U.D.C. 681.3.02：696／697

清水建設研究所報
第30号昭和54年4月

マイクロコンピュータのリアルタイム・オペレーティング・システムについて

——建築設備制御システムにおける適用事例——

佐藤豊勝
小西康之
坂東吉人
（エンジニアリング部）

§1. はじめに

建築設備の運転管理にコンピュータを利用しようとする試みは10余年前より，主にU．S．Aにおいて始められていたが，わが国でも1970年以後，ミニコンピュータの普及とともに一般に関心が持たれるようになってきた．

当初は，設備の制御性の向上とともに高度成長期における人手不足から省力化を目的とするものであったが，1973年のオイルショックを機として，省エネルギー・省資源指向が社会的に重要視されるようになり，さらにこれらシステムを構成する機器の性能対価格比の向上や，ソフトウェアを中心とする制御システム技術の進歩と相まって，ここ数年来急速に普及しつつある．

当研究所では，1972年秋に竣工した研究所本館において，その保有設備を対象としたミニコンピュータによる設備制御システムの運転を開始し，以後これをモデルシステムとして利用し，関連技術の研究開発を進めてきた[1]．

その後，前記システムの難点を改善し，さらにマイクロコンピュータをはじめとするLSIや半導体素子の利用範囲を広げ，大幅な性能の向上をめざした総合的な建築設備制御システムの開発を行ない，1976年秋に大阪で竣工した商業建築物において本システムが実用化されるに至った．現在，引きつづき「BECSS」の名称で多くの建物に適用されつつある[注]．

本文は，上記システムの中核をなすマイクロコンピュータの機能を，効果的に，かつ簡単に利用できることを目的として開発したリアルタイム・オペレーティング・システムについて，その設計方針，構造の詳細，ならびにトータルシステムにおける位置付けを述べたものである．

注）BECSS：Building Environmental Control System by Shimizu.

§2. 建築設備制御システムの概要

従来の建物の設備制御は，空調・給排水衛生・電気・防災・防犯などの設備に対し，個別に運転管理をローカル自動制御盤で行なっている．しかし，この方式では各設備間を横断した情報による総合的な判断を必要とする制御は不可能であり，また大型化されたビルでは多数のオペレータが必要になる．

これらの問題を解決し，より良い制御をするために，多量の情報を高速に処理し得るコンピュータが導入され，建築設備制御の総合管理を行なうシステムが開発された．

当社においては，システムの信頼性，拡張性，施工性および経済性を考慮して開発された独自のシステム（B-ECSS）が存在する．このシステムのデータ伝送制御装置にはマイクロコンピュータを使用しており，これに対して下記の処理機能が要求されている．

- 設備機器からの故障情報や防災・防犯機器からの警報情報を，一般監視制御情報に優先して即時処理しなければならない．
- 多種類で多数の設備制御情報を取り扱わなければならない．その内訳として，機器の起動・停止の入出力情報や状態値，温度・湿度などのアナログ入力値，流量や積算電力量のパルス入力値，モータバルブやダンパのステップ値などがある．
- 設備制御の運転周期は，1年，1週間，1日が基本となっているが，ここでは1日周期のスケジュール制御を必要とする．
- 導入される建物ごとに設備の規模が異なるため，システムの柔軟性を考慮する必要がある．

以上の要求を満足させるには，必然的に本システムが実時間で入出力情報を直接アクセスでき，またプロセス入出力の数ならびに種類の多様性に対応できるリアルタイム・オペレーティング・システムを必要とせざるを得ない．

§3. リアルタイム・オペレーティング・システム(ROS)の設計条件および目標

3.1 ROSの位置付け

当社の建築設備制御システムであるBECSSは，分散階層制御方式を採用しており，そのシステム構成を図—1に示す．このROSはデータ伝送制御装置（DTC）内のマイクロコンピュータ下で稼動し，設備機器の制御や情報の収集，上位コンピュータやオペレータズ・コンソール間における通信制御，データの記録などを統括・管理している．また，DTCは上位コンピュータのダウン時でも，システムとして最低限の運転を継続でき得るバックアップ機能を有している．この機能は，DTCの信頼性が上位コンピュータのそれより高いことが条件であり，その対策としてDTCの二重化が採用された．

次に，本システムの各構成要素の主な機能を以下に述べる（なお，詳細は所報Vol. 27[2]を参照）．

(1)上位コンピュータ

システムを一括制御するものであり，建築設備制御の中枢となるものである．高度な計算を必要とする応用プログラムや，多量のデータを使用する画面処理プログラムなどはここに位置する．

(2)データ伝送制御装置（DTC）

上位コンピュータとの通信，基本的監視制御，ICPの伝送制御，簡易スケジュール制御などの仕事や，上位コンピュータのダウン時にシステムのバックアップを行なう．

(3)バス切換装置

DTCの二重化に対するもので，正常動作のDTCにICPの制御権を渡す機能を有する．

(4)ICP

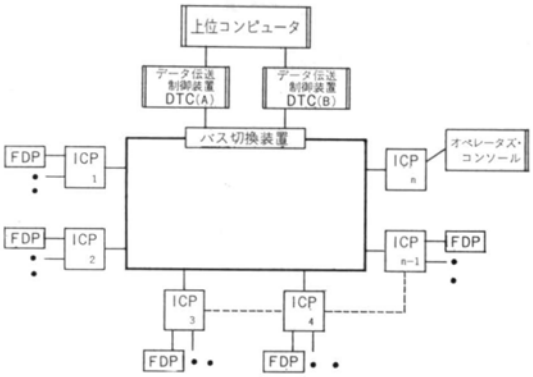
ICPは，DTCとFDPとの間に位置し，相互の信号を授受する装置であり，200～500点の制御要素ごとに1台の割合で設置される．各ICPは8ビットのアドレスを持ち，このアドレスをもとにDTCはICPを選択制御する．

(5)FDP

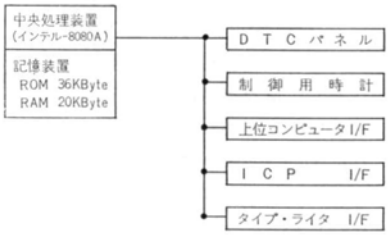
FDPは設備機器とICPとの間に位置し，相互の信号を授受し，信号レベルの交換を行なう装置であり，50～100点の制御要素ごとに1台の割合で設置される．

(6)オペレータズ・コンソール

ICPに対して接続されており，オペレータとの会話システムとして基本的な設備機器の監視や制御を行なう．DTC同様，マイクロコンピュータによるリアルタイム処理が行なわれている．



図—1 BECSSシステム構成図



備考）この他に，I/Oオプションとしてプラズマ・ディスプレイ，ドット・プリンタなどがある

図—2 DTCのハードウェア構成図

3.2 DTCのハードウェアの概要

DTCは，図—2のようなハードウェア構成を持つ．これから説明するROSは，これらのハードウェアをベースに設計するものであり，ハードウェアの良否はこのROSの性能にとって最も重要な要因である．そのために，ROSの開発の前段階として，構成しているハードウェアの性能を把握する必要がある．各構成要素の特徴を以下に示す．

(1)中央処理装置（CPU）

CPUとしてインテル社のマイクロコンピュータ(8080A）が選出された．その理由としては，最も一般的な品種であり，われわれの要求性能を満たしていると判断したためである．

(2)記憶装置

DTCの記憶装置として，プログラムや固定データ領域にROM（Read Only Memory）を，そしてプログラム・ワークや可変データ領域にRAM（Rand-

C P U 名	インテル 8080A
データ語長	8 bit/語
命令語長	8, 16, 24 bit
命令数	78命令
命令実行時間	2 μsec～9 μsec
メモリ容量	64K Byte (直接)
アドレス・モード	直接, レジスタ, レジスタ間接, イミディエート
レジスタ	アキュームレータ(8 bit) 1 ▼A▼ データレジスタ(8 bit) 6 ▼B·C·D·E·H·L▼ プログラムカウンタ(16bit) 1 ▼PC▼ スタックポインタ(16bit) 1 ▼SP▼
フラグ	キャリー, パリティ, サイン, ゼロ, 補助キャリー
入出力ポート	入力 256, 出力 256
バス構成	双方向性データバス (8 bit) アドレスバス (16bit)
クロック	2相 (MOSレベル) 最大2 MHz
スタック	外部メモリ(RAM)
割込み	多重レベル
電源	+12V, +5 V, −5 V, GND

表−1 インテル—8080Aの仕様

時刻	DTU	時刻	DTU
時：分	16進数	時：分	16進数
0：00	0000	12：00	4000
0：45	0400	12：45	4400
1：30	0800	13：30	4800
2：15	0C00	14：15	4C00
3：00	1000	15：00	5000
3：45	1400	15：45	5400
4：30	1800	16：30	5800
5：15	1C00	17：15	5C00
6：00	2000	18：00	6000
6：45	2400	18：45	6400
7：30	2800	19：30	6800
8：15	2C00	20：15	6C00
9：00	3000	21：00	7000
9：45	3400	21：45	7400
10：30	3800	22：30	7800
11：15	3C00	23：15	7C00

表−2 DTU—時刻対応表

am Access Memory) を使用している.

(3)DTCパネル

DTC パネルは, 割込みやデータ入力用の各スイッチと表示ランプより構成されており, 各種のシステム・テストに使用している. 通常は, CPU における最上位割込みレベルに接続される.

(4)制御用時計

制御用時計の機能として, CPU に対し一定時間間隔ごとに割込みを発するインターバル・タイマと, 単位時刻ごとに割込みを発する DTU (Day based Time Unit) とがある.

(5)上位コンピュータ・インターフェイス

上位コンピュータと DTC 間のインターフェイスであり, DTC の入出力情報を上位コンピュータに伝える.

(6)ICP・インターフェイス

ICPのループを構成するメインバスに対するインターフェイスであり, DTC の制御情報をICPへ伝え, またICPからの入力情報を受け入れる.

(7)タイプライタ・インターフェイス

DTCとタイプライタ間のインターフェイスであり, DTC の入出力情報をタイプライタに伝える.

以上の他に, オプションとしてプラズマディスプレイや, ドットプリンタなどのインターフェイスを保有している.

3.3 ROSの設計目標

前述のように, DTC 内においてその情報交換の主要な役割を有するマイクロコンピュータを効率よく稼動させるために, 以下の項目を設計目標に掲げ, ROS の開発を行なった.

(1)使い易さの追求

プログラマーやオペレータにとって, あくまでも便利で使い易い機能が揃ったシステムをめざす.

- 制御用時計を活用して, 豊富なタイマ・サービスを用意する.
- ICPの入出力制御に対し, システム・サービスを提供することでユーザの負担を軽減する.
- ハードレジスタの他に擬似レジスタを設けることでプログラム効率を上げる.
- タスクにレベルを付加し優先処理を行なう.

(2)ハードウェア機能の最大限の活用

アプリケーション・プログラムが, ハードウェア機能を有効に活用できるようなシステムにする必要がある.

- メモリの有効な割り当てを考慮する (データテーブルやスタックなど).
- マルチプログラミングにおける入出力システムの管理を行なう.

(3)システムの適応性と拡張性

ROS 自から諸能力を拡張でき, また機能の変更や組合せが容易に行なえるしくみを考える.

- モニタプログラムのモジュール化を促進する.
- 各建築物において数の異なるICPに対しては, 拡張性が充分に配慮されたデータ構造を設計する必要がある.

(4)応答時間の短縮

リアルタイム処理での応答時間は短かいほど良い.
・伝達システムの開発を行なう.
・プログラム・ステップを短縮する.

§ 4. リアルタイム・オペレーティング・システム(ROS)の機能

ソフトウェアは大別してプログラムの中枢をなすシステム・プログラムと，その下で実際の処理を行なうアプリケーション・プログラムとに分かれる．すなわち，このROSと設備制御を行なう処理プログラムが密接に結合されて大きな効果が得られるものである．以下に，今回開発したROSの詳細を述べる．

4.1 ROSの構成

ROSは，多数のプログラムにより構成されており，その機能目的から，割込み管理，タスク管理，システム・サービス管理，入出力管理のプログラム群に分類される．各プログラム群は図—4のように構成され，多重プログラミングの手法により基本的なリアルタイム処理を実現している．

入出力管理プログラム群を特にIOCS(Input Output Control System)と総称し，ROSの核となる部分と分割されている．これにより，入出力装置の変更に対する柔軟性と拡張性をもたせた．この中で，制御用時計の管理プログラムは他の入出力管理の動きとは異なる．これはタイマ管理プログラムとして，IOCSから独立したものと考えられる．

4.2 ROSの機能

4.2.1 割込み管理

このROSは，ハードウェアの割込み機能を充分に活用することでリアルタイム処理を行なっている．割込みとは，ハードウェアのある事象の発生時や，プログラム中の特別な命令により，プログラムの正常な実行を中断させる例外的な機能である．この機能を有効に活用することによって，プログラムの優先処理や並行処理を実現できる．プログラムにより要求される内部割込みについては後でシステム・サービスの項目で述べる．ここでは，ハードウェアによる外部割込みについて説明する．

A．外部割込みの種類

(1)基本割込み

・DTCパネル割込み

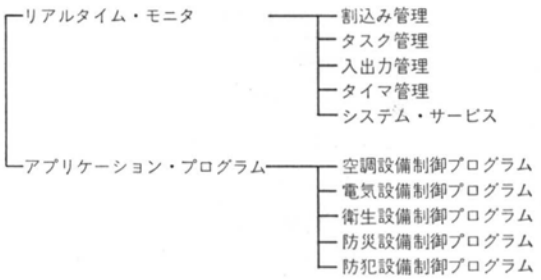
DTCパネル上から特定のタスクを起動することができる機能を持たせた．

・制御用時計割込み

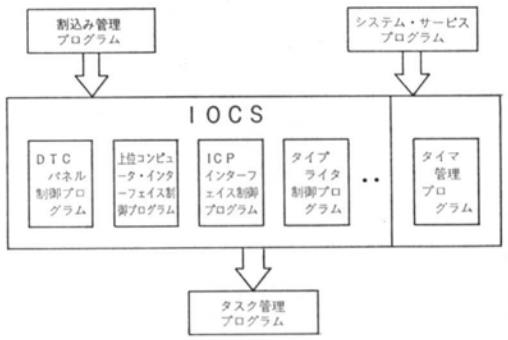
DTUクロックおよびインターバル・クロックの割込みにより，タイマ要求に対する処理を行なう．

(2)入出力装置割込み

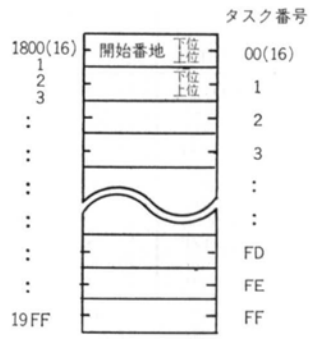
各入出力装置ごとに割込み原因が定義されている．一般的には，CPUへ装置側から処理要求を依頼する要求



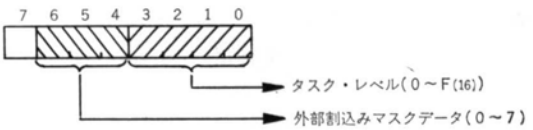
図—3 ソフトウェアの体系



図—4 ROSの構成



図—5 タスク・エントリー・テーブル



図—6 タスク・レベルの仕様

割込みと，CPU からの要求に対する処理の終了を知らせる入出力動作完了割込みに分けられる．

B．外部割込み処理

外部割込みが発生すると，各割込みレベルに対応したリスタート命令がハードウェアにて作られる．CPU はこの命令を実行した結果，その時点でのプログラム・カウンタの値をスタックに移し，リスタート命令に対応した固定のアドレスから実行を開始する．

以上の手順を経て割込み解析ルーチンに制御が渡る．ここでは，割込みにより CPU の使用権をうばわれたタスクの存在を確認し，有ればレジスタ類を退避した後に割込み要因に対応した処理ルーチンに制御を渡す．各処理ルーチンにて処理が終るとモニタ・エンド・ルーチン（スケジューラ）に制御が渡り，実行権を有するタスクの存在を確認し，有ればそのタスクのレジスタ類を復旧し，外部割込み許可命令を出した後に CPU の使用権を与える．もし無い場合には，外部割込み許可命令を出した後に停止する．

4.2.2 タスクの管理

タスクは，システム中の資源（CPU，スタック・エリア・ブロック，入出力装置，プログラム，メモリーなど）を利用する最小の仕事の単位である．ROS では，スタック・エリア・ブロック（SAB）に基づいて管理を行なう．タスクは，起動時に初期データとして2バイトのデータを持つことができ，この特質を有効に活用することで，プログラムのモジュール化およびプログラム効率を高めることができる．

A．タスクの特徴

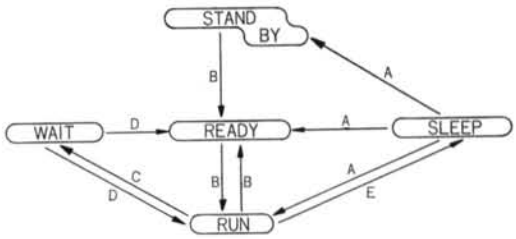
登録できるタスクの数は最大256本とし，内訳はシステム用タスクに32本，残り224本をアプリケーション用タスクに割り当てている．タスクは，タスク・エントリー・テーブルにその実行開始番地を登録することによって認識され，タスク番号が与えられる(図—5)．

また，タスクには優先順位を付けることができ，タスク・レベル・テーブルのタスク番号で示される1バイトの領域に登録する．このタスク・レベルは，上位3ビットで外部割込みのタスク制御を行ない，下位4ビットでタスクの優先順位を表わしている(図—6)．

B．タスクの状態

ROS がタスクの起動要求を受付けてからタスクの停止要求を受け取るまでに，タスクは図—7に示すような幾つかの状態遷移をとる．このタスクの状態は，SAB および TRB（Task Request Block）の待ち行列（Queue）のリンク状態によって知ることができる．

(1)停止状態（SLEEP）



A：タスクの起動（エンター・マクロ，タイマ起動マクロ）
B：スケジューリング
C：事象待ち（入出力装置使用マクロ，ポーズタイム・マクロ）
D：事象待ち終了
E：タスクの消滅（リターン・マクロ）

図—7　タスクの状態遷移

起動要求の出される前の状態，および停止要求を出して実行を終了した後の状態である．すなわち，SAB も TRB も所有していない状態である．

(2)実行待ち状態（READY）

SAB を獲得し，実行準備が整っているにもかかわらず，それより優先順位の高いタスクが実行中のため，実行権を与えられない状態である．

(3)実行準備待ち状態（STAND-BY）

起動要求を受付けられたがSABを獲得できず，TRBに起動要求が記憶され，SABの空きを待つ状態である．

(4)実行状態（RUN）

SABを獲得し実行権を得て，CPUを占有している状態で，実行待ち行列の最前列に SAB がリンクされていることにより示される．

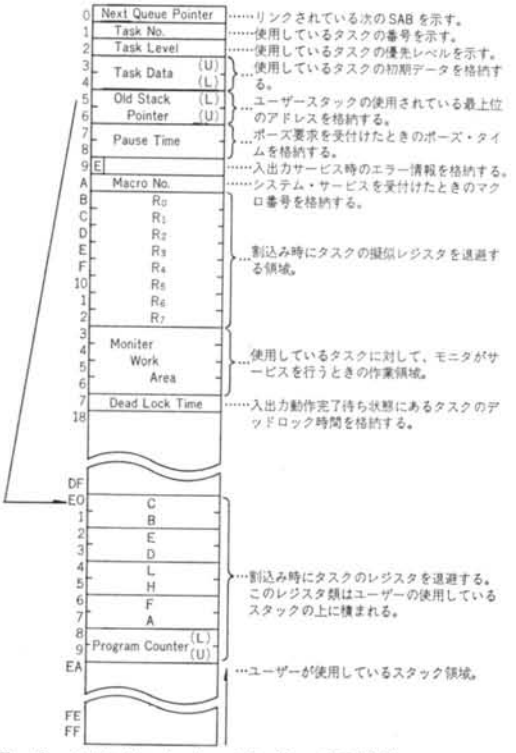
(5)事象待ち状態（WAIT）

SABを獲得しているが何らかの事象の完了，つまり入出力動作の完了（入出力サービス）や指定時間の経過（PAUSEサービス）を待って，一時実行権を放棄している状態である．

C．SABとTRB

SAB がタスクの実行時に割り付けられることは前述した．ROS では，タスクの実行時におけるスタック・エリアとタスク・コントロール・テーブルを一緒にし SAB と定義した．1個の SAB は256バイトであり，8個の SAB をメモリ（RAM）に定義している．この複数の SAB を利用して，タスクの並行処理を可能とした．また，タスクはコントロール・テーブルに続く領域にワーク・エリアを持ことも可能である．

TRBは，タスクの起動要求後の STAND-BY 状態が記憶された4バイトからなるブロックで，SABの空きを待つ行列にリンクされている．またこのブロックは，タイマ・サービスによってタイマ起動要求を記憶するTIB



図－8 SAB (Stack Area Block) の使用状態

(Timer Information Block)としても使用される．TRB と TIB に共通なブロックは，メモリ（RAM）に128個定義されている．

4.2.3 入出力管理

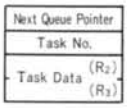
一般に，入出力装置の動作時間は CPU に比べて緩慢であり，制御手続きは極めて複雑である．この入出力装置の動作中の遊び時間を有効に利用し，ユーザーが個々に複雑な入出力制御プログラムを作成することなく，簡単に利用できるように ROS で入出力装置の一括管理を行なった．

ROS では，入出力装置による外部割込みに対する処理をレスポンス・ルーチン，そして入出力装置使用サービス要求に対する処理をコーリング・ルーチンと区分している．すなわち，入出力装置を起動する処理がコーリング・ルーチンで行なわれ，それに対する入出力動作完了割込みによる後処理や，入出力装置からの要求割込みに対する処理がレスポンス・ルーチンで行なわれている．IOCS は，基本的にこれらのルーチンによって構成されている．

A．入出力の種類

入出力装置として，上位コンピュータ•インターフェイス，ICP，タイプライタを具備している．これらの他にプ

1) TRB(タスクのSTAND-BY状態を表わす)



2) TIB(各タイマ・サービスの状態を表わす)

図－9 TRBとTIBの使用状態

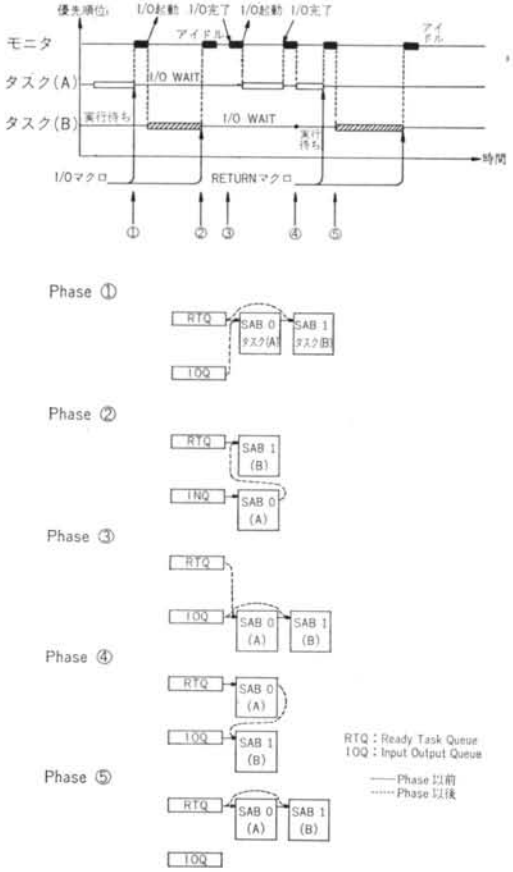
ラズマ・ディスプレイ，ドット•プリンタなどの入出力装置がオプションとして存在する．これらの中で，設備機器とのインターフェイスをなすICPは次に掲げる種類の信号を処理している．このため IOCS では，以下のそれぞれの信号に対して個別の制御ルーチンを有している．

- DO (Digital Output)
- DI (Digital Input)
- PTO (Potentiometer Output)
- AI (Analog Input)
- LPI (Low Speed Pulse Input)
- HPI (High Speed Pulse Input)
- IDI (Interrupt & Digital Input)
- IDIDO (Interrupt & Digital Input/Output)
- OPECON (オペレータズ・コンソール・インターフェイス)

上記のうち，AI，HPI，オペレータズ・コンソール（操作卓）以外は入出力動作待ちが無く，データを即時入出力できる．

B．入出力動作の管理

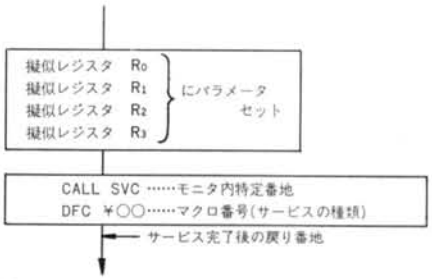
ユーザーが入出力装置を利用する場合，プログラム内で ROS により拡張された擬似レジスタ（特定の RAM メモリを擬似レジスタとして 8 バイト定義している）に必要な情報をセットし，ROS に対して入出力サービス要求を出す手順で行なわれる．このサービス要求により，指定された入出力装置に対応するコーリング・ルーチンが起動され，入出力処理が実行される．



図—10 入出力管理のタイムチャート

入出力動作において，多重処理される複数のタスクが同時に1台の入出力装置を使用することはできない．そこで，1台の入出力装置を競合して使用することを避けるため，各入出力装置ごとに待ち行列を作り入出力装置の運用管理を行なった．したがって，タスクが入出力サービス要求を出すと，そのタスクは入出力装置の動作終了まで無条件に事象待ち状態となり，他のタスクに実行権を空け渡すことになる．これによって，タスクの並行処理が可能となる．

入出力装置の異常状態により入出力動作完了割込みが返されない場合，その入出力装置に対する事象待ち状態のタスクはデッドロック状態に陥る．そして，次々にこの入出力装置を使用するタスクは全てその状態となるため，最終的にシステム全体がデッドロック状態に陥ることになる．これを回避する方法として，各入出力装置ごとに特定の時間を定め，その時間内に応答が無い場合には事象待ち状態のタスクを強制的に消滅させ，オペレータに知らせる方式を取っている．この処理をデッドロック処理と呼び，システム全体のデッドロック状態を無くしている．



図—11 システム・サービス要求形式

図—10に入出力動作のタイム・チャートを示す．

4.2.4 タイマ管理

設備制御システムにおいて，計時機能は必須条件であり，DTC ではシステム全体の基本となる制御用時計を具備している．この制御用時計は2つの機能を有している．1つはDTU（Day based Time Unite）と呼ばれるシステムの時刻を決定するもので，他方は任意の時間間隔を取り出せるインターバル・タイマである．

DTU は24時間（86,400秒）を15ビットで表わされる最大数（32,768）で分割した2.$_{63671875}$秒を制御の単位時間としたもので，DTU が $7FFF_{(16)}$ から0に変化した時点で1日が更新され，午前0時0分が DTUの0に相当する．これは2進計数を基本とするコンピュータ・システムにおいて，時刻の計数を簡単化するための方法として用いている．

インターバル・タイマは8ビットで構成されており，設定範囲に応じての最大値1/16秒・1秒・16秒・256秒の4種類が用意されている．ROS は1/16秒を使用し，入出力装置のデッドロック・タイムの監視，タスクのポーズ・タイムの監視を行なっている．

上記の制御用時計により，ユーザーに対して容易に使用できる各種のタイマ・サービスを提供している．

- 時刻指定によるタスクの起動
- 時刻指定によるタスクの繰返し起動
- 時間間隔指定によるタスクの起動
- 時間間隔指定によるタスクの繰返し起動
- 繰返し起動要求の取消し
- 時間間隔指定によるタスクの休止（ポーズ）

4.2.5 システム・サービス

システムの運用にあたって，入出力制御・タイマ管理などのユーザーに必要な共通の処理をシステム・サービスとし，総合的に管理をする．

A．システム・サービスの要求形式

システム・サービス		マクロ番号	擬似	擬似レジスタの内容		備考
名称	機能	(16進)	レジスタ	呼び出し時	復帰時	
RETURN	タスク自身の実行終了を宣言する	00	R_0	——	——	
			R_1	——	——	
			R_2	——	——	
			R_3	——	——	
ENTER	タスクの起動を要求する	10	R_0	——	——	登録されていないタスクへの起動要求は無視される
			R_1	タスク番号	変化せず	
			R_2	タスク・データ	〃	
			R_3	タスク・データ	〃	
DTU(M) CLEAR	定時刻で起動するタスクの解除を行なう	20	R_0	——	——	該当するものが見つからない場合は無効な命令となる
			R_1	タスク番号	変化せず	
			R_2	DTU (U)	〃	
			R_3	DTU (L)	〃	
DTU(M) SET	定時刻で起動するタスクの登録を行なう	21	R_0	——	——	設定時刻が負の場合，動作の保障はできない
			R_1	タスク番号	変化せず	
			R_2	DTU (U)	〃	
			R_3	DTU (L)	〃	
INTERVAL(M) CLEAR	定時間間隔で起動するタスクの解除を行なう	22	R_0	——	——	該当するものが見つからない場合は無効な命令とみなす
			R_1	タスク番号	変化せず	
			R_2	Interval Time	〃	
			R_3	——	——	
INTERVAL(M) SET	定時間間隔で起動するタスクの登録を行なう	23	R_0	——	——	設定時間間隔が0で指定された場合は，最大時間11分と認識される
			R_1	タスク番号	変化せず	
			R_2	Interval Time	〃	
			R_3	——	——	
DTU(S) SET	一度だけ指定時刻に起動するタスクの登録を行なう	25	R_0	——	——	設定時刻が負の場合，動作の保障はできない
			R_1	タスク番号	変化せず	
			R_2	DTU (U)	〃	
			R_3	DTU (L)	〃	
INTERVAL(S) SET	一度だけ指定間隔時間に起動するタスクの登録を行なう	27	R_0	——	——	設定時間間隔が0で指定された場合は，最大時間45分と認識される
			R_1	タスク番号	変化せず	
			R_2	Interval Time (U)	〃	
			R_3	Interval Time (L)	〃	
PAUSE	指定した時間だけ待ち状態を宣言する	28	R_0	——	——	指定時間が0で指定された場合は，16秒となる
			R_1	——	——	
			R_2	Time	0	
			R_3	——	——	
HOST-COM WRITE	上位コンピュータとの通信を行なう	30	R_0	割込み情報	変化せず	通信時にハードウェア・エラーにより割込みが確認できない場合，デッドロック処理が行なわれる
			R_1	コマンド情報	〃	
			R_2	データ	〃	
			R_3	データ	〃	
ICP WRITE	ＩＣＰ下の各種信号源に対しデータを出力する	50	R_0	ICP アドレス(U)	変化せず	
			R_1	〃 (L)	〃	
			R_2	OUTPUT DATA(1)	〃	
			R_3	〃 (2)	〃	
ICP READ	ＩＣＰ下の各種信号源からデータを引き上げる	51	R_0	ICP アドレス(U)	変化せず	AIに関してはデッド・ロック処理有り
			R_1	〃 (L)	〃	
			R_2	サブ アドレス	INPUT DATA(1)	
			R_3	——	〃 (2)	
OP-CON WRITE	オペコンに対して指定のタスクの起動を要求する	58	R_0	——	——	デッド・ロック処理有り
			R_1	オペコン側タスク番号	変化せず	
			R_2	タスク・データ	〃	
			R_3	タスク・データ	〃	
TYP WRITE	タイプライタへの印字処理を行なう	60	R_0	——	——	印字コードはJIS仕様であり，印字数は256が最大である デッドロック処理有り
			R_1	バッファ先頭番地(U)	——	
			R_2	〃 (L)	——	
			R_3	印字数	0	
TYP READ	タイプライタの鍵盤から文字の読み込み処理を行なう	61	R_0	——	——	(CAN) コードが打鍵された場合，初期状態にもどる
			R_1	バッファ先頭番地(U)	——	
			R_2	〃 (L)	——	
			R_3	終了コード	変化せず	
DOT WRITE	ドットプリンタの印字処理を行なう	64	R_0	——	——	印字コードはASCII仕様であり，最大16行まで可能である デッドロック処理有り
			R_1	バッファ先頭番地(U)	——	
			R_2	〃 (L)	——	
			R_3	打ち出し行数	0	
DOT FEED	ドットプリンタのフィード制御を行なう	65	R_0	——	——	最大16行のフィードが取れる デッド・ロック処理有り
			R_1	——	——	
			R_2	——	——	
			R_3	改行行数	0	
DISP WRITE	プラズマ・ディスプレイの印字処理を行なう	68	R_0	——	——	デッド・ロック処理有り
			R_1	バッファ先頭番地(U)	——	
			R_2	〃 (L)	——	
			R_3	表示データ数	0	
DISP READ	プラズマ・ディスプレイの鍵盤からの読み込み処理を行なう	69	R_0	——	——	デッド・ロック処理有り
			R_1	バッファ先頭番地(U)	——	
			R_2	〃 (L)	——	
			R_3	終了コード	変化せず	

表—3

サービスの要求形式は，サービス要求の宣言，サービスの種類を示すマクロ番号，サービスに必要なパラメータ部からなっている．

まず，擬似レジスタに指定された形式に従ってパラメータをセットする．そして，モニタ内の特定番地をコールする．マクロ番号はコール命令直後に1バイトで指定する(**図—11**参照)．

上記の形式で，ROSに対するサービス要求が行なわれる．このサービス要求が受付けられ処理が完了すると，タスクは復帰する．このとき，擬似レジスタに復帰情報が与えられ，サービス中のエラーを知ることができる．

表—3に各システム，サービスの種類とそのパラメータ，復帰情報を示す．

B．サービス要求処理

前述の形式に従ってシステム・サービス要求があった場合，ROSはこれを内部割込みとして受付け，以下の処理を行なう．

(1)要求を出したタスクのレジスタ類を，そのSAB内に退避する．

(2)マクロ番号を取り出し，対応するサービス処理ルーチンに制御を渡す．

(3)すべての要求処理が完了した場合，元のタスクに復帰させるか，もしその要求がタスクの終了要求であった場合には，そのタスクを停止状態とする．

§5. リアルタイム・オペレーティング・システム(ROS)の評価

5.1 実施システムにおける問題点

当社では，建築設備制御システムを1976年以後，BECSSの名称で数例を完成させている．これまでの経過よりROSについても，単なるプログラムの虫からシステム仕様の変更を要するものまで，いくつかの問題点があった．しかし，その都度対処してきたため，現在では一応その信頼性を確信できる状況にある．

しかしながら，トータルシステムとしての要求性能からROSを再検討したとき，なお以下のような機能改善のための努力が必要であろう．

(1)プログラムのテストおよびデバッグ・ツール

DTCでは，メモリ容量の制限があるうえ，記憶素子にROMを使用しているため，プログラムのテストやデバッグが困難なシステムとなっている．現在は，プログラムの開発を外部計算機で行ない，プログラム・テストはRAMにロードして行なっている．今後は，プログラム開発システムの機能を，より一層充実させていく必要がある．

(2)外部システムとの通信

外部システムとDTC間の通信量が，当初想定していた通信量に比べ増大する傾向になりつつある．これによるシステム効率の低下を防ぐために，通信システムを改良する必要がある．

(3)ハードウェア・エラー処理について

特に，ICPに関する入出力エラーが重要である．ROSのエラー処理は完全とはいえず，ハードウェアのメンテナンス情報へ高めるまでには至っていない．

(4)システム・サブルーチンの管理

システム内には多種類のサブルーチンが存在しているが，形式の統一は行なわれていない．このままでは類似したサブルーチンが数多く存在することになり，メモリ効率の低下につながる．今後は，サブルーチンの形式を統一し，基本サブルーチンや応用サブルーチンをシステム・サブルーチンに昇格させ，ROSで管理する方式を検討する必要がある．

5.2 今後の課題

マイクロコンピュータにおけるROSを開発し実施してきた結果として，従来ミニコンピュータ以上のシステムの領域であったオンライン・リアルタイム制御に，その内容によっては，マイクロコンピュータが充分使用に耐え得ることが明らかになった．しかしその反面，プログラムの作成に多くの問題点が見い出された．これは，ハードウェアの構成がソフトウェアに大きく影響することを意味しているものである．だがしかし，マイクロコンピュータを取りまくハードウェア技術は，まだ進歩の足を止めはしないであろうから，その解決を将来に期待して良いであろう．

このROSに要求される今後の課題は，前節の問題解決に参加することであり，またマイクロコンピュータで実現可能なROSの機能を追求することである．これには，今後のBECSSの運転実績を待たねばならないものも数多く存在すると思われる．

謝辞 このROSの開発にあたり，資料の提供に御協力いただいた計画研究部の山田邦夫・桜井仁の両氏およびエンジニアリング部BECSSグループの方々に深く感謝致します．

＜参考文献＞

1) 荒木睦彦他："清水建設研究所設備のコンピュータ制御システム（その1～その4）" 清水建設研究所報 Vol.22 (1974)
2) 桜井仁他："ビル設備制御における伝送システムの開発" 清水建設研究所報 Vol.27 (1976)
3) 江村潤郎："オペレーティングシステムへの構造的アプローチ(上)" 日本コンピュータ協会 (1972)
4) 穂坂衛他："実時間システム概論" 共立出版 (1970)
5) 穂坂衛他："実時間システム開発" 共立出版 (1970)
6) Glenford J. Myers："ソフトウェアの信頼性" 近代科学社 (1977)
7) 吉谷龍一："ワークデザイン" 日刊工業新聞社 (1974)