マイコンによる環境適応制御のためのプロセス管理システム（第１報）

戸羽　篤也，多田　達実，鈴木　耕裕
中西　洋介，高橋　裕之

An Experiment of Process Monitor System for Circumstantial Control By Micro-Computer（Part I）

Atsuya TOBA,Tatsumi TADA,Takahiro SUZUKI
Yosuke NAKANISHI,Hiroyuki TAKAHASHI

抄　録

機械の自動化を考える上で，その制御系の設計は，機構設計とともにその装置の機能あるいはコストを決定する要素の一つである。自動化装置の機構あるいは動作の制御で，周囲の状況に応じて動きを変化させるという機能の付与を要求されるケースも増えている。これを実現するためには，機構動作を制御するのと並行して周囲状況を検出するなど，複雑なプロセスを実行しなければならない。本課題は，マイクロコンピュータを利用してこうした制御を行うための，低コストで開発効率のよい制御プログラム開発環境の開発を試みたものである。

１．はじめに

近年，工業試験場あるいは 当部に寄せられる自動化装置の開発および試作に関する技術支援の内容を見ると，作業品質，処理速度，コストパフォーマンスなどのような，品質向上面の要求水準が高度化しつつある。

最近では，汎用の制御用デバイス製品が市場に提供され，価格も安くなっているようだが，製品のコストパフォーマンスを考えた場合，やはりカスタマイズされた制御系の採用を検討せざるを得ないであろう。この種の制御を，しかも低コストで実現しようとすれば，安価なマイクロコンピュータシステムを使用するのが最も適当であろう。しかし，マイクロコンピュータを動作させるためのプログラムの記述，動作の確認，プログラム修正などといった手続きには手間を要し，開発効率を重要視する場合には敬遠されがちな作業となる。

本報は，機械および装置に柔軟な動作と機能を付与するために，マイクロコンピュータを利用した動作制御を採用する場合を想定し，プログラム開発時の効率を向上させることを目的に試作した基本ソフトウェアおよび，システム開発環境について報告する。

２．環境適応制御について

近年，自動機械あるいは自動化装置は，その機能面において高度化の一途をたどっている。すなわち，機械といっても，単純に機構の動作を制御するだけでなく，センサを備えて常にある量を監視したり，情報を他の装置とやり取りするような機能を持つものが一般化するようになった。

環境適応制御とは，制御理論の分野において，周囲の環境が変化しても，機械動作の応答性が変わらないように制御則を自動的に変化させる制御手法である。例えば，一自由度系の制御に PID 方式のフィードバック制御を採用する場合，そのアクチェータに接続された機構の慣性要素，すなわち部材重量や荷重に変化があった場合，その変化に応じて，PID 各項のパラメータを変化させるのである。

しかし，本報でいう環境適応制御は，その意味合いを広く解釈し，応答性といった動作品質に関する議論を目的とするのではなく，動作自体の即応性に注目して環境適応性を考えた。つまり，周囲環境の認識と機構駆動の制御を同時並行して実行することによって，例えば，障害物を避けながら進行させたり，一定の間隔を空けながら軌道を倣うなどといった技術に応用するための手法の確立を目的とする。

3．マイコン制御の利用とその課題

機械動作の制御で，それほどの敏速性を必要とせず，逐次状態を変えながら動作が進める機械では，その開発効率の観点から，制御要素がモジュール化され，開発環境も整っているプログラマブルコントローラなどを使用して制御系を組み立てるのが一般的であり，この手法は現在でも有効な手段として多く採用されている。そうしたモジュールの中には，フィードバック制御方式を採用して，高精度な位置決め制度が可能なものも存在する。しかし，この手法の問題点として，各モジュールの動作を逐次実行させるシーケンシャル・プロセッシング方式をとることから，突発的な外乱因子に対して，即座に対応することが難しいことと，各制御モジュールは，製品化される段階で汎用性が高められ，要求する機能の割に装置コストが高くついてしまうという点が，解決されるべき課題であろう。

これに対して，マイクロプロセッサを利用したディジタル制御では，動作制御の高速性，柔軟性に加えて、割り込み処理プロセスを利用した突発要因への対応が比較的簡単にプログラムでき，機構の動作を周囲の環境変化に対応させながら動作に変化を付与することも可能である。しかし，マイクロコンピュータのプログラム開発は，未だに開発環境が整備されておらず，事例ごとに専門技術者が一つ一つプログラムコードを作成しているのが現状である。

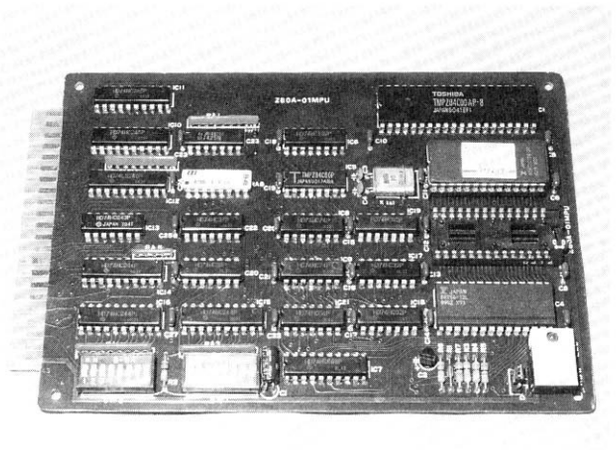
本試験では，中小の企業が製品開発の初期段階，あるいは工場内で独自に使用するコントロール機器の製作で使用することを想定し，安価で使用実績が多いという点で，Zilog 社製 8 ビット語長マイクロプロセッサ（Z 80 型，以下 Z80-MPU と略記する）を採用した。同プロセッサは，その価格の割に機能が高く，洗練されたアーキテクチャにより回路設計も比較的簡単であるゆえに，現在でも，試作装置の開発段階でしばしば使用される。同プロセッサの詳細な仕様は，多くの解説書が出ているので，そちらを参照してもらいたい。[1),2)]

4．試験用ハードウェアの設計と試作

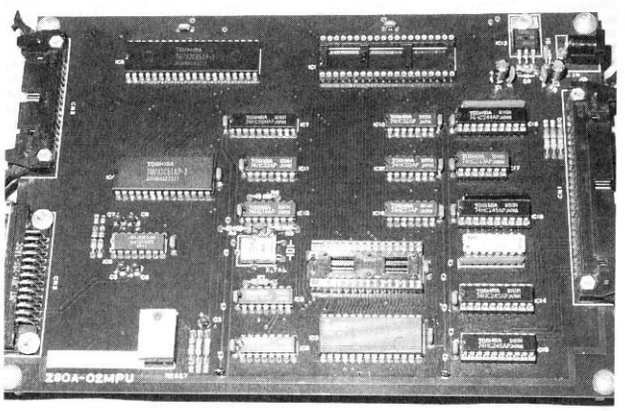
4.1　CPU 回路基板

試験で使用したマイコンシステムのハードウェアに関して簡単に紹介する。試験に用いたマイクロプロセッサユニット基板（CPU 基板）は，当部で独自に回路設計したものを使用した。その外観を図 1 の写真で示す。

図 1 － a）の CPU 基板（Model:01 と表記する）は，CPU とメモリが搭載され，メモリバンク切替え可能なメモリデコード回路を有する。図 1 － b）の CPU 基板（Model:02 と表記する）は，試験用に後で作成したもので，メモリデコード回路は単純にして，若干の入出力インタフェース用 LSI を搭載している。（以下，これら CPU 基板を総称して「MC－Z 80」と呼ぶことにする）



a）CPU 基板（Model MC-Z 80 MPU 01）



b）CPU 基板（Model MC-Z 80 MPU 02）

図 1　試験で使用した CPU 回路基板の外観

MC－Z 80 に搭載したマイクロプロセッサは，Zilog 社製 Z 80 シリーズの中から，最大動作クロックが 4MHz の Z 80 A－CPU を採用した。Z 80－MPU は，メモリアドレス空間の大きさが，64Kbytes（以下，Kbytes＝KB と略記する）であり，MC－Z 80 は，このメモリ空間を，図 2 に示すようなメモリ配置に設定した。全メモリ空間のうち，メモリアドレスの下位半分を ROM 空間（32KB），上位半分を RAM 空間（32 KB）とし，さらにそれぞれの領域を半分ずつシステムとユーザアプリケーションで分け合う。

システム ROM 領域には，ATMS/Z 80 システム（詳細は後述）を構成するシステム BIOS ルーチン，ATMS カーネル，シェル（リモートモニタプログラム）が配置される。また，システム RAM 領域には，システムが動作するのに必要な入出力アダプタエントリ，PCB，タスク管理データブロックなどと，拡張メモリを取り扱うためのページフレームウィンドウとして使用する。

MC－Z 80 の Model：01 には 8KB の ROM ソケットを 2 つと，32KB の RAM ソケットを 1 つ用意した。このメモリ

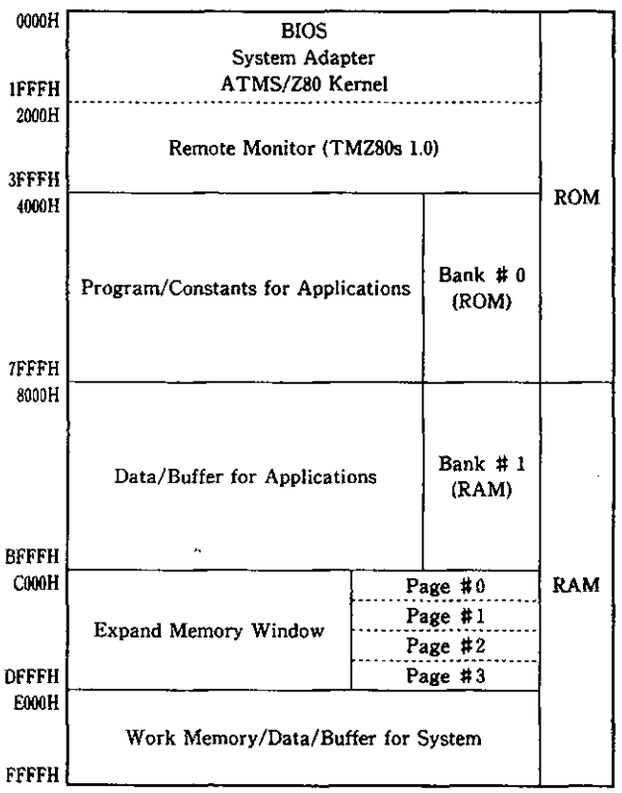


図２　MC-Z 80 のメモリ配置

空間の一部は，バンク切替えによって論理的に切り離すことが可能で，周辺の回路を工夫すると，64KB を超える容量のメモリを使用することができる。

同基板には，様々な動作環境設定をスイッチの切り替えで行えるよう，環境設定用の 16bit 分の DIP スイッチを用意した。また，入出力回路などの拡張性を考慮して，専用のピン配列を持つ外部信号バスを設けた。各端子は，周辺回路の拡張に配慮し，電力増幅用の信号ドライバ IC を介して専用のバスコネクタと接続される。

4.2　テスト用周辺基板

試験を行う際，適宜必要に応じて MC－Z 80 と接続して実験に供するための，入出力用回路プリント基板をいくつか設計し，試作した。MC－Z 80 と他の試験用入出力回路基板等とを接続するために，MC－Z 80 の外部信号接続コネクタのピン配列に合わせて専用のバスコネクタ基板を用意した。これにより，CPU と他の周辺入出力を簡単に組み合わせ，目的のハードウェアシステムを柔軟に構築できるようになる。基板どうしを連結するバスコネクタには，Model：01 用には，4 mm ピッチ，44 端子のコネクタ，Model：02 用には 50 端子のフラットケーブル用基板コネクタを使用した。図３にバスコ

表１　システムバスコネクタ

部品面				半田面			
Pin No.	信号名	方向	機能	Pin No.	信号名	方向	機能
1	A 0	O	アドレスバス	2	A 1	O	アドレスバス
3	A 2	O	アドレスバス	4	A 3	O	アドレスバス
5	A 4	O	アドレスバス	6	A 5	O	アドレスバス
7	A 6	O	アドレスバス	8	A 7	O	アドレスバス
9	A 8	O	アドレスバス	10	A 9	O	アドレスバス
11	A10	O	アドレスバス	12	A11	O	アドレスバス
13	A12	O	アドレスバス	14	A13	O	アドレスバス
15	A14	O	アドレスバス	16	A15	O	アドレスバス
17	D 0	I/O	データバス	18	D 1	I/O	データバス
19	D 2	I/O	データバス	20	D 3	I/O	データバス
21	D 4	I/O	データバス	22	D 5	I/O	データバス
23	D 6	I/O	データバス	24	D 7	I/O	データバス
25	/MREQ	O	メモリの選択	26	/IORQ	O	I/O の選択
27	/RD	O	読み出し許可	28	/WR	O	書き込み許可
29	/INT	I	割込み要求	30	/NMI	I	強制割込み要求
31	/BUSREQ	I	バス切離し要求	32	/BUSAK	O	バス要求認識
33	/M 1	O	マシンサイクル１	34	/RFSH	O	メモリリフレッシュ
35	/RESET	I	リセット	36	CLOCK	I	基準動作クロック
37	/WAIT	I	アクセス待ち要求	38	/HALT	O	CPU 動作停止
39	IEI	I	割込み許可入力	40	IEO	O	割込み許可出力
41	Vdd		＋12V	42	Vee		－12V
43	Vcc		＋５V（電源）	44	GND		０V（接地）

注：表中の「方向」は，CPU 側から見て，I は入力，O は出力信号であることを示す。また，信号名の前にスラッシュが付記されている信号ラインは，

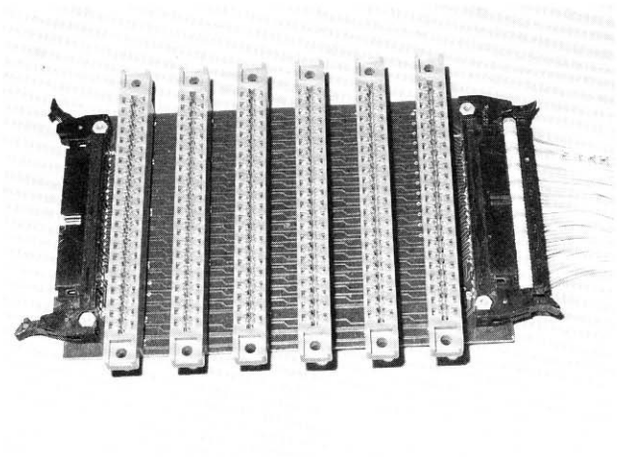


図3　回路構成用バスコネクタ基板の外観

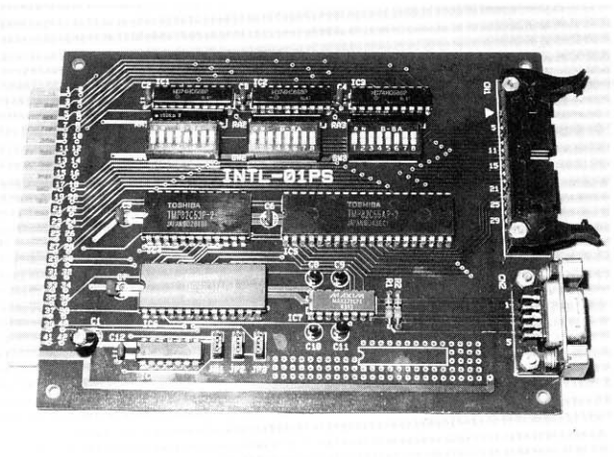


図5　テスト用スイッチ基板の外観

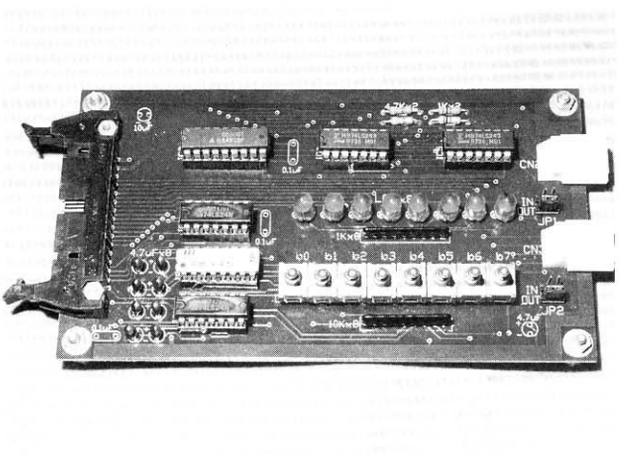


図4　試験用標準入出力インタフェース基板の外観

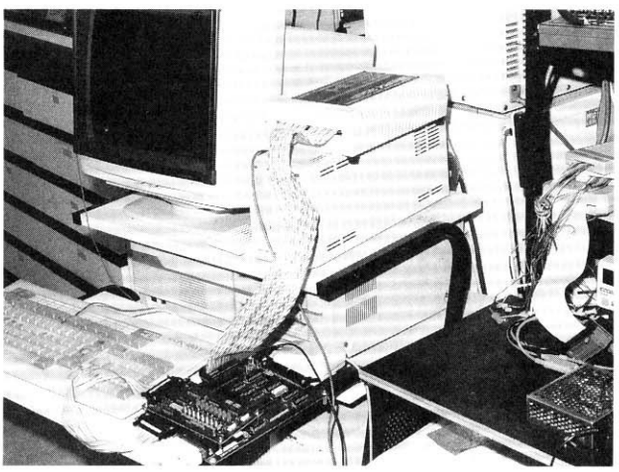


図6　システム開発で使用したICEの外観

ネクタ基板の外観写真を示す。一枚のバスコネクタ基板には，6つの回路基板が接続可能で，割込み処理の優先度を制御する信号端子である39番と40番以外は，端子どうしが並列に配線接続される。バスコネクタの各端子配列は，表1に示すとおりである。(50P基盤コネクタの場合も1～44番を使用する)

また，図4の写真は，標準的な入出力試験用に作成した回路基板である。この基板上には，シリアルデータ転送，パラレルデータ転送，タイマ/カウンタ用のLSIが必要に応じて実装でき，先のCPU基板の標準的な入出力用に使用する。特に，シリアルデータ通信用LSI（インテルi 8251-USART）は，後述のモニタプログラムを介して，ホストコンピュータ（パーソナルコンピュータ）と接続して，プログラムの転送やデバッグ作業に利用する。

図5の写真は，簡単なビット列入出力試験用の基板である。基板には，8ビットのディジタルトグルスイッチ（入力用）と，8ビットのLED（出力用）の他，4ビットずつ入出力方向の設定が可能な基板用端子が2つ配置されている。この基板は，標準入出力基板のパラレル入出力LSI(インテルi 8255－PPI）のインタフェースコネクタに接続される。

4.3　システム開発環境

本試験で，ハードウェアの動作試験およびソフトウェアのデバッグ作業は，図6の写真に示すZAX社製インサーキットエミュレータ（ICE；ERX－308P型）を使用した。エミュレータのコントロールは，NEC製パーソナルコンピュータ（PC－9801VX型；CPU：i 80286Clock：10MHz）で行なった。また，Z 80－MPUの機械語コードを作成するツール類として，マイクロテックリサーチ社製Z 80－MPU用クロスマクロアセンブラ（MRI ASM 80 v6.0），同社製Z80－MPU用クロス　Cコンパイラ（MRI MCC v3.0），および同社製オブジェクトリンカ，ライブラリアン等を使用した。[3][4] アセンブラおよびC言語プログラムソースの編集は，ライフボート社製マルチスクリーンテキストエディタ（RED2 V2.15）を使用した。

5．基本ソフトウェアの設計と試作

5.1　ATMS/Z 80の位置づけと役割

本試験の目的は，周囲の環境の変化に応じて機構の動作を適切な状態に保つような，複雑なマイコン制御プログラムを

効率よく開発する手法について検討することである。機械制御を目的としているものではないが，これと似た目的で，パーソナルコンピュータのプログラム開発やプログラムの実行を効率よく行うための，オペレーティングシステム（OS；Operating System）と呼ばれる基本ソフトが使用される。パソコン用のOSは，主に外部磁気記憶装置とのデータ転送を中心にしたプログラムモジュールの集まりと，アプリケーション開発に必要なソフトウェアツール類を用意している。

今回のマイコン用基本ソフトウェアシステムの試作・開発では，パソコン用OSとして広く普及しているMS－DOSの設計思想を参考にした。すなわち，BIOSと呼ばれる共通プログラムモジュールと，それを利用するためのプログラム呼出し手順，入出力装置のハードウェアと基幹フロープログラムとを補間する入出力アダプタおよびデバイスドライバ，基本的な入出力手順を規格化した標準入出力などの考え方を参考にしながら，マイコンに搭載し，機械の制御を想定した基本ソフトウェアシステムを設計し，試作した。[5)6)]

さらにプログラムコードのモジュール化を進めるため，ハードウェアによるタイマ割り込みを利用した複数プロセスの時分割式並列実行処理の仕組みについても考察した。以下，この基本ソフトウェアシステムを，高機能時分割プロセス管理システムという意味合いで，本報においては〝ATMS/Z80〟(Advanced Time-Sheering Monitor System for Z80-MPUの略称）と呼ぶことにする。

図7は，ATMS/Z80を用いてプログラムを作成するまでの手続きを示したものである。ATMS/Z80は，MC－Z80仕様の差異を吸収し，開発された機械制御アプリケーションを円滑に実行するための環境を提供する。また，プログラム開発段階を想定し，ホストと接続してプログラムの転送やデバッグを行なうためのモニタプログラム（Remote Monitor; TMZ 80 s 1.0)を付加し，MS－DOSのShellのような役割を担う。これらは，メモリ空間のROM領域に常駐する。

目的の機構制御および周囲環境検出プログラムは，テキストエディタでプログラムソースが記述され，アセンブラあるいはコンパイラによってリロケータブルオブジェクトが生成される。生成されるオブジェクトには，プログラムと入出力用アダプタとの２つのケースがあるが，詳細な説明は後述する。これらオブジェクトと，予めオブジェクトの形で用意されている入出力アダプタやC言語関数ライブラリなどとリンクされて実行コードが生成され，RAMあるいはROMにロードして実行される。その後，デバッグ作業を経て，実働プログラムが完成されていくのである。

本試験の目的は，冒頭にも述べたように，マイコン制御用プログラムの開発効率の向上である。そのために検討すべき課題として，1）プログラムのモジュール化，2）共通プログラムの利用手続き，3）高級言語とのインタフェース等が挙げられる。ATMS/Z80システムの設計では，これらの課題を解決するべく，その構成を1）メモリ配置，割込みシーケンス管理，アダプタ入出力手続き，デバッグ環境等の基礎的なプログラムをROMに常駐させる，2）入出力手続きを

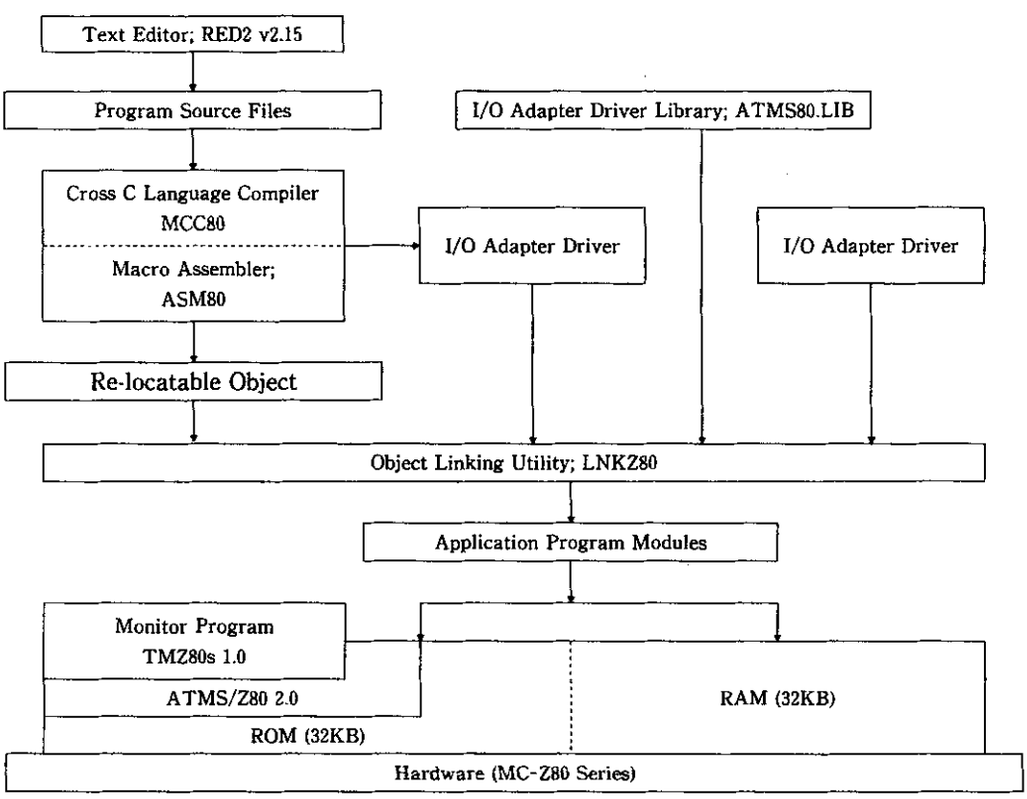
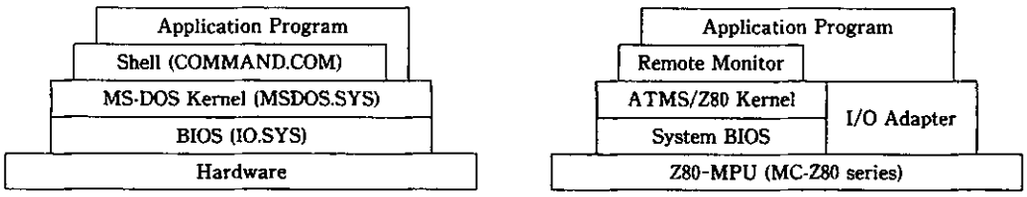


図7　プログラムモジュールの開発手順



a）MS-DOS の例　　b）ATMS/Z80 のシステム構成

図 8　ATMS/Z 80 システムの構成

統一し，一部のプログラムの交換や変更が全体のプログラムに及ぼす影響を最小限にする，3）作成済みのプログラムモジュールは部品化し，再利用可能となるように工夫する，等に配慮した。基本的には，図 8 に示すように，MS－DOS の構成をヒントに，ATMS/Z80 を BIOS（Basic Input/Output System），入出力アダプタ，システムカーネル，C 言語入出力用関数ライブラリ等で構成することとした。[7)]

5.2　入出力アダプタ

5.2.1　BIOS と入出力アダプタ

ハードウェアを制御するプログラムを部品化する場合，できるだけデータの入出力手続きが共通化されているのが望ましい。これを実現するためには，入出力装置が異なっても，発行した入出力要求に対して，そのハードウェアの差異を感じないようにその要求に応える仕組みが必要である。

ATMS/Z80 では，入出力装置のハードウェアと入出力要求に応えるためのプログラムを合わせて，「入出力アダプタ」として取り扱う。特に，そのプログラム部分は，「アダプタドライバ」と呼び，システムに予め ROM で常駐させるか，プログラムとともにメモリにロードされる。

アダプタドライバが提供するサービスは，いつでも，どこからでも一定の手続きで呼び出しが可能でなければならない。この種の機能を実現する手法として BIOS と呼ばれる考え方がある。BIOS は，ハードウェア等の詳細を知らなくてもシステム側が用意するサービス機能を引き出して利用できるように工夫されたプログラムの集合で，「ソフトウエア割込み」と呼ばれる手続きを利用してサービスが呼び出されるのが一般的である。

ATMS/Z80 の入出力アダプタには，具体的なデータの入出力を行なうもののほか，割込み処理の設定，入出力デバイスの管理，メモリ資源の管理プロセスの管理および並行処理など，システム管理と密接に関わる部分のサービスを提供するものもアダプタの一つとして考え，特に「システムアダプタ」と呼ぶことにした。ATMS/Z80 は，これら 2 種類のアダプタドライバを，原則として「システムアダプタ」は ROM 領域に常駐させ，「入出力アダプタ」を必要に応じてプログラムにリンクしてメモリにロードすることを想定した。いずれの場合も，Z 80－MPU のソフトウエア割込みの処理手続きを利用して，BIOS サービスとして利用する。

5.2.2　入出力アダプタの構造

ATMS/Z80 の入出力アダプタは，基本ソフトウェアの本質的な部分である「ATMS カーネル」と，入出力デバイスとの間で，I/O ポートに対する入出力手続きを標準化する役割を担う。アダプタドライバは，そのための橋渡しをするもので，ATMS カーネルに対するデータの受け渡しを行うための独特な構造を持つ。

アダプタドライバは，ハードウェアと密接な関係があることから，原則として入出力装置の開発に併せて用意する。アダプタドライバの入出力ルーチンは，直接 ATMS カーネルから呼び出されるため，その記述は，アダプタドライバの構造や書式に則っていなければならない。しかし，この決まりを守っている限り，入出力装置はアダプタドライバとともに部品化され，別なケースでの再利用も容易になる。

ATMS/Z80 の入出力アダプタは，データの受け渡しを行うための最低限度の機能をもったプログラムである。アダプタドライバは，図 9 に示すように，アダプタ ID，インストールブロック，コーリングシーケンスブロック，プロシジャブロックの 4 つの部分で構成される。

アダプタ ID は，システムが入出力アダプタを認識するための 16 バイトのデータで，8 文字以内のアダプタ名と，1 バイトの属性データが含まれる。属性データは，アダプタの入出力属性を定義する情報で，1 バイトの入出力および複数バイトデータ列の入出力に対して，許可や禁止を定義することで，ATMS/Z80 の標準入出力手続きで誤った操作や無駄なプロセスの実行を省く目的で参照される。

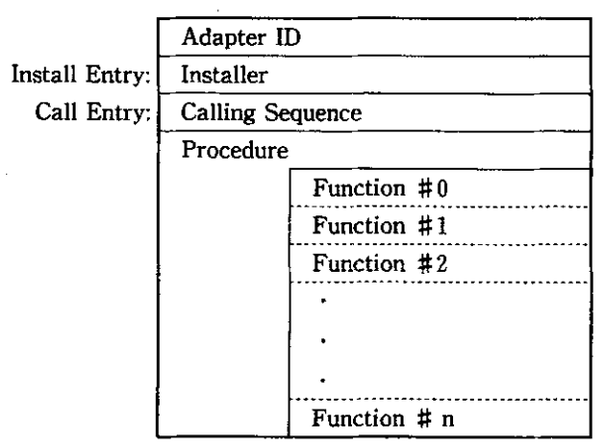


図 9　入出力アダプタの構造

表２　入出力アダプタのファンクション

番号	機能ID	処　理　内　容
0	Init	入出力デバイスの初期化.
1	Setup	入出力デバイスの機能設定.
2	Reset	入出力デバイスの再設定.
3	Name	アダプタ名の取得.
4	Read	データの入力.
5	Write	データの出力.
6	Stat	アダプタステータスの取得.
7	Cond	入出力条件の設定.
8	RBcnt	入力バッファの検索.
9	WBcnt	出力バッファの検索.
10	RBflsh	入力バッファクリア.
11	WBflsh	出力バッファクリア.

インストーラブロックは，アプリケーションプログラムが入出力アダプタを認識し，ATMS の管理下におくために，システム起動の最初の段階で呼び出されるプログラムが記述される。メモリにロードされた際，このブロックの先頭のメモリアドレスを「インストール・エントリ・ポイント」と呼ぶ。このプログラムの先頭には，決められた書式のパブリックシンボルが割り当てられ，このエントリポイントのメモリアドレスは，オブジェクトをリンクする段階で，プログラムに認識される。インストールエントリのパブリックシンボル名は，「ADVRnn」とし，「nn」にはアダプタ番号として 00 H～3 FH の中から 16 進 2 桁を現わす ASCII コードで埋める。インストーラブロックに記述するプログラムは，最低限アダプタの後述のコールエントリポイントをシステム側に認識させる必要がある。

コールシーケンスブロックは，ATMS カーネルからのデータ入出力要求を解釈し，目的に応じたルーチンへ分岐するプログラムを記述する。メモリにロードされた際のこのブロックの先頭アドレスを「コール・エントリ・ポイント」と呼び，このアドレスは，前述のインストーラにより，ATMS/Z80 の入出力システムテーブルに登録され，ATMS カーネルは，このテーブルを参照してアダプタへの入出力要求を発行する。アダプタドライバのプロシジャブロックは，データの入出力のほか，入出力装置の初期化，設定を行うためのプログラムを記述する。ATMS/Z80 は，表３に示すファンクションを用意し，アダプタドライバでは必要に応じてそのいくつかについて処理プログラムを記述する。アダプタがサポートしないファンクションには，エラーリターン処理を記述する。

5.2.3　アダプタのインストール

ATM/Z80 の入出力アダプタは，独自に記述してリロケータブルオブジェクトとして用意するか，ライブラリに登録されているものを，アプリケーションのリンク時に組み込むか，どちらかの方法でインストールされる。MC－Z80 は，リセット後の起動の段階でアダプタを初期化するが，その時組み込まれたアダプタのインストールエントリを順次呼び出し，正常に終了コードを返したアダプタのコールエントリアドレスを，ATMS/Z80 標準入出力用コールベクタテーブルに登録する。ATMS カーネルは，以降，このテーブルを参照して接続された入出力装置とのデータ受け渡しを行なう。

5.2.4　システムアダプタ

ATMS/Z80 は，主にハードウェアの違いを吸収するとともに，モニタ機能などの最低限の入出力を提供し，またはシステムの補助的な役割を担う入出力アダプタをいくつか用意する。これらは，データ入出力用のアダプタドライバとは性質が異なるため，「システムアダプタ」と呼んで区別する。

ATMS/Z80 は，表４に示すようなシステムアダプタを用意し，MC－Z80 の仕様に応じて，システムプログラムの一部として常駐 BIOS の形でインストールする。

これらのうち，「KBD」，「DSP」，「COM」，「LPT」のアダプタ名を持つアダプタドライバは，ATMS カーネルの標準入出力をサポートするもので，内容は入出力アダプタと同じものである。特に，COM アダプタは，リモートモニタで使用するシリアル通信(RS-232c)をサポートする。この他，「INT」，「DEBUG」，「BMS」，「EMMZ80」等のアダプタ名を持つアダプタドライバは，システムのハードウェア管理に直接関与する。これらのシステムアダプタで提供するファンクションは，

表３　ATMS/Z 80　2.0 のシステムアダプタ

識別子	外部参照シンボル	定義ファイル	アダプタ名	管　理　対　象
INT	ADVR00	INTRPT. ADP	INT	割り込み処理環境
DBG	ADVR01	DEBUG. ADP	DEBUG	シングルステップ処理
BMS	ADVR02	BMS. ADP	BMS	バンク切り替えメモリ
EMM	ADVR03	EMM. ADP	EMMZ80	拡張メモリ
KBD	ADVR04	CSL. ADP	KBD	キーボード
DSP	ADVR05		DSP	LED ディスプレイ
COM	ADVR06	COM. ADP	COM	シリアル通信
LPT	ADVR07	LPT. ADP	LPT	ラインプリンタ

表4　ATMS/Z80のソフトウェア割り込み

割込みレベル	オペコード	分岐先	処　理　の　内　容
IRQ #0	RST 00H (C7H)	0000H	システムの初期化
IRQ #1	RST 08H (CFH)	0008H	システムプログラムのリスタート
IRQ #2	RST 10H (D7H)	0010H	ファンクションコール
IRQ #3	RST 18H (DFH)	0018H	BIOS コール
IRQ #4	RST 20H (E7H)	0020H	未定義
IRQ #5	RST 28H (EFH)	0028H	未定義
IRQ #6	RST 30H (F7H)	0030H	ブレイクポインタ処理
IRQ #7	RST 38H (FFH)	0038H	タイマ割込み

表3で示したものとは全く意味が異なっており，アプリケーションプログラムから直接呼び出して使用することは想定しない。

5.2.5　BIOS サービスの呼出し

入出力アダプタは，「BIOS コール」という手続きにしたがって呼び出すが，いつでも，どこからでも呼び出し可能な手続き手段として，Z80－MPU がもつ「ソフトウェア割込み」の機能を利用する。

Z80－MPU のソフトウェア割込みは，表5に示すように8つのレベルまで利用することができる。このうち2つをシステムの初期化等の処理に，2つを BIOS 等のシステムサービスのために予約する。BIOS コールによる入出力アダプタの利用は，割込み要求レベル3を使用して行う。詳細な呼び出し手続きの説明は省略する。

5.3　ATMS/Z80 カーネル

ATMS カーネルは，システム BIOS および入出力アダプタの上位に位置し，ATMS/Z80 の本質的な管理と処理を行う。それらの機能は，「ファンクションコール」という呼出し手続きによって利用するが，これには BIOS と同様にソフトウェア割込み機能（IRQ レベル2）を使用する。

ATMS カーネルは，最もハードウェアに近い BIOS モジュールを統合管理し，シェル（リモートモニタ）や高級言語との中継ぎ的役割を担う。ATMS カーネルは，データ入出力手続きである「標準入出力」の管理，メモリ資源やプロセスの実行に関わる管理などを行う。特に，プロセス管理では，時間分割型のマルチタスク並列処理を実現するための仕組みについても検討した。

5.3.1　標準入出力とポートハンドル

標準入出力は，ATMS カーネルが，その機能を十分発揮するように入出力アダプタを管理し，アプリケーションプログラムからのデータ入出力要求に応えるシステムサービスである。標準入出力は，図10のような構造をもつデータブロック（PCB;Port Control Block）によって管理する。これにより，

オフセット	情報の内容
+0	I/O ポートアクセスフラグ
+1	入出力アダプタ番号
+2	アダプタ番号（8文字）
+3	
+4	
+5	
+6	
+7	
+8	
+9	
+10	入出力バッファポインタ
+11	
+12	（未使用）
+13	
+14	
+15	

図10　PCB のデータ構成

例えば複数タスクで1つの入出力アダプタを共有するような場合でも，タスクごとにそのアダプタとの入出力が可能かどうか確認しながらデータの受け渡しすることができるようになる。ATMS カーネルは，PCB をいくつか用意し，アプリケーションの要求に応じて，逐次 PCB と入出力アダプタとを関連付ける。この際，PCB はアプリケーションに対して使用する PCB の管理番号を返し，以降アプリケーションは標準入出力を使用する場合，その番号を使って入出力を行なう。この時の管理番号を，特に「ポートハンドル」と呼ぶ。これらの関係を整理して図11に示す。

5.3.2　メモリ管理

ATMS/Z80 は，MCB（Memory Control Block）という管理用データブロックを使用してメモリ管理を行う。MCB は，図12に示すように10バイトの大きさをもつ。MCB の最初の2バイトは，その MCB が管理する利用可能なメモリの先頭アドレスを格納する。具体的には MCB の先頭アドレス

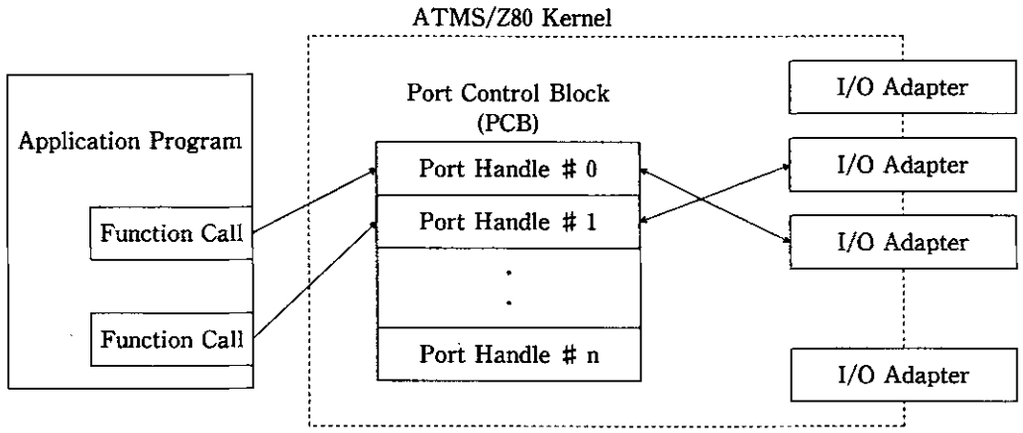


図 11　ポートハンドルと PCB

オフセット	情報の内容
+0	メモリ先頭アドレス
+1	
+2	メモリ制御フラグ
+3	メモリバンク番号
+4	メモリブロックのサイズ
+5	
+6	直前のＭＣＢポインタ
+7	
+8	直後のＭＣＢポインタ
+9	

図 12　MCB のデータ構造

に MCB の大きさ（10 bytes）を加えた値となる。ATMS カーネルは，MCB の先頭アドレスとこの 2 バイトのデータを比較して，MCB の情報が有効かどうかを判断する。MCB の第 3 バイトは，MCB が管理するメモリブロックに関する属性および用途を示す 1 バイトの管理情報である。

また，MCB どうしは「直前ポインタ」と「次後ポインタ」とによるチェイン情報で連結される。ATMS カーネルは，MCB チェインを検索し，要求される容量のメモリブロックが割当ておよび解放を行う。メモリ管理情報および，メモリ割当ての説明は割愛する。

5.3.3　時分割タスク並列処理

ATMS/Z80 は，機械制御および周辺環境検出用アプリケーションプログラムを単純化する手段として，いくつかのプログラム単位（以下,「タスク」という）を同時並行で実行させる仕組みについても検討を試みた。複数プログラムの並列実行は，技術的には，短い時間でハードウェア割込みを発生させ，そのタイミングごとにタスクの実行を切り替えることで実現する「時分割」による並列処理を採用した。

ATMS/Z80 で，複数の並列実行させるべきタスクの選定は，「タスクスケジューラ」と呼ばれる特殊なプログラムが行なう。ATMS/Z80 では，8 つのレベルのタスクプライオリティ（優先度レベル）を設け，タスクをいずれかの優先レベルに配置する。また，各々のタスクには，1）実行，2）調整，3）待機，4）休止の 4 つの動作レベルを定義し，柔軟なプロセスの実行管理を行う。

タスクスケジューラは，タスクの実行を，図 13 に示すようなタスク管理メモリブロック（TCB；Task Control Block）の情報に基づいて管理する。タスクスケジューラは，時間割込みが発生するたびに，現在実行していたタスクの状態を記憶し，図 14 のように優先レベルの最も高い Priority ＃ 0 に設定されたタスクから，TCB の情報を検査していき，次に実行すべきタスクを選定する。ATMS/Z80 は，最も優先度の低いレベル（Priority ＃ 7）に，モニタプログラムをインストールし，他のレベルに実行すべきタスクがインストールされていない場合は，少なくともモニタが起動することになる。

オフセット	情報の内容
+0	タスク制御フラグ
+1	待時間カウンタ
+2	
+3	
+4	SP レジスタ
+5	
+6	バンク番号（Code）
+7	バンク番号（Data）
+8	PC の初期値
+9	
+10	SP の初期値
+11	
+12	バンク初期番号（Code）
+13	バンク初期番号（Data）
+14	（未使用）
+15	

図 13　TCB のデータ構造

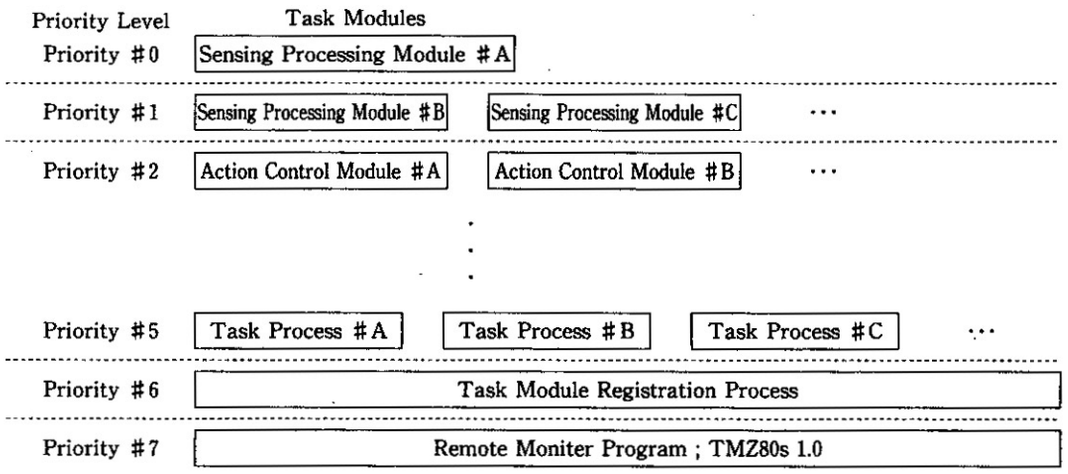


図14　ATMSカーネルのタスク管理

タスクの実行は，基本的にスケジューラが管理するが，必要に応じてタスクスケジューラを介さずに任意の優先レベルのタスクを起動する手段も用意する。これは，優先レベルの高いタスクが一定の処理を終えて，1つ低位あるいは特定の優先レベルに実行権を渡す場合などに使用する。

もし，優先レベルの高いタスクが無限に繰返す処理を実行すると，それより低い優先レベルのタスクは，全く実行されなくなる。このような事態を回避するために，優先レベルの高いタスクは，急を要する一定の処理を終えた後は，低位の優先レベルへ実行権を渡すようにしなければならない。また，あるタスクで入出力を行った結果，その情報を直ちに別なタスクに知らせる場合，直接特定のタスクへ実行権を渡さなければならないことも考えられる。

こうした状況に対応するため，ATMSカーネルは，タスクの起動あるいは終了をファンクションコールのサービスとして提供する。

5.3.4　ATMS/Z80のシステム起動

ATMS/Z80システムは，電源の投入，Reset信号によるリブート，IRQ＃0のRST命令のいずれかでシステム再起動が要求されると，システムワーク領域メモリの内容を検査し，データが不確定状態で電源投入直後と判断される時は，システムアダプタ，メモリ管理テーブル，I/Oポート，タスク管理テーブルなどを初期化する。システムワーク領域のデータが有効であると判断される時は，それらの情報をそのまま引き続いて使用する。

これは，何かの理由による誤動作でプログラムカウンタが0000H番地を指すとき，それまで記憶していたタスク管理情報をクリアしてしまうのを避けるための措置である。ワークメモリ領域の検査が終了すると，システムはタスクスケジューラに制御を渡す。電源投入後の初期状態において，タスク管理レジスタには最も低いプライオリティレベルにモニタプログラムのみが登録されており，MC－Z80を起動後の最初の初期化状態でタスクスケジューラが起動されると，モニタプログラムが起動する。

5.4　リモートモニタ

ATMS/Z80は，Shellに位置づけられるモニタプログラムとしてリモートモニタ（TMZ80s）を用意する。TMZ80sは，システムが用意する標準入出力の一つである「STDAUX」（Port handle＃2）を使用し，ホストコンピュータ（パーソナルコンピュータなど）とシリアルコミュニケーション（RS－232c）を介してデータを交換しながらメモリ内容とZ80－MPUのレジスタの参照および変更を行なう小規模のプログラムである。リモートモニタは，BIOS，システムアダプタ，ATMSカーネル等とともにPriority＃7のタスクとしてシステムROMの一部に書き込まれる。

5.5　C言語インタフェース

5.5.1　C言語によるプログラム開発

アプリケーション開発に高級言語を用いた場合，アセンブラレベルの関数呼出しなどで高級言語とのインタフェースをとるために，生成されるコードが幾分大きくなり，処理の際にもその分の処理時間（「オーバヘッド」ともいう）を費やすため，プログラムの実行効率は悪くなるが，プログラムの移植性，ドキュメント性などに優れていることから，開発効率の点で有利である。[8)]

マイクロコンピュータによるデジタル制御には，ビット単位の操作など，柔軟なコーディングが必要であることから，C言語が使用されることが多い。ATMS/Z80は，Z80－MPU用クロスCコンパイラに，Microtech Research社製MCC80　3.0を使ってアプリケーション開発を行なえるように工夫した。このコンパイラは，ANSI C言語仕様に準じたソーステキストをコンパイルすることができるツールである。

5.5.2　MRI MCC80 3.0 のC言語インタフェース

C言語では，プログラムモジュールを「関数」という単位で取り扱われる。表5に基本的なANSI C言語仕様の関数定義の要素を示す。関数の定義で重要なのは，関数名，引数，戻り値などの扱い方である。これらの扱いがアセンブラレベルで具体的にどう展開されるかという決めごとが「言語インタフェース」である。

MCC80　3.0では，C言語関数名は，アセンブラコードレベルで，関数名の先頭にアンダスコア（下線；'_'）を付けて外部参照シンボルとして定義される。また，引数は，HL，DE，BCレジスタペアを優先的に使用し，足りないときはスタックメモリを使用する。関数の戻り値も，同じく基本的にレジスタペアによって返される。

Z80－MPU用クロスアセンブラMRI-ASM80　6.0では，表6に示すようなセグメント配置が可能である。アプリケーションを最終的なロードモジュールにする場合，各セグメントの配置メモリアドレスを設定する必要がある。これには，特殊な定義ファイルが使われる。

ANSI C言語仕様についての詳細な解説は，これに関する参考書が多く出版されているので，そちらを参照されたい。

5.5.3　ATMSカーネルインタフェース'C.SYS'

ATMS/Z80で実行する制御用アプリケーションの開発を，C言語コンパイラを使用して行なう場合，ATMSカーネルとの最低限のインタフェースを整備しなければならない。特に，C言語では，プログラムのエントリポイントであるmain（）関数が，必ずしもモジュールの先頭にあるとは限らず，制御を返すexit（）がコールされた後はシステム側で処理しなければならない。

これらを解決する手段として，ATMS/Z80は，C言語インタフェース用モジュール「C.SYS」を用意する。このモジュールは，アプリケーションモジュールの先頭位置にリンクされる。Z80－CPU用のクロスCコンパイラMCC80　3.0では，main（）関数，exit（）関数がアセンブラレベルでは，それぞれ_main，_exitという外部参照シンボルで取り扱われる。C.SYSモジュールは，ATMSタスク管理システムで実行するための初期化手続きの後，プログラムエントリポイント_mainをコールする手続きと，_exitで制御を返された後のATMS/Z80のタスク管理システムから除去する手続きを行なうためのコードを含む。

表5　C言語関数の構成

Public データ宣言		static *type* c;
外部関数宣言		extern *type* extfunc (*type* arg..);
関数名，型，引数宣言		*type* function (*type* arg..)
関数本体	auto 変数宣言	{
	プロシジャ	*type* val = 10; ・ extfunc (val); ・
	戻り値設定	return (code); }

表6　Z 80-MPU用アセンブラASM 80　6.0のセグメント構成

SEGMENT	配　置　要　素
CSEG	関数名。本体プログラム。
XSEG	非初期化データ。（文字列など）
DSEG	static 変数データ。
STACK	auto 変数データ。一部の引数。

6．考　察

6.1　動作性能予測

ATMS/Z80は，入出力装置とのデータ入出力を中心としたプロセス標準化の具体的な手法として考案したものであり，機械制御と周囲環境の検出を同時に実行するような複雑なプログラムを，高い信頼性を維持しつつ，できるだけ効率よく開発することを目的の一つとしている。プログラムを単純化することは，単に無駄を省き，実行効率を向上させるのみならず，誤動作発生の機会を軽減し，デバッグ作業も容易になるという効果が期待できる。

機械制御系の自由度が多くなると，それぞれの自由度系を同時に制御するようなプログラムを，一つのアプリケーションとして構成するのは，コードも複雑化し，各々の自由度系をシーケンシャルに制御する手法となる。これに対して，プログラムコードを単純化し，複数の自由度系を時間的に並列して動作させるためには，個々の自由度の制御系をそれぞれ単独のCPUシステムで制御する手法を採るか，あるいは本システムの様に時分割で擬似的に並列処理を実行するかのどちらかを選択しなければならない。

MC－Z80に搭載したZ80A－CPUは，最大動作クロックが4MHzであることから，1クロックサイクルが0.25μ秒（1μ秒＝百万分の1秒）である。さらに，1つのステートメントを実行するのに，簡単なメモリの読み書きサイクルで4クロック，少し複雑な処理では10数クロックを要するから，Z80A－CPUはNo Waitで動作したとしても，1つのステートメントを実行するのに1～3μ秒かかる。時分割でタスク切り替えを行なう場合，タイマ割り込み要求が起こると，スケジューラは，その時点でのCPU内部のレジスタ内容をメモリに退避し，TCBテーブルを検査して次に実行すべきタスクを選定し，タスク管理メモリからCPUレジスタの内容を復帰して制御をCPUに渡す。この処理には240ステート以上のプログラムコードが必要で，およそ0.5m秒程度の時間を要する。このオーバヘッドに見合うタスク切り替え時間

を考察すると，10m 秒程度が適当と考え，MC－Z80 ではそのタイミングでタスク切り替えの割込み要求が発生するように設計した。

ここで，このタスク切替えタイミングと機械の制御性について考える。もし，1 自由度系の機械動作で，単純にこのタイミングで位置偏差をチェックし操作量を決定して制御しようとすると，50Hz 程度の周波数特性を得ることができる。機械制御では，数 10kHz の周波数特性まで評価するケースが一般的であるが，それほど厳しい特性が要求されず，動作域が小さく，慣性力の影響も小さいという条件の下では，十分利用に耐えうると考える。また，高い周波数特性が問題になるケースでは，入出力アダプタの設計において，アナログ系の周波数特性の高いドライバを採用し，プログラムは位置の指示だけを行なうようにするなど，トレードオフをハードウェアの方へ重くすればよい。その手法によれば，例えば 5 自由度系の並列処理で 10Hz 程度の制御性しか実現できなくても，適用の仕方によっては，かなりの動作制御に関するケースで利用できるものと考える。

6.2　解決すべき課題

マイコン制御でフィードバック制御手法を用いた位置決めを行なう場合，対象の位置の検出，偏差の計算，操作量の決定などといった一連のプロセスをプログラムしなければならない。このプログラムの実行の中で，多くの割合を占めるのは，実数計算の処理であろう。Z80－MPU は，実数はもとより，乗除算のインストラクション命令を持っていないため，実数計算は，ライブラリプログラムを利用しなければならない。しかしながら，その計算プログラムの実行は，同プロセッサにとって大きな負担となり，制御性能を圧迫してしまう。

例えば，16 語長のインテル i 8086 プロセッサのような，より高品位なプロセッサにおいても，浮動少数点実数どうしの計算は，負荷の大きいもので，これを軽減するために，i 8087 のような演算専用のプロセッサを用いるケースが一般的になっている。[9)]

本システムの設計においても，数値演算専用のプロセッサの利用も視野に入れて検討した。Z80－MPU は，特に利用すべき演算用プロセッサの指定はないため，個々の仕様に沿って検討した結果，モトローラ社製の MC－68882 というプロセッサが，Z80－MPU の I/O インタフェースと接続できることを見出した。しかし，動作クロックが異なるため，回路に工夫が必要であること，計算の内容によっては，1 つの計算に数百クロックが必要であること，浮動小数点実数のフォーマットが，IEEE 規格と異なり，フォーマット変換の手続きが必要であることなど，その利用にはまだ課題を残している。

世の中の動向を見ると，マイコン制御に持ちうるべきプロセッサも年々高性能化するとともに，価格も安くなっている。また，機械制御のような用途へは，DSP（Digital Signal Processor）と呼ばれるプロセッサの利用も進んでいる。当然そうした高性能プロセッサの利用も検討しているが，Z80－MPUの手軽さは，工業試験場によせられる技術相談の中でも，それほど大げさでない課題に適用するケースでは，依然利用価値を有しており，今後もしばらくその利用を検討する場面に出くわすであろうと考える。

7．おわりに

周囲環境の状況を検出しながら機械を制御する手段の一つとして，マイコン制御による制御手段に注目し，それを採用する際に課題となるプログラム開発効率の向上という視点から，その基盤となる基本システムについて検討を行なった。システムの基本設計にあたり，コンピュータ基本システムである OS の概念や，具体的には MS－DOS の設計思想を参考にし，とくに入出力系の部品化と複数プロセスの並列処理に力点を置いた。

また，ここで検討された内容は，別な研究課題である「PC ネットワークを利用した生産管理システム」や，その後に予定している「製造業における遠隔作業システム」などの検討課題での応用に供しながら，さらにその実用性について検討を進めていきたいと考えている。次報では，本システムの具体的な利用事例を中心に，様々な視点から性能を評価した結果を報告する。

参考文献

1）額田忠之：Z80 ファミリ・ハンドブック（CQ 出版社）
2）横田英一：マイクロコンピュータ　Z－80 の使い方（オーム社）
3）日本マイクロテックリサーチ：ASM80 ドキュメンテーションセット（1992．4）
4）日本マイクロテックリサーチ：MCC80 ドキュメンテーションセット（1992．4）
5）エー・ピー・ラボ：MS-DOS　エンサイクロペディア Vol.1（アスキー出版局）
6）エー・ピー・ラボ：MS-DOS　エンサイクロペディア Vol.2（アスキー出版局）
7）Phoenix Technologies： System BIOS forIBM PCs，Compatibles，and EISA Computers　2nd Edition（Addison-Wesley Publishing Co.Inc.）
8）石田晴久：パソコン言語学（アスキー出版局）
9）J.F.Palmer,S.P.Morse：8087 入門（御牧　義　訳，啓学出版）