ソフトウェア工学 63－1
（１９８８．１１．２５）

セル制御のためのプログラミング環境

天明　崇　長谷川 雅樹
日本アイ・ビー・エム株式会社　東京基礎研究所

セル制御のプログランミングには、分散環境にある生産機器の制御、リアル・タイム制御、ラピッド・プロトタイピングのためのプログランミング手法が要求される。

本稿では、研究開発中のセル制御のためのプログラミング環境について報告する。まず、分散環境にある生産機器の制御のために導入したプロセス・フロー・モデルと、分散マシン・モデルについて述べる。次に、ソフトウェアの生産性を上げかつワークセルのリアル・タイム制御、ラピッド・プロトタイピングのための、ソフトウェア・ツール群について述べる。ユーザインターフェスとしては、マルチ・ウインドゥを採用し、さらにプロセス・フロー・プログランミングについては、視覚化もはかっている。

Programmning Environments for Manufacturing Cell Control

Takashi TEMMYO and Masaki HASEGAWA
I B M Research, Tokyo Research Laboratory
5-19, Sambancho, Chiyoda-ku, Tokyo 102, Japan

Cell programming is a necessary programming method for automated machine control in a distributed environment, real-time control in a manufacturing cell, and rapid prototyping.

This paper describes programming environments that we are developing for manufacturing cell control, and introduces the process flow model and the distributed machine model for control of distributed automated machines. Software tools raise the software productivity and provide programmability for real-time control and rapid prototyping. For user interface, a multi-window system is adopted and the process flow program is visualized.

１．はじめに

セル制御のプログランミングには、分散環境にある生産機器の制御、リアル・タイム制御、ラピッド・プロトタイピングのためのプログランミング手法が要求される。これは、セル制御がワークセル内の多種多様な生産機器（プログラマブル・ロジック・コントローラ（ＰＬＣ）、ロボット、搬送機器、ＮＣマシンなど）を統合的に制御するからである。今日、産業界ではＣＩＭ（ Computer Integrated Manufacturing ）の構築が重要な課題となっている。これは情報技術と制御技術を効率的に結合し、生産性を高めることを目的としている。特にワークセルは、上流の生産情報と下流の生産機器制御の接点になっており、より重要性を増している。一方、市場ニーズの多様化、製品の短命化、多品種少量生産、リードタイムの短縮化に伴い、セル制御ソフトウェアの生産性が大きな問題となっている。

制御分野では、ハードウェアにおいてはマイクロ・プロセッサ等の発達によってリアル・タイム性を重視した研究が急速に進んでいる。また、マイクロ・プロセッサを内蔵した生産機器も数多く出現し、ワークセル内で分散環境を構築している。他方、ソフトウェアにおいては個々の生産機器の貧弱なプログランミング環境があるのみで、統合されたワークセルのためのプログランミング環境はない。

本稿では、ＰＳ／５５上にワークセルのために統合されたプログランミング環境を構築したセル制御統合システム（ＭａＣＣＳ）を作成したので、その概要を報告する。これは、ワークセルの中にパーソナル・コンピュータを導入することによって、セル制御のプログラミング、デバッグ、実行監視が同一環境で行えることによる。まず、現在のワークセルの特徴とプログランミング環境ついて述べる。次に、ＭａＣＣＳの設計方針とその方針に従ったトップ・ダウン・プログラミング、ユーザ・インタフェース、ソフトウェア・ツール、環境設定について述べる。

２．現在のワークセルの特徴とプログランミング環境

ワークセルは、典型的な離散型システムであり、その中にある生産機器群を有機的に且つ効率的に活用し、製品を作成することにある。製品を作成するための制御の流れは、事象駆動型プロセスである。ワークセル内で起こるプロセスの種類は、次の二つがある。

１）連続プロセス
２）並列プロセス

さらに、ワークセル特有の複雑なプロセスとしては、生産機器間で協調動作をさせて一連の作業を行わせる、協調プロセスがある。

また、ワークセルは、生産機器構成の観点から大きく次の四つに分類することができる。

１）ＰＬＣセル
２）ロボット・セル
３）ＮＣマシン・セル
４）マン・マシン混在セル

これらのセルには、それぞれ機能的に特徴がある。まずＰＬＣセルは、プロセス・シーケンスの制御を主にするセルで、単純な生産機器群のシーケンス制御をＰＬＣが行う。ロボット・セルは、ＰＬＣセルの中にロボット群が入り、ロボットを用いた柔軟でダイナミックな制御を行うセルで、ロボット群のシーケンス制御はＰＬＣが行う。制御は、ロボットとＰＬＣの２階層で行う。ＮＣマシン・セルは、ＰＬＣセルの中にＮＣマシン群が入り、ＮＣマシン主体のセルで、ＮＣマシン間のシーケンス制御をＰＬＣが行う典型的な事象駆動型の制御が行なわれる。マン・マシン混在セルは、セルの中に生産機器群と人間が混在して作業を行っているセルで、セル全体のシーケンス制御をＰＬＣが行う。

ワークセルの特徴は、非同期性及び並列性にある。これらの特徴をとらえたプロセスのプログラミングができることが、ワークセルのプログラミング環境として重要である。

従来からのプログラミング環境では、プログラマはワークセル内にある生産機器に対して、それぞれ固有の言語でプログランミングを行っていた。生産機器からみた言語レベルは、ＰＬＣにはラダー・チャート、ロボットにはロボット言語、ＮＣマシンにはＡＰＴである。コンピュータの導入によりＣＡＤを用いたオフ・ライン・プログラミングもソフトウェアの生産性を上げるために用いられてきた手法である。しかしながら、ワークセルにおいては、生産機器及び生産機器間の調整が大切であり、オン・サイトにおけるワークセルのプログラミング環境が、ソフトウェアの生産性により重要となる。

３．ＭａＣＣＳの概要

ＭａＣＣＳは、、ＰＳ／５５上にワークセルのために統合されたプログランミング環境を構築したセル制御統合システムである。本システムでは、ワークセル内の分散環境にある生産機器群を、リアル・タイム性を重視した制御用のプログランミングが行える。また、ＣＩＭ概念を重視して、生産情報・機器制御情報等を監視及びログを取る機能を持っている。

３．１　設計方針

ワークセル内の多種多様な生産機器に対して統一したプログラミング・インターフェースを供給する。特に従

来からのプログランミング手法及びセルの生産機器固有の言語に捕われることなく、最新のワークセル環境にあったプログランミング手法を提示することである。さらに、プログラマがワークセルの制御プログラムを設計する課程に従って、プログラミングが行えるようにする。

ワークセルの制御プログラムを、プログラマが設計を手助けしながら行えるプログランミング環境を提供する。ユーザ・インターフェースの観点から対話性を重視し、操作性を考慮してマルチ・ウィンドウ及びメニュ選択を採用する。さらにワークセルの特徴である非同期性及び並列性のあるプロセス制御のプログラミングをグラフィクス機能を用いて視覚化を行う。

ソフトウェアには、Royce のウォターフォール・モデル[5]にあるようにソフトウェアの要求仕様、デザイン、コーディング、テスト、実行という課程がある。このソフトウェア作成サイクルをスムーズにすることがソフトウェアの生産性を上げることになる。ワークセルにおいては、ソフトウェアのテストと実行は、生産機器を止めて行わなければならず、その間は製品の生産ができなくなるので、ラピッド・プロトタイピングの技術が必要となる。これを行えるためのソフトウェア・ツール群を提供する。

ワークセルの分散環境にある生産機器の構成を把握し、セル制御及び生産機器制御のプログランミングができる環境を提供する。生産機器の変更等が行なわれても、プログラムの変更を最小限にするようなソフトウェア構成をもつ。さらに、個々の生産機器が持つ固有の機能に対しては、システムとしてオープン・アーキテクチャとなる。また、ワークセルでは、製品のエンジニアリング・チェンジ等が頻繁に起こることが考えられ、ソフトウェアの短命化とあいまって、ソフトウェアのモジュール化そして再利用が重要となる。このためには、ソフトウェア的に環境設定が自由に行えるプログラム構造を持たせる。

以上の点をまとめると、ＭａＣＣＳの基本設計方針は次のようになる。

1）ユーザが直感的に理解しやすく、さらに多種多様な生産機器に対して統一したプログラミング・インターフェースを供給する。
2）マルチ・ウィンドウ及びメニュ選択を用い、さらにビットマップ・デスプレイを用いた視覚化プログランミングを取り入れユーザ・インターフェスの向上させる。
3）ラピッド・プロトタイピングをするためのソフトウェア・ツール群を提供する。
4）分散環境に対応するためのソフトウェアの部品化および再利用するためのプログラム構造を持つ。

３．２　トップ・ダウン・プログラミング

トップ・ダウン・プログラミングは、本システムが採用したプログラミング手法で、プログラマがワークセルの制御プログラムを設計する課程に従って、プログラミングが行える。セル制御のトップ・レベルのプログラミングは、プロセス・シーケンスのプログラミングである。ここでは、離散型システムのプロセスの記述を行なう。この記述に、本システムでは拡張ペトリネットに基づいたプロセス・フロー・モデルを導入した。プロセス・フローの記述に拡張ペトリネットをモデルとして用いたものとしては、他に GRAFCET[9], MFC[10], C-net[11] 等がある。このプロセス・フロー・モデルでは、単位プロセスをタスクと呼ぶ。タスクの内容を記述したものが、タスク・プログラムである。セカンド・レベルのプログラミングは、タスクにおけるワークセル内の多種多様な生産機器を制御するプログラミングである。タスク・プログラムは、これらの生産機器に対して統一した生産機器言語で記述される。この生産機器統一言語に、本システムでは対象指向モデルに基づいた分散マシン・モデルを導入した。

ユーザは、プロセス・フロー・モデルに従ってプロセスを記述し、次にプロセスの中のタスクを生産機器統一言語を用いて記述する。タスクの記述は複数のマシン・モデルによつて記述することができるので複雑な作業も容易に記述が行える。これによって本システムの特徴である、二階層プログラミング（初めにプロセス・プログラミングを行ない、次にタスク・プログラミング行う）が実現される。

３．２．１　プロセス・フロー・モデル

離散型システムのプロセスは、タスクの繋りで表される。事象駆動型プロセスの特徴である非同期性および並列性は、これによって記述できる。しかしながら実際のワークセルのプロセスには、タスク間で同期をとる必要がある。このためには、タスク間の繋りに同期をとるための機構を付け加えなくてはいけない。この機構は、複数のタスクの終了を待って次のタスクを実行したり、あるタスク終了後に次の複数のタスクの実行を同時に行う機構である。この機構は、制御がタスク間を移動する内部条件となる。さらにタスクの実行状態を示す機構トークンを作ると、プロセス・フロー・モデルの基本モデルができあがる。この基本モデルは、安全で且つ活性の性質を持つ制限された（多重枝なし、自己ループなし）ペトリネットと等価と考える。

ここで、タスクの実行を開始または同期をとる内部条件に加えて外部条件を導入し、この基本モデルを拡張する。この外部条件は、内部条件がタスクの実行状態よりタスク間を制御が移動する条件なのに対して、外部から

制御の移動条件を付加するものである。外部条件は、タスクの遅延やセンサ及びサブシステムの信号、タイマー、カウンタによるタスクの実行開始などに用いられる。この外部条件を前述の基本モデルに加えたものを、プロセス・フロー・モデルとする。これを、グラフ表現したものが、プロセス・フロー・グラフとなる。

プロセス・フロー・グラフの記述法は、タスクをタスク・プレースで、タスク間を制御が移動する条件をコンディション・ゲートで表現する。タスク・プレースとコンディション・ゲートの繋りを有向アークで表現する。外部条件は、アローで表現しコンディション・ゲートに付加する。一つのコンディション・ゲートに複数の外部条件がある場合は、それらの外部条件の論理値を論理演算し、その結果の論理値を、その外部条件の代表値とする。トークンは、タスクの実行状態を表し、制御がそのタスクにあることを示す。タスク間の制御の移動は、トークンの移動として表す。プロセスの流れは、左から右に進む。プロセス・フロー・グラフの記述例を、図１に示す。

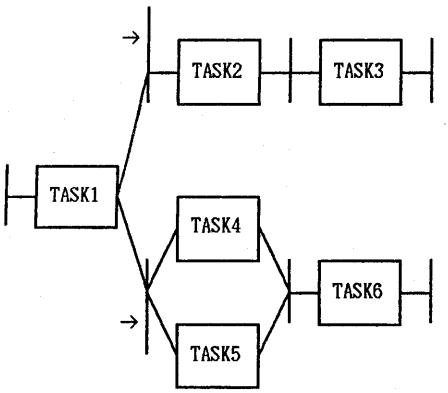

図１　プロセス・フロー・グラフ

このモデルによって記述できるプロセスの種類は、プロセスの基本構成要素である次の５つのプロセスである。

１）連続プロセス
２）並列プロセス
３）分岐プロセス
４）結合プロセス
５）環プロセス

このプロセス・フロー・モデルの基本構成要素を、図２に示す。この５種類のプロセスが記述できれば、複雑なプロセスは、これらの組合せで実現できる。またこのモデルでは、分岐プロセス、結合プロセス、環プロセスが持つ非決定性を、外部条件を用いることによって決定性モデルとして取り扱える。

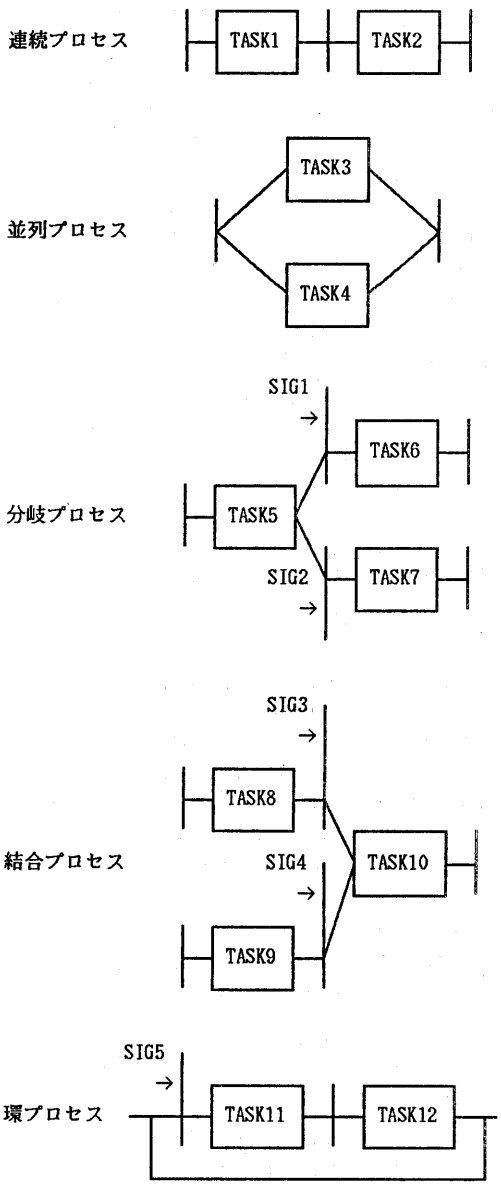

図２　プロセス・フロー・グラフの基本構成要素

３．２．２　分散マシン・モデル

ワークセル内の生産機器は色々あるが、ここでは柔軟な制御ができる生産機器であるプログラム可能な機器（ＰＬＣ、ロボット、パーソナル・コンピュータ、等）を生産機器ユニットとしてモデルの対象にしている。さらに、単純な生産機器でもＰＬＣ、パーソナル・コン

ピュータを介して制御すれば、システムとしては柔軟な制御ができるので、これらもここで取り扱うモデルの対象の生産機器ユニットしている。これらの生産機器ユニットを以後マシンと呼ぶ。ここで定義したマシンの性質は、内部状態を保持し、制御のためのコマンドが規定され、さらに通信機能を持つので、このモデルは対象指向モデルとして取り扱うことができる。図3に、分散マシン・モデルを示す。実線で丸く囲まれたものが、生産機器を表し、点線で囲まれたものが単純な生産機器とＰＬＣまたはパーソナル・コンピュータの組合せた生産機器ユニットを表している。矢印が生産機器ユニット間の通信経路を表している。通信経路は、実際に生産機器間に配線されたものが表現できる。

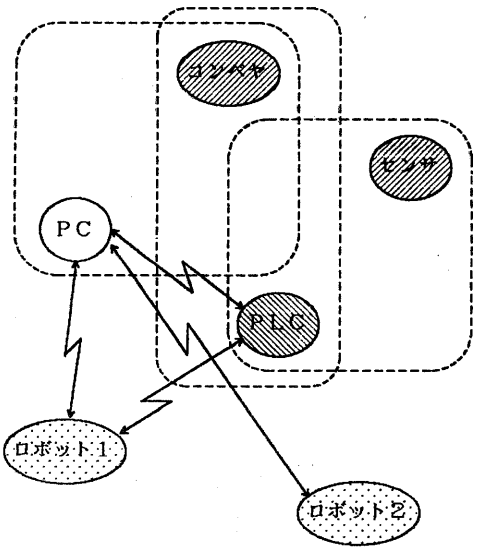

図3　分散マシン・モデル

本システムにおけるマシンの記述法は、マシンをクラス・オブジェクトと設定し、マシンの制御コマンドはインスタンス・メソッドとする。マシンに論理名が付いたときインスタンス・オブジェクトとなる。マシンに新しい拡張タイプができたり、ユーザがコマンドを拡張した場合には、インヘリタンスを用いて、新しいマシンのクラス・オブジェクトを設定できる。

このモデルによって記述できるマシンは、前述の生産機器ユニットである。またマシンに対するインスタンス・メソッドの共有化をおこない、マシンの制御コマンドの統一化をおこなっている。本システムでは、マシンの論理名と統一コマンドを組合せて生産機器統一言語と呼んでいる。

３．３　ユーザ・インタフェース

本システムでは、対話性を重視し、操作性を考慮してマルチ・ウィンドウ及びメニュ選択を採用した。マウス、キーボードによる入力、マルチ・ウインドウによる出力により、プログラミングとデバッグ等が同一環境で行なえるのが大きな特徴である。メニュ選択によって、本システムの各種のソフトウェア・ツール群にアクセスする。図3に、本システムのプログラミングの一例を示す。

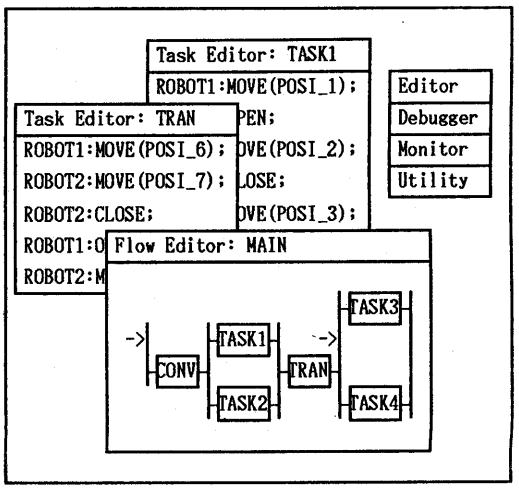

図4　プログラミング

マルチ・ウィンドウ上で稼働するソフトウェア・ツール群は、全て一様なユーザ・インタフェースを提供している。ウィンドウ操作は、基本的にマウスで行う。スクロール操作は、一般ウィンドウ・システムと同様である。ウィンドウの拡大縮小・移動、ソフトウェア・ツールが扱うデータの拡大縮小・格納等の一般的操作は、マウスでタイトル・バーを選択すれば行える。マウスは２ボタンを採用しており、ソフトウェア・ツール固有の操作については、データ領域で右ボタンを押し、データ操作については、データ領域で左ボタンを押す。

プロセス・フロー・プログラミングにおいては、プロセス・フロー・モデルを視覚化してビジュアル・プログラミングで実現している。プロセス・フロー・モデルでは、ワークセルの特徴である非同期性及び並列性のあるプロセス制御のグラフを用いて表現しており、このプログラミングをグラフィクス機能を用いて視覚化を行う。またプロセス・フロー・プログラミングをおこなっているグラフィクス・エディタからタスク・プログラミングをおこなうテキスト・エディタを呼ぶことができる。

３．４　ソフトウェア・ツール

本システムでは、ラピッド・プロトタイピングと信頼性を向上するために以下のソフトウェア・ツール群を持っている。

３．４．１　エディタ

図４は、プログラミングの一例であるが、この図中には、マルチ・ウィンドウ上にプロセス・フロー・エディタとタスク・エディタがある。

プロセス・フロー・エディタはプロセス・フローをグラフィック記号を用いて記述するグラフィックス・エディタである。プロセス・フロー・モデルに基づいたプロセス・フローではタスクの並列制御および連続制御そして同期の記述は容易である。しかしシステムとしては、タスク間の資源の共有は行なえないことがあるので、資源の相互排他制御が必要となる。例えば、あるタスクでロボットを使用していれば、他のタスクでは、そのロボットは使用できないことからも明らかである。このプロセス・フロー・エディタではタスク間の資源の使用状況を調べながらタスク内の資源の相互の排他制御のチェックを行なう機能を持っている。

タスク・エディタは、ワークセル内の多種多様な生産機器制御を生産機器統一言語で記述するテキスト・エディタである。これで、タスク・レベルにおける複数の生産機器の作業を統一的に記述できる。

環境エディタは、ワークセル内の物理的状態とプログランミングにおける論理的状態の対応を記述するテキスト・エディタである。これにより、ワークセル内の生産機器等の情報とプログランミングにおける論理的情報の対応付けを記述できる。

エディタ間のデータ移動は、スクラッチ・パッドを介して行う。

３．４．２　デバッガ／インタプリタ

デバッグ機能は、ロボット動作位置の修正、工程のタクト・タイムの表示、生産機器のコマンド・レベルでのデバッグがある。生産機器のコマンド・レベルでのデバッグおよびロボット動作位置の修正は、通信回線を通じ生産機器とパーソナル・コンピュータが対話的に制御を行っている。このとき、生産機器統一言語のコマンドは、インタプリタが生産機器固有の言語に変換を行う。しかしながら、生産機器によっては機器自身の特性により実行できないコマンドがある。このために、システムでは各生産機器の特性を記述した機器特性テーブルを持っているので、デバッガは、この機器特性テーブルを利用して、実行不可能なコマンドをチェックする機能を持っている。

また、複数のロボットを制御するために、本システムでは世界座標系をもちいているので、ロボットの位置の教示・修正を世界座標系からロボットの局所座標系に変換して行なう。つぎに工程のタクト・タイムの表示はプロセス・フローにおけるタスクのバランスの良い配置に役立つ。生産機器のコマンド・レベルでデバッグは、複数の生産機器の協調動作を作成する上で必須のデバック機能である。

３．４．３　コンパイラ／実行モニタ

ワークセルのプロセスの実行は、リアル・タイム性が要求される。このため、本システムでは、プロセス・フロー・プログラムとタスク・プログラムを環境設定ファイルを参照しながら、コンパイラが、ワークセル内の生産機器の固有の言語に変換する。稼働時には、コンパイルされた言語が、各生産機器にダウン・ロードされ実行される。実行モニタは、生産機器と通信回線を通じて、各種の情報を採りこみパーソナル・コンピュータの画面に表示する。尚且つ本システムの場合、プロセス・フロー・プログラムが、プロセス・フロー・モニタとなる。

３．４．４　システム・コマンド

本システムは、ＰＳ／５５のＪＰＣ－ＤＯＳ上で実現されており、ＪＰＣ－ＤＯＳコマンドに準じたシステム・コマンドを扱うコマンド・ハンドラがある。コマンド入出力用には、システム・ウィンドウを介して行う。また、このウィンドウには、システム・メッセージの出力用にも使用される。エディタ等の生成もこのシステム・ウィンドウ介して行うこともできる。

３．４．５　ログ・データ

制御システムのおいては、ログ・データは、重要なものであり、ユーザは、タスク・プログラムの中にログ・コマンドを書くことができる。ログ・データは、ログ・ウィンドウとディスクに書き出される。ログ・データは、稼働時のデータの集計のみでなく、デバッグ時にも有効に使用できる。また、異常時にはログ・ウィンドウが画面の中央に現れ警告、メッセージを出すコマンドもタスク・プログラムの中に記述できる。

３．５　環境設定

環境設定は、ワークセル内の分散環境にある生産機器の構成を把握し、セル制御及び生産機器制御のプログランミングするために必要となる。さらに、ソフトウェアのモジュール化そして再利用は、生産機器の変更等が行なわれた場合プログラムの変更を最小限にし、特にワークセルでは頻繁に起こる製品のエンジニアリング・チェンジ等に対しても、ソフトウェアの短命化に対しても有効である。このために、ソフトウェア的に環境設定が自

由に行えるプログラム構造を持つ。 環境設定する事項は、生産機器の論理名、生産機器の論理Ｉ／Ｏ名、共有変数さらに、特別なものとしてロボットの位置座標とその論理名がある。生産機器の論理名は、生産機器統一言語で扱う論理名で、これにより複数の同一の生産機器があっても、インタプリタやコンパイラで、それぞれの生産機器にコマンドを送ることができる。ワークセル内の生産機器のＩ／Ｏの登録は、論理名で行う。この登録Ｉ／Ｏは、プログラミングの中でＩ／Ｏ名として使用される。共有変数は、プログラミング環境全体で使用する変数である。

機器特性テーブルは、本システムがオープン・アーキテクチャになっているため個々の生産機器が持つ固有の機能の追加に対して記述するテーブルである。また、生産機器統一言語のタスク・プログラムを統一的に記述できるが、しかし生産機器によっては、機器自身の特性により実行できないコマンドもある。このときにも、本システムでは機器特性テーブルにその生産機器の特性を記述する。

生産機器統一言語から、生産機器固有の言語への変換は、一度中間言語のおとし、そこからポスト。プロセッサを用いて生産機器固有の言語への変換している。ポスト・プロセッサは、ライブラリィとして各生産機器ごとにあり、インタプリタやコンパイラが引いてきて使う。

４．実施例

本システムを、二台のロボットと一基のベルト・コンベーヤと一つのビジョン・サブシステムと一台のＰＬＣで構成される実験室のロボット・セルに適応した。このセルでは、三種類の製品が流れている。ここでは、ビジョン・サブシステムのプログラムは、サブシステム内でプログラミングされており、それらのプログラムがタスクに対応してプロセス・フロー・プログラミングが行なわれている。

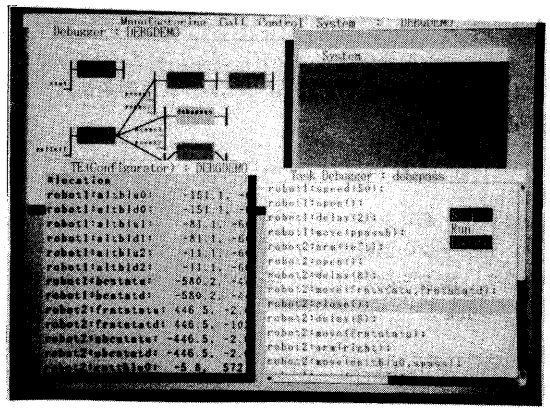

図５　デバッグ環境

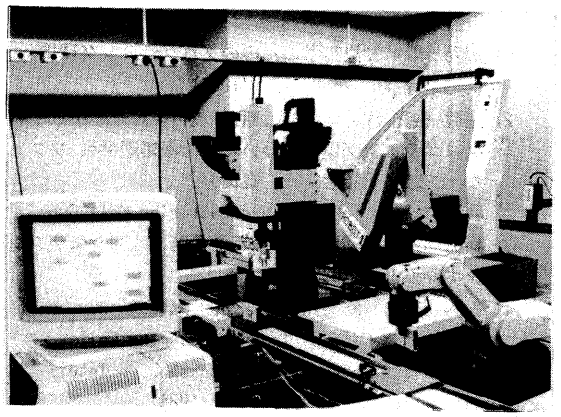

図６　セル制御の実行及び監視

本システムを、工場の中で実際に適応した例は、三台のロボットと二台の位置決め装置と二基のベルト・コンベーヤと一台のＰＬＣで構成されるロボット・セルに適応した。このセルでは、四種類の製品が流れている。セルのプロセス制御という観点からみれば、ＰＬＣによるインターロック制御である。この適応例においては、従来のプログランミング手法に比べて、本システムを用いる事によりプログランミング時間が三部分の一になった。

５．おわりに

本稿では、分散環境にある生産機器の制御、リアル・タイム制御、ラピッド・プロトタイピング等を行えるプログランミング環境を構築したＭａＣＣＳについて報告した。

本システムは、セル制御のプログラミング、デバッグ、実行監視が同一環境で行う。分散環境にある生産機器の制御のためにプロセス・フロー・モデルと、分散マシン・モデルを導入し、トップ・ダウン・プログラミングを実現した。ユーザインターフェスとしてマルチ・ウインドゥを採用し、その上にリアル・タイム制御、ラピッド・プロトタイピングをするための、ソフトウェア・ツール群を開発した。分散環境の設定を行え且つソフトウェアの部品化および再利用するためのプログラミング構造に作成した。

今後の課題としては、グローバルな環境設定をローカルにタスク及び外部信号レベルまで行えるようにし、ソフトウェアの部品化及び再利用を進めてゆく。

謝辞

本研究を進めるにあたり、ご討論ご協力いただきました日本アイ・ビー・エム（株）東京基礎研究所松家英雄氏に深く感謝いたします。

参考文献

1) 奥乃 博 : 知識工学のためのプログラミング環境, 情報処理 Vol.26 No.12, (1985).
2) 猪股 俊光,片野田 守人,西村 義行：Ｃｏｓｍｏｓのプログラミング環境について, 情報処理学会 ソフトウェア工学研究会60-2, (1988).
3) 長谷川 雅樹,天明 崇：Manufacturing Cell Control System, 計測自動制御学会第１４回システムシンポジウム, (1988).
4) Temmyo, T., Hasegawa, M. and Matsuka H. : Disrubuted Control in Manufacturing Cells, Proc. IEEE International Workshop on Intelligent Robots and Systems, (1988).
5) Royce, W. : Managing the development of large software systems:Concepts and techniques, Proc. IEEE International Conference on Software Engineering, (1987).
6) Andrew, G.R. and Schneider, F.B.: Concepts and Notations for Concurrent Programming, ACM Comput. Surv., Vol.15 No.1, (1983).
7) Peterson, J.L. : Petri Net Theory and the Modeling of Systems, Prentice-Hall Inc., (1981). 市川 惇信,小林 重信 訳, ペトリネット入門 ー情報システムのモデル化ー,共立出版, (1984).
8) Agerwala, T.: Putting Petri Nets to Work, IEEE Computer, Vol.12 No.12, (1979).
9) Le Group de travail Systèmes logiques de l'AFCET : Pour une représentation normalisée du cahier des charges d'un automatisme logique, Automatique et Informatique Industrielles, No.61, (1977).
10) 長谷川 健介,高橋 宏治,増田 良介,大野 秀嶺：非連続システム制御のためのマーク流れ線図の提案, 計測自動制御学会論文集, Vol.20 No.2, (1984).
11) 村田 智洋,薦田 憲久,松本 邦顕：ペトリネットに基づく高フレキシブルＦＡ制御システム,計測自動制御学会論文集, Vol.20 No.9, (1984).
12) Luqi and Berzins, V. : Rapidly Prototyping Real-Time Systems, IEEE Software Vol.5 No.5 (1988).
13) Stefik, M. and Bobrow, D. : Object-Oriented Programming: Themes and Varations, The AI magazine Vol.6 No.4, (1986).
14) 塚本 享治：ロボット言語に必要なシステム制御機能と抽象化機能, 日本ロボット学会誌, Vol.2 No.2, (1984).

付録　プロセス・フロー・グラフの定義

プロセス・フロー・グラフG_{PF}は、

$$G_{PF} = (T, C, I_T, O_T, \mu, E)$$

ここで、

$T = \{T_1, T_2, \ldots, T_M\}$ M>0 タスク・プレースの集合

$C = \{C_1, C_2, \ldots, C_L\}$ L>1 コンディション・ゲイトの集合

$T \cap C = \phi$ タスク・プレースの集合とコンディション・ゲイトの集合とは互いに素

$I_T : T \rightarrow C$ タスク・プレースからコンディション・ゲイトへの入力関数

$O_T : C \rightarrow T$ コンディション・ゲイトからタスク・プレースへの出力関数

$\mu : T \rightarrow \{0, 1\}$ タスク・プレースのマーキング関数

$E = \{E_1, \ldots, E_K\}$ K≧0 外部条件の集合

任意の時刻tにおける、任意のコンディション・ゲイトC_kに関するタスク・プレースT_IからT_Oへのトークンの移動条件式は、

$$\#_{(T_I, T_O)}(t) = \prod_{i=1}^{m1} \mu(T_i(t)) \wedge \overline{\prod_{o=1}^{m2} \mu(T_o(t))} \wedge E(t)$$

但し、m1>0,m2>0

となる。時刻ｔ＋１における新しいマーキングは、タスク・プレースT_Iでは、

$$\mu(T_i(t+1)) = \mu(T_i(t)) - \#_{(T_I, T_O)}(t),\ i=1 \ldots m1$$

タスク・プレースT_Oでは、

$$\mu(T_o(t+1)) = \mu(T_o(t)) + \#_{(T_I, T_O)}(t),\ o=1 \ldots m2$$

となる.

トークンの発生は、コンディション・ゲイトC_kにおいて、入力関数I_Tがない場合であり、移動条件式の右辺の項を、

$$\prod_{i=1}^{m1} \mu(T_i(t)) = 1$$

としたときである。

トークンの消滅は、コンディション・ゲイトC_kにおいて、出力関数O_Tがない場合であり、移動条件式の右辺の項を、

$$\overline{\prod_{o=1}^{m2} \mu(T_o(t))} = 1$$

としたときである。