ON	ON	1	1	0
4	ON		ON	1	0	1
5		ON		1	1	1

（a）　　（b）

Fig. 2　Example timing chart (a) and pattern table (b).

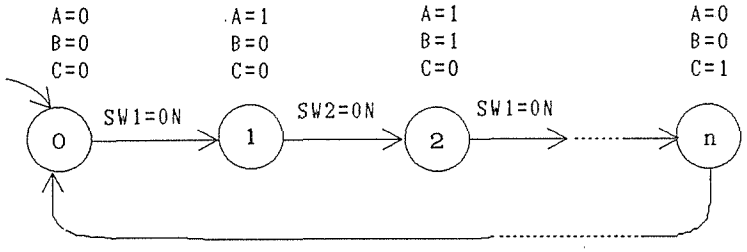


Fig. 3　A state transition diagram for state machine sequence.

そこで，「制御の各段階」を一つの状態に割り当て，その状態が入力などによってある条件になったときに次の状態に分岐せよということを，状態遷移図で記述する．この制御方法はステートマシンと呼ばれ，図3にも示すとおりタイムチャートなどで表される直線的な制御も容易に書き表すことができる．

ステートマシンではある状態内に留まっているときの出力しか記述できない．この欠点を補うためステートマシンを拡張して状態が遷移する際の CPU による処理を記述するようにしたものが状態遷移法で考える PSM (Procedural State Machine) である．同じ状態に留まるときは，もとの状態に遷移させそのときの仕事を記述することにより，ステートマシンで記述できる処理はすべて PSM で書き表すことができる．また PSM ではある状態を通過したかどうかの判定や，何回通過したかなどの変数を扱った処理を容易に記述できる．

3.　シーケンサプログラムの構成

状態遷移法の特徴には並列処理が容易に行えることが挙げられる．それには処理を各状態ごとに条件判断と仕事からなるブロックに分けた PSM を用意し，複数の PSM を各ブロックが終了するごとに順次切り替えて行う．PSM の切り替えは外部割り込みによるものとは異なり，それぞれの PSM においてブロックの処理が終了したときであり，各ブロックはそれ自体で完結された形である．そのためタイマ等外部割り込みを用いて処理を

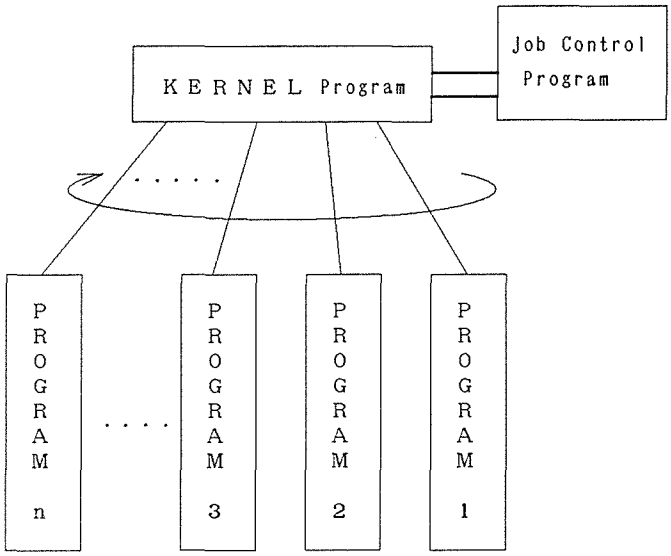


Fig. 4 A block diagram for a kernel program and sequence programs.

切り替える方法に比べ，レジスタ内容の退避や復帰などの処理が不要となり，オーバーヘッドを大幅に短縮できる．その代わり，全 PSM を一通り(1ループ)実行するのに要する時間はプログラムの処理時間によって規定されるため，一定時間以内に1ループを終了する必要がある場合は条件分岐なども考慮にいれて最長処理時間を求め，場合によってはルーチンの書き替えを要する場合や，各 PSM を処理する頻度を変える必要が起こってくる．

本システムでは処理の汎用性を重視して，この状態遷移法による制御を記述できる仮想言語（これで記述されたプログラムを「制御プログラム」と呼ぶことにする）を設計した．その制御プログラムを実行するためのカーネルプログラムは複数の制御プログラムを順次切り替えて実行する．（図4）さらにカーネルプログラムに付随した制御管理プログラムによって現在いくつの制御プログラムが動いているかを常に監視し，必要に応じて仕事の割り当てを行う．このため制御プログラムの個数は処理状況に応じて変えることができ，不要な処理を切り放すことでオーバーヘッドをさらに短縮するようになっている．

4. 制御プログラム

制御プログラムは通常のコンピュータ言語とは異なり処理の手順を記述するのではなく，状態遷移条件とその条件が成立したときに実行する仕事を並べた表の形で表されている．

図5に制御プログラムの表形式を示す．PSM は0から順番に番号がつけられている．PSM ポインタは各 PSM の番号と PSM ごとに用意するステータスポインタ表とを対応させる．ステータスポインタ表は対応する PSM の状態値と各状態ごとに用意する条件／仕事表（c／w表）とを対応させる．c／w表はこの状態からの遷移の数だけ行があり，各行は個々の遷移の条件と遷移の際に処理される仕事を記述する．この記述はカーネルプログラムが直接読んで処理する形式となっている．

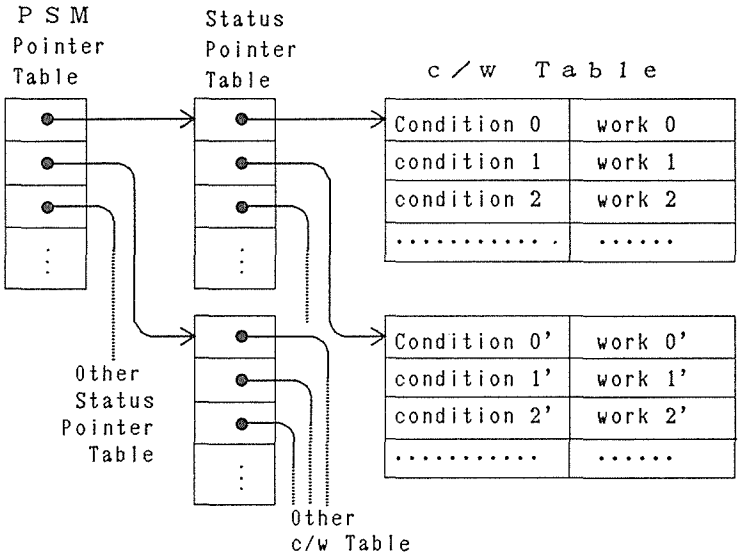


Fig. 5 Algorithm for a kernel process written in PASCAL.

```
case status of
0 : if <cond.1> then
    begin
      status := next_st.1;
      <work.1>
    end
  else
    if <cond.2> then
    begin
      status := next_st.2;
      <work.2>
    end

    else <work.3>
1 : if <cond.3> then ...
    :
    :
n : if <cond.m> then ...
end
```

Fig. 6 Process of interfacing two programs.

この表をカーネルがどの様に処理するかを PASCAL で表したものを図6に示す．
ここでは簡単のため制御プログラムは一つだけの場合について説明する．

・制御プログラムが現在どの状態にあるかを状態変数から読みだし，その処理に分岐する．（*case* 文）
・そこに記述されている *if* 文の条件式の評価を行い，条件が成立していれば *then* 以下で表された仕事をしてループの先頭に戻る．この仕事の中では必ず次に遷移する状態番号を状態変数にセットして状態の遷移を記述する．
・条件が成立しないときは，*else* 以下の *if* 文を実行する．
・最後の *else* 以下の処理には状態の遷移は記述しない．すなわち，もとの状態に遷移することを表している．

これらの条件式や仕事列の中には入出力ポートのアクセスだけではなく，変数の参照や数式を記述できる．工程歩進方式のシーケンスでは，繰り返しの回数や時間待ちなどの処理には特別な記述を行うのが普通であるが，状態遷移法を用いたこの方法ではポートへの入出力と同様に変数を取り扱えるため，それらの処理の記述を統一できる．

5. 変数の取り扱い

変数にはその制御プログラムの中でのみ有効なローカル変数と，全制御プログラムにわたって有効なグローバル変数とがあり通常はローカル変数を用い，制御プログラム間での変数の受渡しや制御の同期を取るときにはグローバル変数を用いる．

複数の制御プログラムのグローバル変数を用いた結合の方法について具体例を挙げて説明する（図7）．いま，2つの制御プログラムがありそれぞれのプログラムA，プログラムBとする．プログラムAは機械の制御を行いその中で10秒間の時間待ちが必要で，プロ

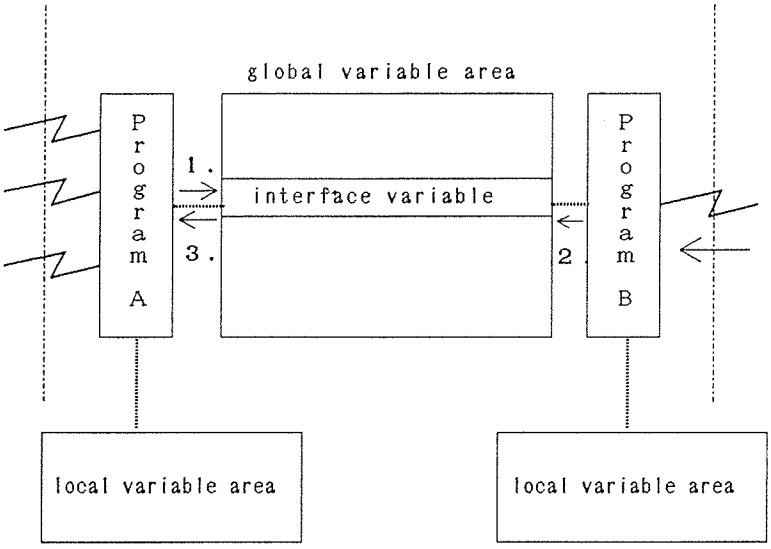


Fig. 7 A block diagram for a sequence program.

グラムBはタイマ処理を行うプログラムであるとする．ここでプログラムAはタイマ処理をプログラムBに任せる．

1. プログラムAはまずグローバル変数領域の所定の位置（インターフェイス変数）に時間待ちの秒数（10）を書き込む．
2. プログラムBはインターフェイス変数を常に監視し，値が0でなくなるとその変数を1秒間隔の外部クロック信号を基準にしてその値が0になるまでカウントダウンを行う．
3. プログラムAはその間，インターフェイス変数を監視し0になるまでその状態に留まる．

これらの操作によってインターフェイス変数に書き込まれた10という値をプログラムBが0にする間，プログラムAはその状態で10秒間の時間待ちを行う．

このようにインターフェイス変数を介して各制御プログラム間で同期を取ったり，緊急停止などの非常時に関連する処理だけを停止させることもできる．

6. シーケンサネットワークの構成

本システムではネットワーク制御機能を持った制御プログラムを各シーケンサボード上に置いて，おのおののボードを相互接続することによりネットワークシステムを構成することを考えた．ネットワークを構成する物理的手法はこの際論理的にはどの様なものであってもよい．例えば光ケーブルを利用したAnarchy，同軸ケーブルのEthernet，RS-232-C，RS422等の無手順ネット，等々処理速度の要請に応じて自由に選んでよい．このネットワーク制御機能によりネットワークを介して制御プログラムのソースを読み込んだり（ダウンロード），グローバル変数領域をボード間にわたって拡張する機能を持たせる(図8)．

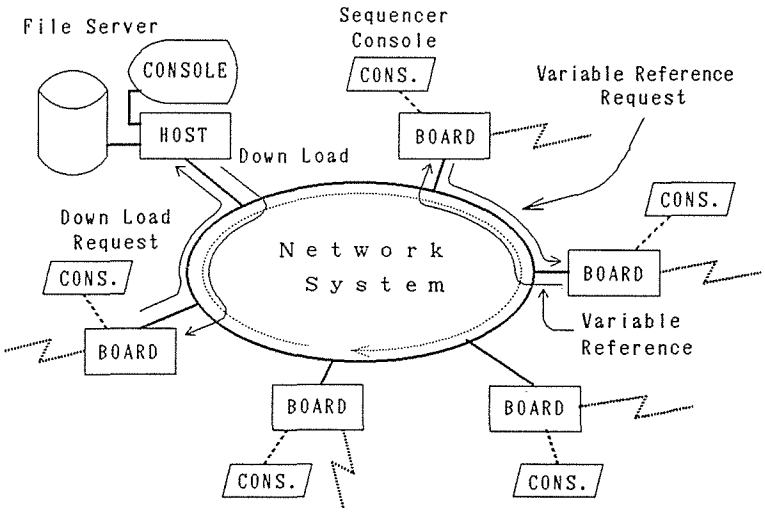


Fig. 8　A block diagram for the sequencer network.

ネットワークに接続されたシーケンサボードはそれぞれが独立しており，ホストコンピュータを介さずに直接通信が行える．従ってホストコンピュータはおもに，外部記憶装置の中にある制御プログラムをシーケンスボードからの要求に応じて転送するファイルサーバの働きと，それぞれのボードのシーケンスの進行状況の把握及びチェックを行う．

このネットワークシステムで変数の共有化は各制御プログラム，シーケンスボード間の処理の同期を取ったり緊急停止などの連絡に用いられる．逆に言えば，各ボードは原則として独立したシーケンサであり，それらのシーケンサを統合するためにボード間のグローバル変数が用意されている．

各ボードのコンソールからシーケンスの切り替え指示があったときには，そのボードから回線を通じて他のボード上に必要とする制御プログラムがあるかどうかを尋ね，あればそのボードから直接プログラムのソースを転送する．そのためにネットワーク制御機能には各ボード上に制御プログラムのディレクトリを持ち，ネットワークを介して他のボードのディレクトリとのマージや検索を行う機能を有する．

さらにその機能を利用することで，変数領域などのワークエリアだけをメモリ上に展開し，制御プログラムのソースをネットワークで参照しながら実行することも可能で，各ボード上にはワークエリア分のメモリを用意するだけで大きな制御プログラムを実行することができる．複数のボード上で同じ制御プログラムを実行するときには，メモリの使用効率が高くなり有効な手段である．

7. 入出力の整合性

入出力ポートの指定は論理ポート番号で行い，実際のポートアドレス・ビット番号への変換には変換テーブル（BIOS テーブル）を用意する．入出力は必ずその変換テーブルを参照して行うため，異なった CPU を使ったシーケンサなどに於いても制御プログラムを変更せずに実行することができ，プログラムの機械に対する独立性が高くなる．また，複数の制御プログラム間におけるインターフェイス変数の割り当てを管理する変数制御プログラムを別に用意することにより，各制御プログラムの独立性を高めて，プログラムをライブラリ管理できるようにする．

このように，プログラムの機械に対する依存性を少なくし，プログラム同士の整合性を変数制御プログラムで良くすることによって各制御プログラムに汎用性が現れてくる．

8. おわりに

ネットワーク機能を持った新しいシーケンサシステムについて述べた．このシステムでは，制御系自体は仮想言語で記述されている．このためカーネルプログラムの仕様と，通信プロトコルを合わせることによって，異なった CPU，メモリ構成，入出力ポートなどの違いを吸収することができる．必要に応じてより性能の高い CPU を使ったシーケンサや，入出力ポートやメモリを拡張したシーケンサをネットワークに容易に接続できる．このようにより柔軟性の高いシステムを構成することが可能となる．

謝　　辞

試作品の制作に関しタカノ株式会社より協力をいただいたので，ここに記して感謝の意を捧げたい．

参 考 文 献

1) H. Sugiura, K. Tomizawa, Y. Chiba, and T. Shimbo: System architecture of MICREX-F series; Fuji Electr Rev, Vol. 32, No. 1, pp. 2–8 (1986)

2) Y. Nakamura, and Y. Fuwa: A Simple Programing Method of State Transition Diagrams for Parallel Processings; Journal of Information Processing, Vol. 5, No. 3, pp. 148–154 (1982)

3) Knoop, A. R: Fundamentals of Relay Circuit Design; pp. 25–31, Reihold Publishing Corporation (1965)

4) 中島吉雄：シーケンス制御の基礎と応用；pp. 203–224, オーム社 (1977)