ソフトウェア工学 120－15
（１９９８．７．２４）

シーケンス制御プログラムのデバッグ支援システム

大西　淳、西田 洋靖

立命館大学 理工学部 情報学科
525-8577 滋賀県草津市野路東 1-1-1
e-mail: ohnishi@cs.ritsumei.ac.jp

シーケンス制御プログラムの動作を検証するための手法と手法に基づいて試作したデバッグ支援システムを紹介する。実環境でシーケンス制御プログラムを動作させる前に、デバッグ支援システムで提供する仮想環境でのシーケンス制御プログラムの動作を確認することでプログラムの不具合による誤作動を防ぎ、実機を安全に稼働させることが可能となる。試作したデバッグ支援システムによるシーケンス制御プログラムのデバッグの過程を例を用いて説明する。

A Debugging Method/System of Sequence Control Programs

Atsushi OHNISHI, Hiroyasu NISHIDA

Department of Computer Science, Ritsumeikan University
Kusatsu, Shiga 525-8577, Japan

A debugging method of sequence control programs is presented. This method provides to simulate a sequence control program in a virtual environment and verify the behaviors of the program. This method prevents from errors caused by malfunction of the program. Both a prototype system based on the method and debugging way with this system are illustrated with examples.

1 はじめに

シーケンスコントローラ用のソフトウェア(以下シーケンス制御プログラムと呼ぶ)を作成する過程で、それがプログラム開発者の意図する動作をするかどうかを検証することは困難なことであり、また実機に入力して検証することは、シーケンス制御プログラムに誤りがあると実機が暴走したり、破損する危険がある。

我々はシーケンス制御プログラム開発の支援を目標に

1. プログラミング作成支援
2. パソコン上でのプログラム動作の検証
3. シーケンサを用いたプログラム動作の検証

に関する研究を進めている。ステップ1では、ユーザがあらかじめ制御システム用に用意している部品データベースから部品を選択し、組み合わせることによって制御対象システムを定義し、さらに部品間の関係を定義することによって制御用プログラムを自動生成する。制御対象システムの定義や部品間の関係の定義は対話的にかつビジュアルにすることによって使い勝手の良いシーケンス制御プログラム開発支援環境を目指している[3, 4]。ステップ2では、生成されたプログラムの正しさを検証するために、パソコン上で制御対象システムとシーケンサの両方をシミュレートし、さらに制御用プログラムの動作をアニメーションとして表示することによって、開発者に動作確認をしてもらう。ステップ3では、シーケンサに制御用プログラムをロードし、制御対象システムだけをシミュレートすることによって、プログラムの動作を確認してもらう。

本稿ではステップ2を中心に紹介する。本研究では、ステップ1で自動生成されるプログラムの代わりに、開発者が作成したシーケンス制御プログラムを入力とし、そのプログラムで動作する実機の動きをアニメーションとして出力するようなシミュレータを開発した。

2 シーケンス制御プログラミング作成支援

最初にシーケンス制御プログラムの制御対象や制御機器、およびステップ1でのシーケンス制御プログラミング支援手法について概略を紹介する。

産業用の自動機械や装置には、いろいろな制御機器が数多く使用されている。シーケンス制御プログラムの制御対象はモータ、シリンダ、電磁弁、ソレノイド、電磁クラッチなどがあり、制御機器には押ボタンスイッチ、セレクタスイッチといった**入力用機器**、制御用リレー、タイマといった**制御・演算用機器**、電磁接触器、電磁開閉器といった**出力用機器**、リミットスイッチ、光電スイッチといった**検出用機器**、表示灯、ブザー、ベルといった**表示・警報用機器**がある[2]。

これらの制御対象・制御機器をアイコン部品としてあらかじめデータベースに登録しておき、利用者は部品を選択して組み合わせることによって制御対象システムを定義する。さらに部品の初期状態やどういった条件で何がアクティベートされるかといった部品間の関係を定義することによって、シーケンス制御プログラムを自動生成する手法を研究している[4]。

これにより、従来のラダー図によるプログラミングに比べ、より分かりやすく、また組み合わせた部品の再利用による効率化の改善が得られると考えている。

3 シーケンス制御プログラムの動作検証

シーケンス制御プログラムを作成する過程で、それが設計者の意図する動作をするかどうかの検証は困難であり、また実機に入力して検証する場合、プログラムに誤りがあると、機器が暴走したり破損する危険がある。このため、シーケンス制御プログラムを入力とし、そのプログラムで動作する実機の動きをアニメーションとして出力するシミュレータを作成することで、設計者やプログラマが実機の動作を検証し、バグを除去する。

制御対象としたシステムは加工部品が流れる2本のコンベアと、そのコンベアに架けられた橋を加工作業用のアームが行き来するもので、加工部品の位置をセンサーで、アームの位置をリミットスイッチで検出する。作業位置に到着した加工部品を検出するとコンベアが停止し、アームが移動して加工作業を行なう。加工作業が済み、未加工部品が未到着だとコンベアが再度動きだし、アームは初期位置に戻る。

シミュレータは加工部品の流れとそれに伴うアームの動きをシミュレートする。シミュレータはアームの制御内容を記述したシーケンス制御プログラムの格納されたファイルを入力とし、シミュレートして、シーケンス制御プログラムに対応したアームの動きをディスプレイ上にビジュアルな形で表現する。シミュレータの入出力の関係を図1に示す。

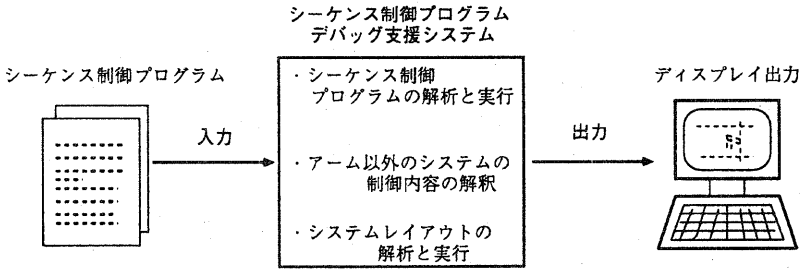

図 1: シミュレータの入出力

3.1 入力部

入力として受け取るシーケンス制御プログラムは、三菱電機(株)のMELSEC-A シリーズ[1]の「使いやすく実践的な命令」である26種の基本シーケンス命令から、接点命令、出力命令及び、プログラムエンド命令の8種類の命令を選択した。本稿での対象システムの制御は、これらの選択した命令によって実現できる。選択した命令の一覧を表1に示す。

表 1: 命令語一覧

番号	命令記号	機能
接点	LD	論理演算開始
接点	LDI	論理否定演算開始
接点	AND	論理積
接点	ANI	論理積否定
接点	OR	論理和
接点	ORI	論理和否定
出力	OUT	コイル出力
プログラムエンド	END	プログラム終了処理

命令に付随するオペランド部は図2のシーケンサ入出力接続図で示すように、シーケンサ入力部には、加工部品を加工場所で検知するセンサ"Sen1" (上コンベア),"Sen2" (下コンベア)と、アームの位置を検知するリミットスイッチ"LS0"(中央), "LS1"(上コンベア),"LS2"(下コンベア)が接続され、シーケンサ出力部にはアーム移動の際に使用するモータの回転方向(アームの移動方向"UP","DOWN")を決定するリレーが接続されているものと想定し、これらの値をとる。また、"M0" ～"M4" の内部メモリ、"T1","T2" のタイマーを使用できるものとする。よって、シーケンス制御プログラムのオペランド部は上記のもののいずれかとなる。オペランドの一覧を表2に示す。

表 2: オペランド一覧

入出力	記号	接続先
入力	Sen1	上コンベア部品検知センサ
入力	Sen2	下コンベア部品検知センサ
入力	LS1	アーム位置検知センサ（上端）
入力	LS0	アーム位置検知センサ（中央）
入力	LS2	アーム位置検知センサ（下端）
出力	UP	モーター回転リレー（上方向）
出力	DOWN	モーター回転リレー（下方向）
内部	M0	内部メモリ
内部	M1	内部メモリ
内部	M2	内部メモリ
内部	M3	内部メモリ
内部	M4	内部メモリ
内部	T1	加工時間タイマー（上コンベア）
内部	T2	加工時間タイマー（下コンベア）

3.2 出力部

シミュレータのディスプレイ出力の様子を図3に示す。図3の出力例を用いシステムのレイアウトについて説明する。画面上部と下部には加工される部品の移動用のコンベアが2本あり、部品は疑似的に箱型で表されている。また、部品は加工前は赤色、加工後は青色で示される。画面中央のコンベアに架けられた橋は加工作業用のアームが移動するためのもので、アーム

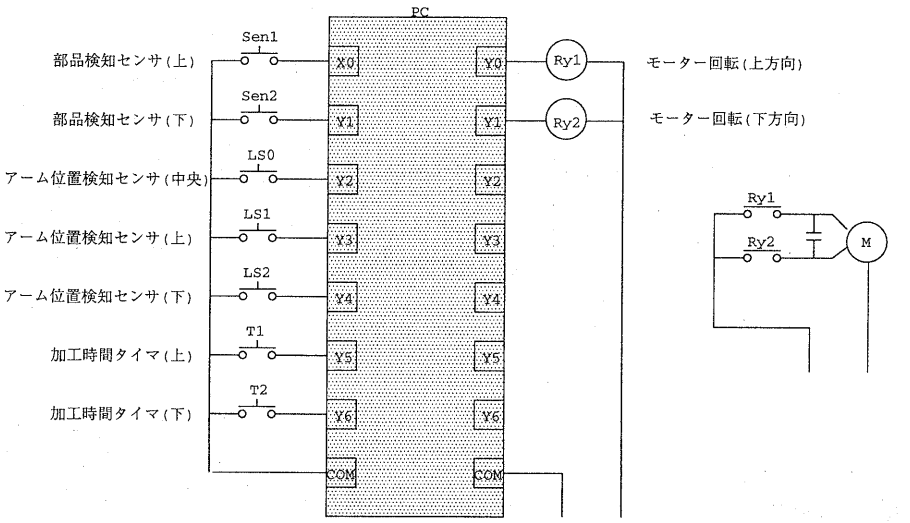

図 2: シーケンサ入出力接続図

に付けられた移動用モーターを介して上コンベアと下コンベアの間を移動できるような機構が組み込まれているものとする。この橋の左側にある信号が加工部品検知用のセンサの状態を表す。センサは加工場所で部品を検知している状態では赤色、部品の加工終了後、部品待ちの状態では青色を示す。画面左側の信号はコンベアの運転状況を知らせるもので、運転中は青色、停止中は赤色、部品の加工中でコンベアが一時停止している時は黄色を示す。画面上部のメッセージは、シミュレータとシーケンス制御プログラムの関係を示すガイド表示である。

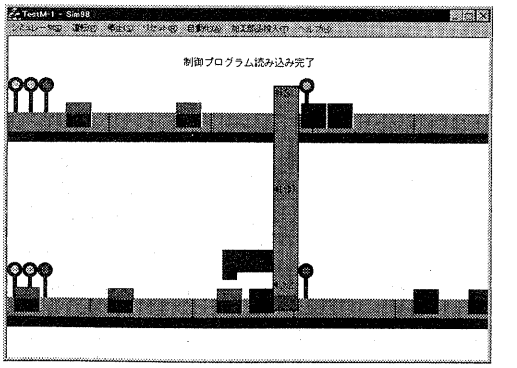

図 3: ディスプレイの出力例

4 デバッグ支援システムの実行例

本章では、3章で紹介した制御対象システムを制御するプログラムを作成する過程を想定し、段階を追って作成した5つのシーケンス制御プログラムについてシミュレートし、入力として受けとるプログラムとその出力である実行結果について説明する。プログラムのコメント欄の左矢印は新たに付け加えた行を示す。

4.1 シミュレーション1

加工部品の加工位置への到着を示すセンサ1がONならばアームを上へ移動、センサ2がONならば下へ移動するプログラムを考える。

- 入力プログラム1

```
LD    Sen1      Sen1 が ON なら
OUT   UP        上へ移動

LD    Sen2      Sen2 が ON なら
OUT   DOWN      下へ移動

END
```

図4は、下コンベアの加工部品検知センサがアームが部品を検知し、アームが下方向へ移動したが、アーム

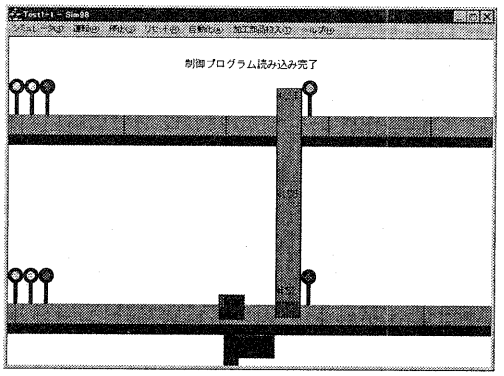

図 4: シミュレーション 1 の出力例

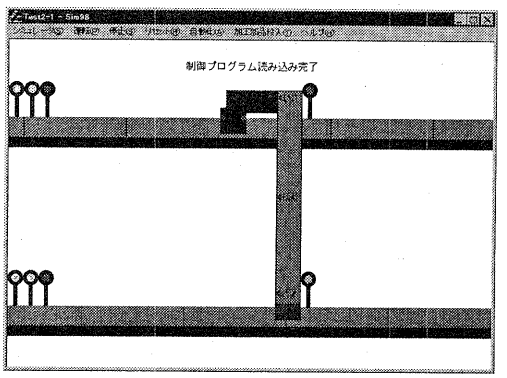

図 5: シミュレーション 2 の出力例 (1)

の動作限界を考慮しなかったため、橋の先端を越えて移動してしまい、シミュレータ上では宙に浮いているように見える状態である。これを実機に置き換えると、アームが暴走し、システムが破損するなどの状況が予想される。

4.2 シミュレーション 2

シミュレーション 1 の結果をふまえ、アームの位置検知センサを用い、加工場所で停止できるようにプログラムを改良する。

- 入力プログラム 2

```
LD    Sen1
ANI   LS1       <-LS1 の位置で停止
OUT   UP

LD    Sen2
ANI   LS2       <-LS2 の位置で停止
OUT   DOWN

END
```

図 5では、加工場所でアームが停止し、部品の加工が開始されたのが分かる。しかし、図 6の様に部品検知センサ Sen1 と Sen2 の両方が ON の時は、アームが上方向にしか移動しないというプログラマの考慮洩れによる誤動作が検出された。

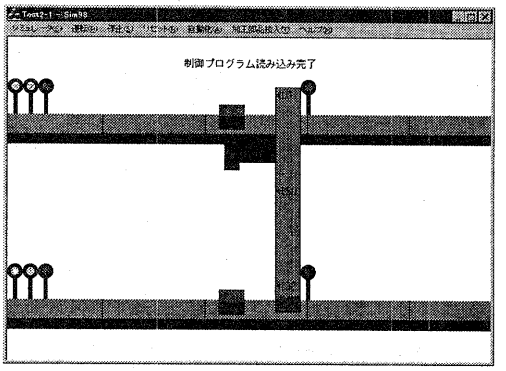

図 6: シミュレーション 2 の出力例 (2)

4.3 シミュレーション 3

シミュレーション 2 の結果をふまえ、プログラムにインタロックとセンサが両方 ON の時の指令（ここでは下移動）を加えるたものをプログラム 3 に示す。

- 入力プログラム 3

```
LD    Sen1
ANI   Sen2      <- インタロック
ANI   LS1
OUT   UP

LD    Sen2
ANI   Sen1      <- インタロック
ANI   LS2
OUT   DOWN
```

```
LD    Sen1      <-Sen1 が ON で
AND   Sen2      <-Sen2 も ON なら
ANI   LS2       <-LS2 の位置まで
OUT   DOWN      <- 下へ移動

END
```

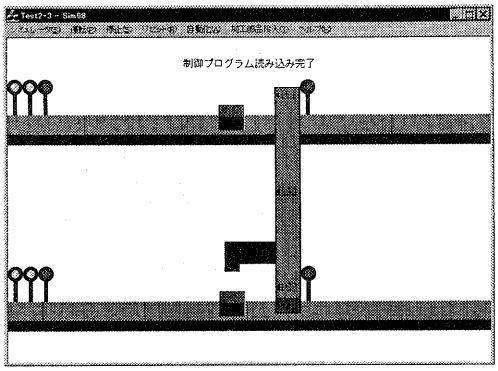

図 7: シミュレーション 3 の出力例 (1)

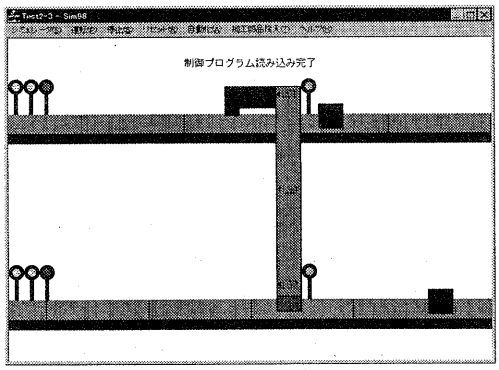

図 8: シミュレーション 3 の出力例 (2)

図 7は、加工部品検知センサが両方 ON の時、アームが下へ移動しているところである。一応のところ問題なく動作しているが、図 8の様に、加工後のアームは加工位置に留まったままなので、部品の検知センサが両方 OFF ならばアームの初期位置である橋の中央に戻るように変更する。

4.4 シミュレーション 4

センサが両方 OFF の時はアームは初期位置である中央に戻るようにする。

- 入力プログラム 4

```
LD    Sen1
ANI   Sen2
ANI   LS1
OUT   UP

LDI   Sen1      <-Sen1 が OFF でアームが
AND   LS1       <- 加工場所にあるなら
ANI   LS0       <- 中央まで
OUT   DOWN      <- 下へ移動

LD    Sen2
ANI   Sen1
ANI   LS2
OUT   DOWN

LDI   Sen2      <-Sen2 が OFF でアームが
AND   LS2       <- 加工場所にあるなら
ANI   LS0       <- 中央まで
OUT   UP        <- 上へ移動

LD    Sen1
AND   Sen2
ANI   LS2
OUT   DOWN

END
```

図 9では、プログラマの意図に反して、部品の加工後も加工場所にアームは留まっている。厳密にいえば、1 ドット戻って停止している状態である。これは、プログラムの第 2 (4) ブロックの 2 行目がアームが加工場所を離れた時点でオフになったためであり、各センサのスイッチのみをオペランドとして記述するプログラムでは、これ以上の複雑な制御内容の記述には無理があることになる。そこで内部メモリを使用する。

4.5 シミュレーション 5

シミュレーション 4 のプログラムと同様の内容を、内部メモリを使用して記述する。

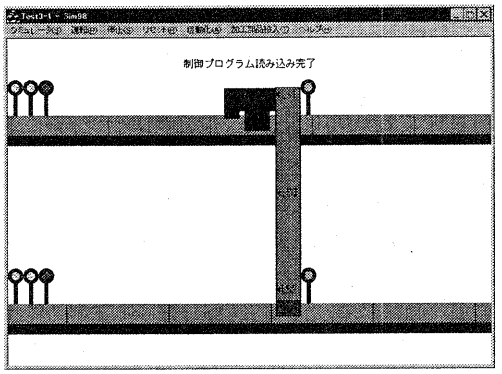

図 9: シミュレーション 4 の出力例

- **入力プログラム 5**

```
LD    Sen1      Sen1 が ON で
ANI   Sen2      Sen2 が OFF なら
OUT   M1        M1 を ON にする

LD    M1        M1 が ON でアームが
ANI   LS1       加工場所にないなら
OUT   UP        上へ移動

LDI   Sen1      Sen1 が OFF で
AND   Sen2      Sen2 が ON なら
OUT   M2        M2 を ON にする

LD    M2        M2 が ON でアームが
ANI   LS2       加工場所にないなら
OUT   DOWN      下へ移動

LDI   Sen1      Sen1 が OFF でアームが
AND   LS1       加工位置にあるなら
OUT   M3        M3 を ON にする

LD    M3        M3 を ON なら
ANI   LS0       中央まで
OUT   DOWN      下へ移動

LDI   Sen2      Sen2 が OFF でアームが
AND   LS2       加工位置にあるなら
OUT   M4        M4 を ON にする

LD    M4        M4 を ON なら
ANI   LS0       中央まで
OUT   UP        上へ移動

LD    Sen1      Sen1 が ON で
AND   Sen2      Sen2 も ON なら
OUT   M2        M2 を ON にする

END
```

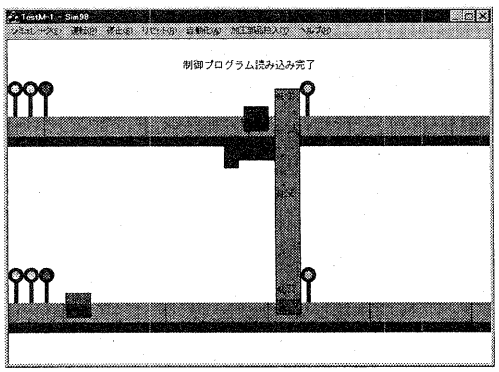

図 10: シミュレーション 5 の出力例 (1)

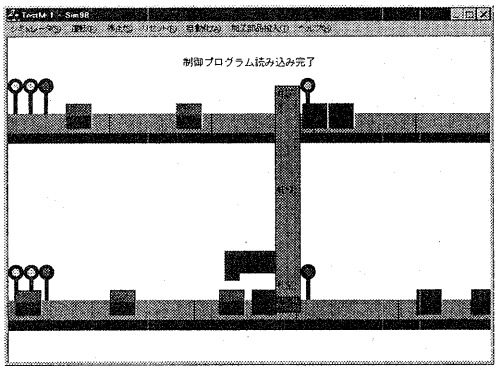

図 11: シミュレーション 5 の出力例 (2)

図 10では、加工後のアームが中央へ戻っていく様子が分かる。これで図 11の様に、不規則に流れる部品を連続的に加工するアーム制御のシーケンス制御プログラムが完成した。

4.6 デバッグ支援システムの評価

1つのプログラムを作成する過程を想定し、段階を追って作成した5つのプログラムについてシミュレートしたが、その都度シミュレータの出力でプログラムで記述した動きを確認することにより、そのプログラムの問題点や、特性を確認することができた。シミュレーション1の様なたった5行の未完のプログラムから、シミュレーション5の様に、不規則に流れる部品を連続的に加工するプログラムまで、一度に完成品を作るのではなく、基本的なプログラムから問題になったところを改善していく形で肉付けしていき、完成度の高いプログラムを容易に作成できた。このようにシーケンス制御プログラムの開発、特にデバッグに効果があることを確認した。表3にシミュレーション番号とデバッグの過程を示す。

表 3: シミュレーションの推移

番号	変更点	問題点
1	−	アームが暴走
2	LS を使用	プログラムに考慮洩れ
3	インタロックを加える	アームを中央へ戻す
4	両方 OFF の時は中央へ	プログラムの限界
5	内部メモリの使用へ	−

5 おわりに

本研究ではシーケンス制御プログラムによる対象システムの制御をビジュアルな形で提示するシミュレーション機能を備えたデバッグ支援システムを開発した。本システムは Windows95 パソコン上で Microsoft Visual C++ Ver.4.0 によって開発している。これによりシーケンス制御プログラムを作成する際に、一度に完成させるのではなく、プログラムが未完成の場合でも段階的に実行可能になるとともに、いきなり実環境でテストするのでなく、シミュレータによって動作を確認しながらプログラムを作成できるようになった。設計者の負担を軽減するとともにプログラムの不具合を容易に検出できるようになった。

今後の改良点としては、

1. シーケンス制御プログラムで用いられる命令語の拡張
2. 制御対象システムの定義

があげられる。

3.1節で述べたようにシミュレータで実行可能な命令は26種の基本的な命令の一部に制限されている。このためそれ以外の命令が含まれたプログラムは実行できない。1章で述べたようにシーケンス制御プログラムはステップ1で自動生成されるため、選択した基本的な命令だけを用いたプログラムとして生成されるようにすれば、ステップ2で実行できなくなるという問題は解消されるが、生成のしやすさや効率等を考慮の上、シミュレータで実行可能な命令を拡大したいと考えている。

また、今回は制御対象システムとして、特定のものを想定して用いているが、本来はこれもステップ1で定義されるものを入力として利用するようにしたい。

新たな制御対象システムを用いてデバッグ支援システムの機能評価を進めたいと考えている。

謝辞

本研究プロジェクトの研究メンバである立命館大学大学院理工学研究科大西研究室の博士後期課程2回生の慎ヨン日君、博士前期課程1回生の廣瀬直樹君に感謝する。

参考文献

[1] 三菱電機：「三菱汎用シーケンサ MELSEC-A(中・大規模対応型)」, 1995.

[2] 小野孝治、塩田泰仁、中野　正：「シーケンス制御技術」、産業図書、1989.

[3] 慎ヨン日、大西　淳：「ビジュアルな制御用プログラミング手法」、情報処理学会研究報告, Vol.97, No.47, SE114-9, 1997, pp.65-72.

[4] Shin, Y., Ohnishi, A.: "A Visual Method for Developing Sequence Controller Programs," Proc. 3rd World Conf. on Integrated Design & Process Technology (IDPT'98), 1998 (to appear).

[5] Valaer, L.A., Babb II, R.G.: "Choosing a user interface development tool," IEEE Software, vol.14, no.4, pp.29-39, 1997.

[6] 吉本 久泰：「プログラマブルコントローラ シーケンス制御 入門から活用へ」、東京電機大学出版局、1996