U.D.C. [681.32.06:658.562.011.56]:[681.323.014:681.5]

制御用ソフトウェア機能一貫テストシステム“HITEST/F”

Functional Testing Facilities for Industrial Software System ``HITEST/F''

大島啓二* *Keiji Ôshima*
林　利弘* *Toshihiro Hayashi*
海永正博** *Masahiro Kainaga*
薄井勝夫*** *Katsuo Usui*

日立製作所は，ソフトウェア開発工程の60%近くを占める，テスト及びデバッグ作業の信頼性，生産性を向上させることができる，ソフトウェア機能一貫テストシステム“HITEST/F”を開発した。

“HITEST/F”では，テスト手続記述言語TPL/IIによって静的なテストが，またプロセスシミュレータによって動的なテストが効率的に行なえるようにすることによって，プログラムのモジュールテストから組合せ・総合テスト，調整テストまでを一貫して支援している。

本システムは，既に実システムに幅広く適用され，制御用ソフトウェアのテスト作業で，所期の効果を挙げている。

1 緒　言

ソフトウェアの生産に当たっては，品質の高いソフトウェアを高い生産性で製作できることが重要である。これを達成するには二つのアプローチがある。一つはまず，設計・製作段階でソフトウェアの不良が紛れ込むことを防ぎ，「バグ(不良)のないプログラム」を作るようにするアプローチであり，他の一つは，紛れ込んでしまった不良を効率良く摘出し修正するというアプローチである。これら二つのアプローチを調和させることによって，初めて高品質のソフトウェアを生産性高く実現できる。日立制御用コンピュータHIDIC 80シリーズのソフトウェア生産支援のための各種ツール[1]の中で，SPL[2](Software Production Language)は前者のアプローチを支援するものであり，本稿で述べる“HITEST/F”[1](Hitachi Integrated Test System/Function)は後者のアプローチを支援するものである。

“HITEST/F”は，

(1) テスト方法の標準化
(2) テスト操作の自動化
(3) テストデータ作成の容易化
(4) テスト結果確認の容易化

をねらいとして開発したもので，昭和49年に開発したTPL(Test Program Language)[3]の適用経験を基に，個々のテスト支援機能を見直し，更にテスト範囲をモジュールテストレベルから全テストフェーズへと拡大し，ソフトウェア機能の一貫テスト支援システムとして体系化したものである。

本稿では，“HITEST/F”の開発思想，方針，構成，機能及び適用状況について述べる。

2 “HITEST/F”開発の基本思想と方針

計算機制御システムの品質確認では，

(1) 一つ一つのプログラムのアルゴリズムのテスト(モジュールテスト)が効率良く，かつ信頼性高く行なえるようにすること。
(2) プログラムを複数個組み合わせ並行動作させ，システムとしての機能が正しいことを確認するテスト(システムテスト)が効率良く，かつ信頼性高く行なえるようにすること。

が重要である。更に計算機制御システムでは，

(a) プロセス側の事情もあり，必ずしもコンピュータ主導形で調整を行なえない場合も多い。
(b) プロセスとの結合試験時の，不具合原因の切分けが難しい。

といった問題があるため，

(3) 現地でプロセスと結合する前に，プロセスとのインタフェースの確認を含めた品質確認が，事前に十分行なえるようにすること。

も必要なことである。

以上述べたようなニーズを踏まえ，近年発展の著しいソフトウェア工学技法を積極的に取り入れ，計算機制御ソフトウェア開発作業の流れに沿って一貫した考え方で品質確認，テストが行なえるようにしたシステムが“HITEST/F”である。その開発基本方針は，以下に述べるとおりである。

(1) 一貫テスト支援

テスト作業の方法を標準化し，かつ自動化することにより，テスト効率及び信頼性向上の実現を全テストフェーズにわたり可能とする。そのために，テスト手続記述言語TPL/IIを，プログラムのモジュールテストから組合せ・総合テスト，調整テストまでを統一して支援できるようにする。

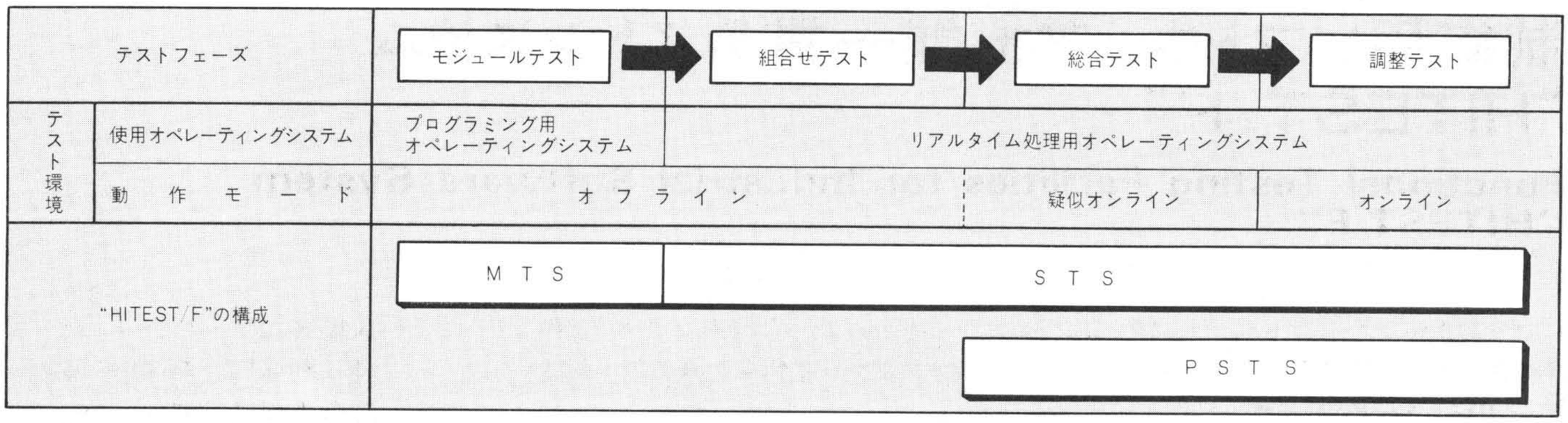
(2) 疑似オンラインテスト支援

プロセスとコンピュータを直接接続できない段階でも，プロセスとのリンケージテストを疑似オンラインテストとして行なえるようにする。そのために，コンピュータとプロセスの間のインタフェースを標準化し，この標準インタフェースの上にプロセスシミュレーションテストを実現する。プロセスシミュレーションテスト機能には，TPL/IIを用いた静的テストと，プロセスモデルによる動的テストの2種類を設ける。

(3) TPL/IIのホスト言語独立性

“HITEST/F”は，どのような言語で書かれたプログラムで

* 日立製作所大みか工場　** 日立製作所システム開発研究所　*** 日立製作所日立研究所



注：略語説明 "HITEST/F"(Hitachi Integrated Test System/Function) STS(System Test System)
MTS(Module Test System) PSTS(Process Simulation Test System)

図1 "HITEST/F"の構成 "HITEST/F"は，MTS，STS及びPSTSの3サブシステムから成り，全テストフェーズをカバーしている。

もテストできるテストシステムとする。そのために，テスト手続記述言語TPL/IIをホスト言語(被テストプログラム記述言語)に依存しない形とする。

3 "HITEST/F"の構成

3.1 "HITEST/F"を構成するサブシステム

図1にテスト環境(テストを行なう際に前提となるオペレーティングシステムやハードウェアの条件)と，その環境下での"HITEST/F"の各サブシステムの位置づけを示す。

MTS(Module Test System)は，主としてオフラインのモジュールテストを効率的に行なえるようにするサブシステムであり，バックグラウンドのプログラミング用オペレーティングシステムの下で動作する。

STS(System Test System)は，タスクの単体テストから組合せ・総合テスト，更に調整テストまでを支援するサブシステムであり，フォアグラウンドのリアルタイム処理用オペレーティングシステムの下で動作する。

PSTS(Process Simulation Test System)は，従来オンサイトだけで行なわれていたオンラインテストを，できるだけ早い段階で疑似オンラインテストとして行なえるようにするサブシステムである。動的なテストデータの生成は，PMDL(Process Model Description Language：プロセスモデル記述言語)を用いて作られるプラントや，プロセスのモデルによって行なわれる。

3.2 MTS，STSとTPL/II

従来のテスト作業は，各々のテスト作業者が，個別に用意されたテストツールを用いて行なっていた。そのためにテスト方法が統一されず，テスト自体も個人の能力に依存したレベルの低いものとなり，全体としてテスト作業の効率，信頼性の低下を招いていた。

こういった問題を解決するために，"HITEST/F"では，

(1) テストデータを含むテスト手順を一種の「プログラム」としてとらえ，更にこの「テストプログラム」を通常のプログラムより容易に，かつ信頼性高く作成できるようにする。

(2) このテストプログラムを，「実行モニタ」によって効率良く実行させる。

ことによって，テスト手順の標準化，テスト操作自動化の実現を図った。

TPL/IIは，この「テストプログラム」を記述するための言語であり，MTS，STSは実行モニタに相当する。MTS，STSとTPL/IIの関係を**図2**に示す。

4 "HITEST/F"の機能

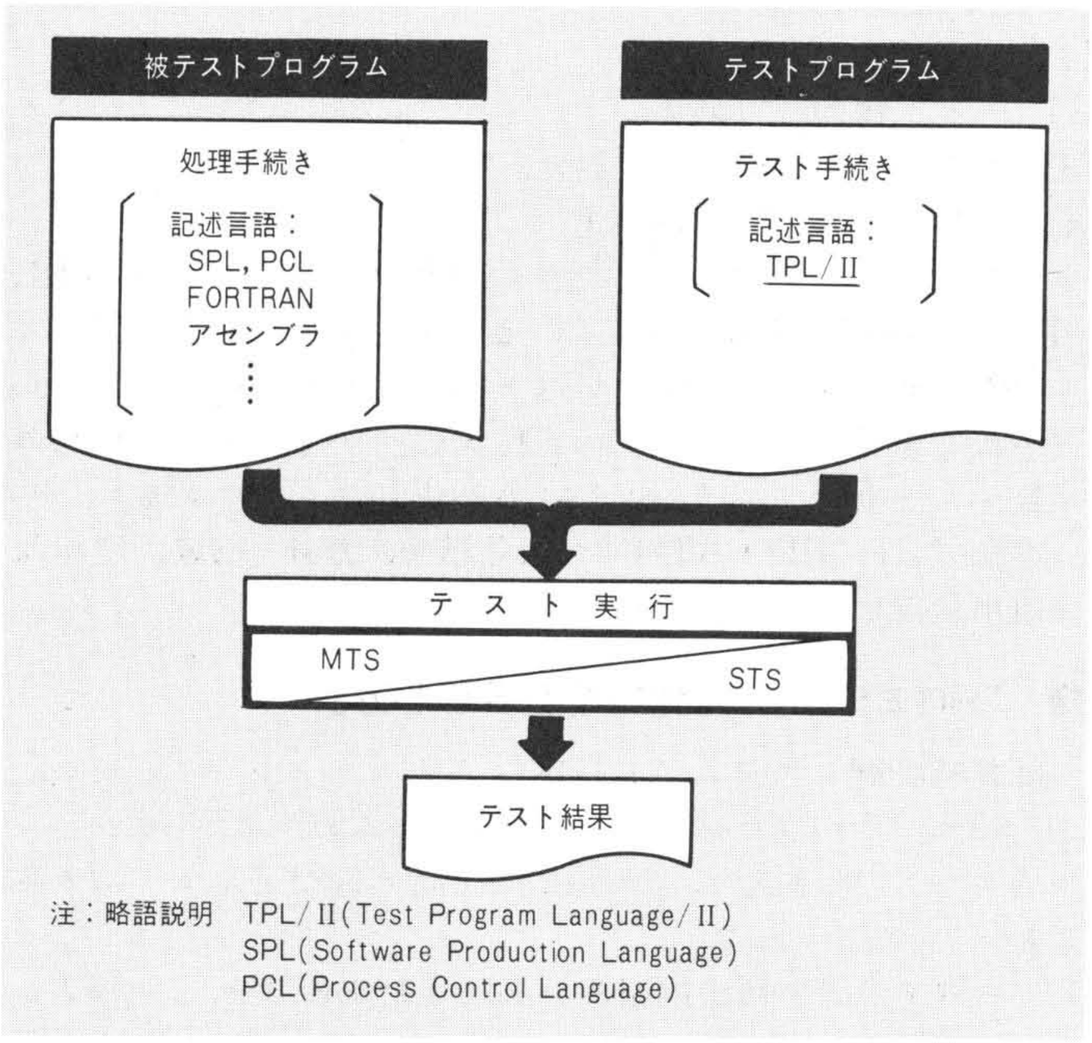
"HITEST/F"は**図3**に示すような機能をもっており，これらの機能を利用して，ユーザーは効率良くかつ信頼性高くテストを行なうことができる。以下に各機能の概要について説明する。

4.1 一貫テスト機能

プログラムのモジュールテスト及びシステムテスト(組合せ・総合テスト，調整テスト)で，各テストフェーズの特異性を吸収しながら，一貫した手順，方法でテストが行なえるようにした機能であり，具体的にはこれを「モジュールテスト及びシステムテストでのテストプログラムの共通使用」によって実現している。その結果，

(1) あらかじめ総合テストを想定して，システムテスト用のテストプログラムを作成しておけば，テストプログラムの作成が全テストフェーズを通じて1回で済む。

(2) 被テストプログラムに変更が生じた場合にも，システム



注：略語説明 TPL/II(Test Program Language/II)
SPL(Software Production Language)
PCL(Process Control Language)

図2 MTS，STSとTPL/II MTS，STSは，TPL/IIで書かれたテストプログラムを，被テストプログラムと結合してテストを実行させるモニタの役割を果たす。

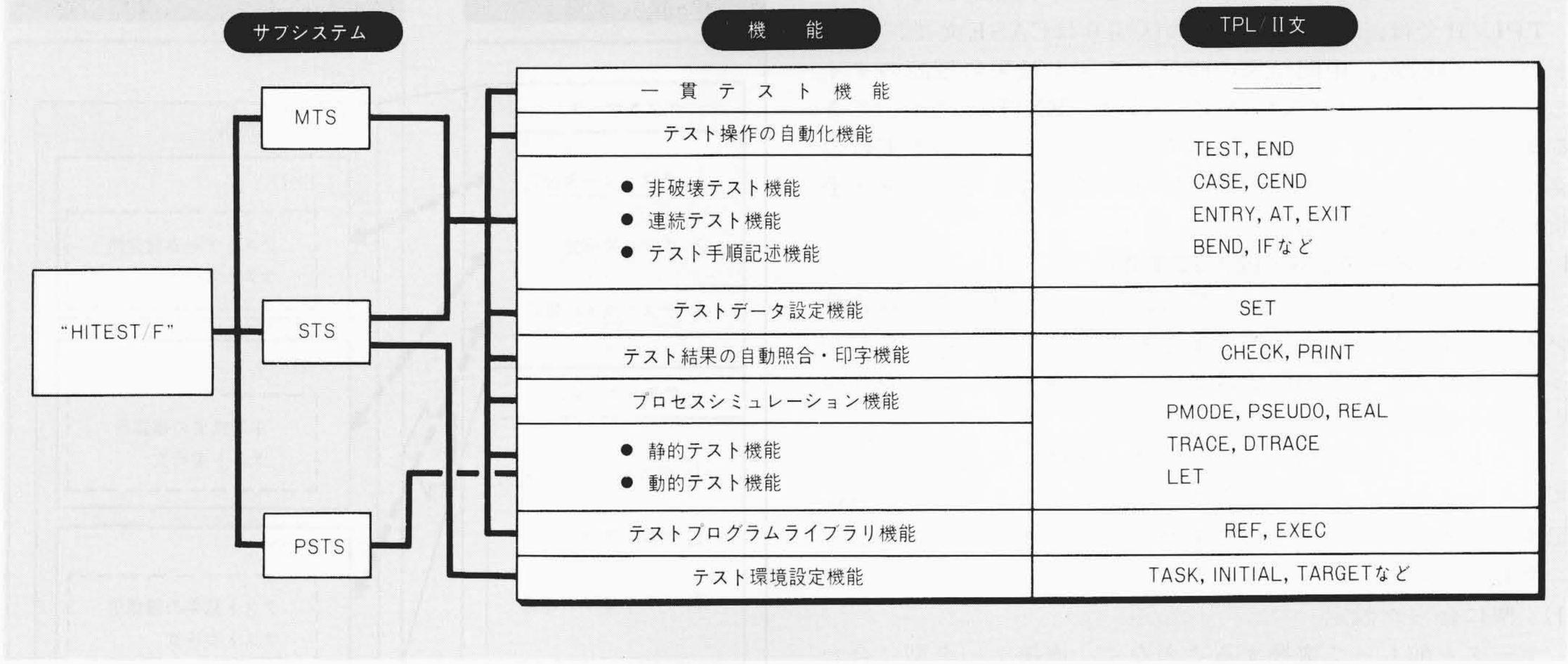


図3　“HITEST/F”の機能　　これらの機能を利用して，効率良くかつ信頼性高くテストを行なうことができる。

テストからモジュールテストにもどっての機能確認テストが非常に簡単にできる。

(3)　すべてのテストプログラムが再使用可能となるため，テストプログラムの変更によるテスト品質の低下を防止することができる。

TPL/IIではこれらの機能を実現するために，モジュールテストとシステムテストの間で言語としての互換性を保ちながら，各々のフェーズで特有なテスト機能によってフェーズごとの特異性を吸収するようにしている。一貫テスト機能の概念を**図4**に示す。

4.2　テスト操作の自動化機能

(1)　非破壊テスト機能

従来のテスト作業では，テストデータの設定や結果の確認のためのテスト用ステートメントを，被テストプログラムの中に埋め込んで(被テストプログラムをいったん破壊して)テストを実行させることも多かった。しかしこういった方法では，

(a)　テスト用ステートメントの挿入・削除に伴う操作ミスが起きやすい。

(b)　テスト用ステートメントを入れたままにしておくと，被テストプログラム自体のドキュメント性が悪くなる。

(c)　テスト用ステートメントを削除すると，被テストプログラムに修正が生じたあとの再テストが困難になる。

といった問題を生ずる。

したがって，この問題を解決するために，被テストプログラム中にテスト用ステートメントを混在させないことが必要となる。“HITEST/F”ではテスト用ステートメントをテストプログラムとして，被テストプログラムから独立させることによってこの非破壊テスト機能を実現している(**図2**)。

(2)　連続テスト機能

1回のテストランで1テストケースしか実行できないとすれば，あまりにも効率が悪い。そのために，被テストプログラムの中にテスト用ステートメントを挿入し，複数回ループさせることも可能であるが，結果として被テストプログラムを破壊することになり好ましくない。

連続テスト機能とは，被テストプログラムには手を加えず，テスト仕様に規定した複数のテストケースを連続して行なわせるようにした機能であり，テスト操作の自動化には不可欠のものである。この機能によってコンピュータ使用時間の節約が図れることはもちろん，1回の実行ですべてのテストケースに含まれる不良が発見できるので，不良原因の究明・対策が非常に能率良く行なえるようになる。

(3)　テスト手順記述機能

いかに必要十分なテスト仕様を立案したとしても，その仕様がテスト実行に正しく反映されなければ意味がない。

テスト手順記述機能はそのために設けた機能で，テスト仕様に示された手順どおりに，自然な形でテストプログラムが書けるようにするものである。この機能により，テスト仕様作成からテストプログラム作成への連続性を保つことができる。また，テストプログラムだけでテストの手順が一目で分かるので，テストプログラム自身を一つの「テスト仕様ドキ

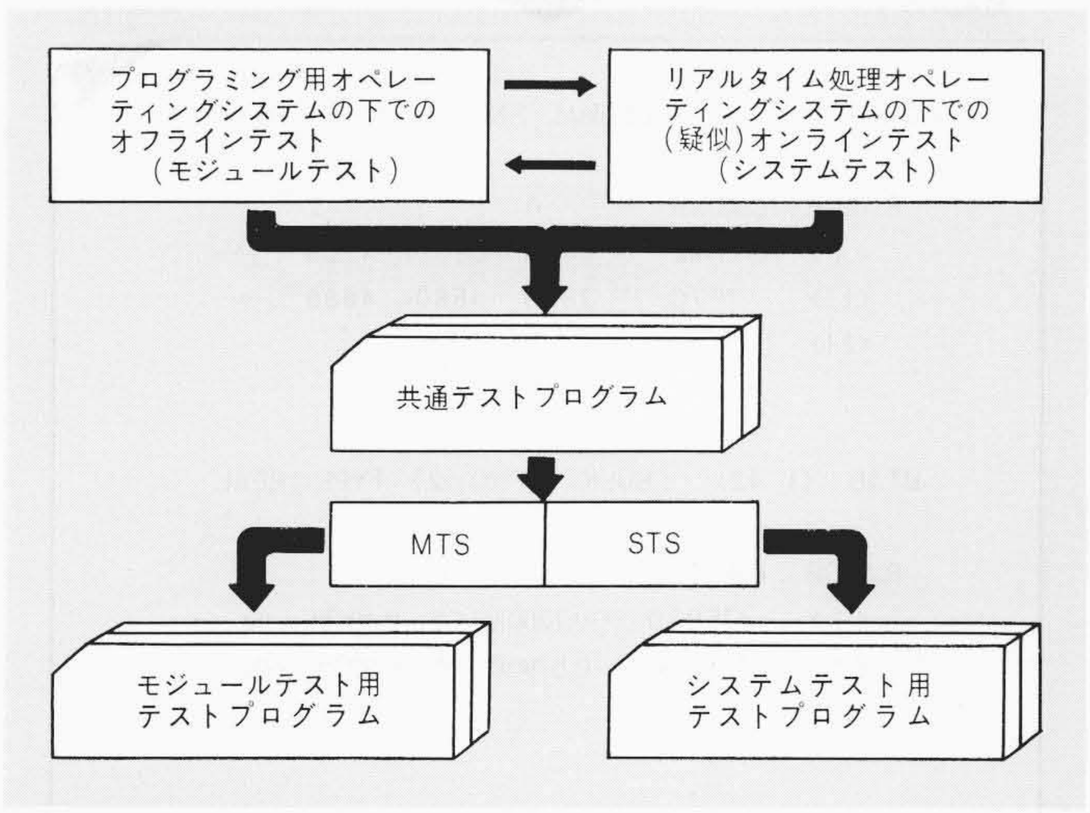


図4　一貫テスト機能　　モジュールテスト用のテストプログラム，システムテスト用のテストプログラムを別々に作成する必要はない。TPL/IIで共通のテストプログラムを作成しておけば，各々のテストプログラムはMTS，STSによって自動生成される。

ュメント」として扱うことができるという利点もある。

TPL/IIでは，テストケースの区切りはCASE文で，テストデータの設定，中間結果の確認，テスト結果の確認のタイミングは，それぞれENTRY文，AT文，EXIT文で指示できるようになっている。またテスト結果により，テスト実行の流れを細かく制御できるようにIF文を設けている。テスト手順記述機能を**図5**に示す。

4.3 テストデータ設定機能(SET文)

テストデータに誤りがあっては，被テストプログラム自身が正しくても，テスト結果が正しいものにならない。したがって，テストデータの作成はできるだけ簡単に，かつ誤りなく行なう必要がある。

テストデータ設定機能とは，テスト仕様に規定した入力テストデータを，シンボリックなレベルでテストプログラムに記述できるようにしたものであり，次のようなバリエーションをもっている。

(1) 型に合った設定

データを前もって変換することなく，直接データ型に合わせた設定が可能である(整定数，実定数，文字定数など)。

(2) 連続設定

指定した変数のアドレスから連続的にN語のデータ設定が可能である。

(3) 繰返し設定

同一のデータの大量設定が可能である。

SET文の例

```
SET  A＝2(“2020”, 10, 5(‘ABC’))
(変数Aの先頭から  2×(2＋5×2)＝24語のデータが設定される。)
```

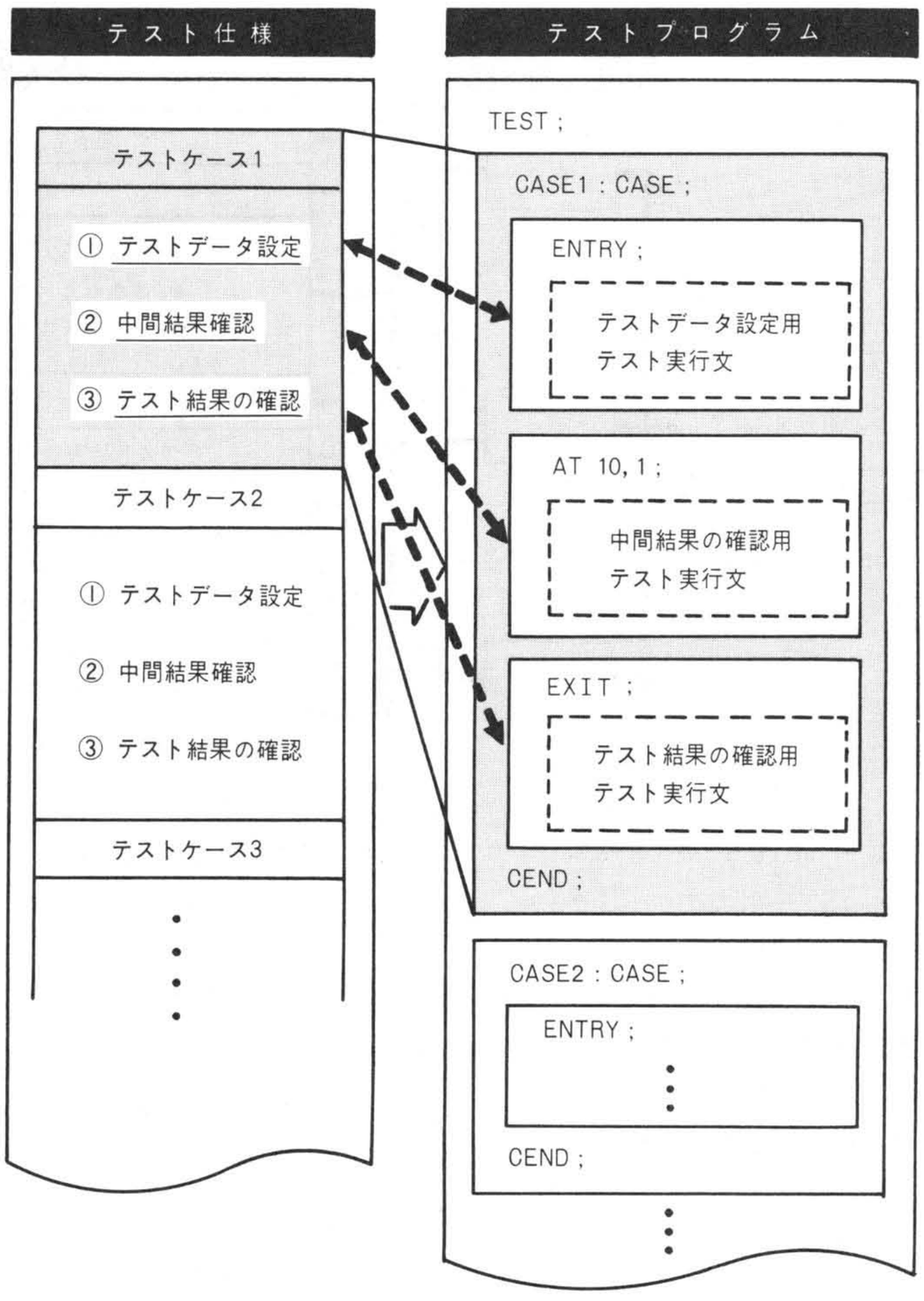


図5 テスト手順記述機能 テスト仕様に沿って，テストプログラムを自然な形で記述できる。したがって，テストプログラム自身を一つの「テスト仕様ドキュメント」として扱うことができる。

4.4 テスト結果の自動照合，印字機能(CHECK文，PRINT文)

テストデータに誤りがなく，テスト実行が正しく行なわれたとしても，テスト結果に確認ミスがあっては，行なったテスト自身がむだになってしまう。したがって，テスト結果はできるだけ見やすいドキュメントとすることが必要であり，可能ならば予想値との自動照合ができるようにすることが，テスト結果の良否の判定上望ましい。

テスト結果の印字は，TPL/IIのPRINT文によって行なわれる(**図6**)。PRINT文は，テスト結果の確認を誤りなく，かつ効率良く行なえるようにするため，

(1) 印字型指定

ユーザーが指定した任意の印字型(16進型，10進型，実数型，文字型)でテスト結果を印字できる。

(2) 印字フォーマットコントロール

テスト結果を編集し，必要なデータを一つのまとまったドキュメントとして出力する。

などの豊富なバリエーションをもっている。

テスト結果の自動照合は，TPL/IIのCHECK文を用いて行なう。CHECK文の使用例を**図7**に示す。CHECK文もSET文と同様なバリエーションをもっており〔型に合わせたチェック，連続チェック，繰返しチェックなど〕，効率的な照合が可能である。照合結果は簡潔なメッセージの形で印字出力されるので，確認ミスを犯すことはほとんどない。また照合結果の‘OK’，‘ERROR’のメッセージだけでテスト結果の良否が判定できるので，(テストプログラムの作成には多少時間がかかっても)テスト結果確認の時間が大幅に削減できる。

GTAB, BTABの二つのエリアをテスト結果として印字出力する。

TPL/IIによる記述

```
PRINT GTAB <1-100-10> ;
PRINT E : BTAB <*-3> ;
```

結果の印字

```
GTAB <1-100> (GLOBAL BNO=0) TYPE=HEXA

R.ADDR   C.ADDR        0      1      2
  <1>    /7F72   “ 440A   0000   47C9  ……
 <11>    /7F7C   “ 33FB   A880   4685  ……
 <21>
  ⋮

BTAB <1-42> (BULK   FNO=2) TYPE=REAL

R.ADDR   B.ADDR          0             1
  <1>    /47E8@0   0.1050E+02   0.1005E+03  …
  <7>    /47E8@6  −0.5790E+05   0.3235E+02  …
 <13>
  ⋮
```

図6 テスト結果の印字機能 PRINT文によって，大量のテスト結果データの確認を誤りなく，かつ効率良く行なうことができる。

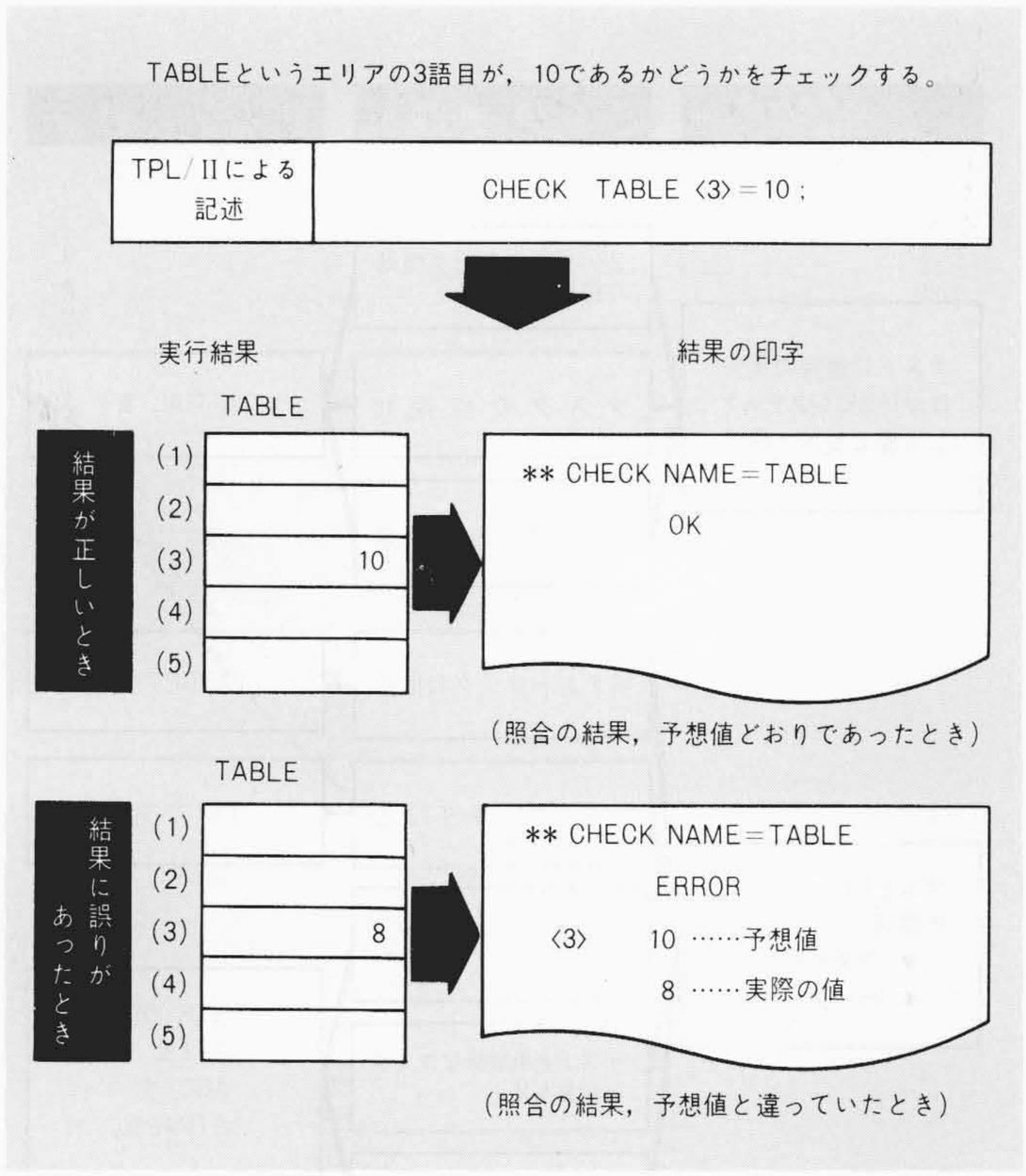


図7　テスト結果の自動照合機能　OK，ERRORのメッセージだけでテスト結果の良否が判定できるので，テスト結果確認の時間が少なくて済む。

4.5　プロセスシミュレーション機能

プロセスシミュレーション機能は，プロセス入出力のための標準サブシステムであるPDCS(Process Data Control System：基本プロセス入出力システム)[4]の中に，実際に入出力アクセスを行なうモード(実モード)と行なわないモード(模擬モード)とを切り替える機構を設けることによって実現している(**図8**)。

(1)　静的テスト機能

従来，プロセスとコンピュータを結合する前段階では，プロセスの代わりに，スイッチやランプをもったシミュレーション用パネルを用いて，プロセスシミュレーションを行なっていた。しかしこの場合，人間によるパネルスイッチの操作，ランプの点・消灯の目視確認が必要で，操作ミスや確認ミスが発生しやすかった。また「正確な再現テストが困難。」，「テスト結果が記録として残らない。」という問題もあった。

MTS，STSではこういった静的なテストを，シミュレーション用パネルなしに行なえるようにしている。すなわち，模擬モードでは，あらかじめテストプログラム(LET文)からプロセス入力データを模擬データファイル経由で被テストプログラムに渡すことによって，プロセス入力の模擬を行なう。プロセス出力は実際の出力を行なう代わりに，出力データをプリンタに印字する(TRACE文)。

(2)　動的テスト機能

本機能はプロセスのダイナミックな動きそのものをシミュレーションすることによって，疑似オンラインテストを行なえるようにしたものである。このようにすれば，性能を含めての品質確認が早い段階でレベル高く行なえる。このためには，プロセスの動作を模擬するプロセスモデルが必要である。このプロセスモデルは，ユーザープログラムから出力された制御用プロセス出力を受けて模擬プロセスとして動作し，その結果を疑似プロセス入力としてユーザープログラムに返す機能をもつ。

PSTSでは，ユーザーがこのプロセスモデルを容易に作成できるように，プロセスモデル記述言語PMDLとして各種のマクロ命令を用意している。

4.6　テストプログラムライブラリ機能

テストデータの設定やテスト結果の出力などのテスト手続きの中には，総合テスト用のテストデータのように，複数のユーザープログラムの間で共通なものも多い。これら共通なテスト手続きを1箇所にまとめ，必要に応じて使用できるようにしておけば，テストプログラムを重複して作成するむだがなくなる。また，テストプログラムに変更が生じたときの修正も容易となる。“HITEST/F”ではテストプログラムでも，通常のプログラムと同様な「ライブラリ」の概念を導入した(**図9**)。

本機能によって，モジュールテストからシステムテストまで，同一のテストデータが複数の利用者間で共通に繰返し利用できるので，

(1)　テストデータ作成量の大幅削減
(2)　コンピュータ使用時間の短縮
(3)　再テストの容易化
(4)　テストデータの一元管理による信頼性の向上

を図ることができる。

4.7　テスト環境設定機能

システムテストでは，モジュールテストを行なうときと違ってテストされる側のシステムに様々な不確定要素がある。

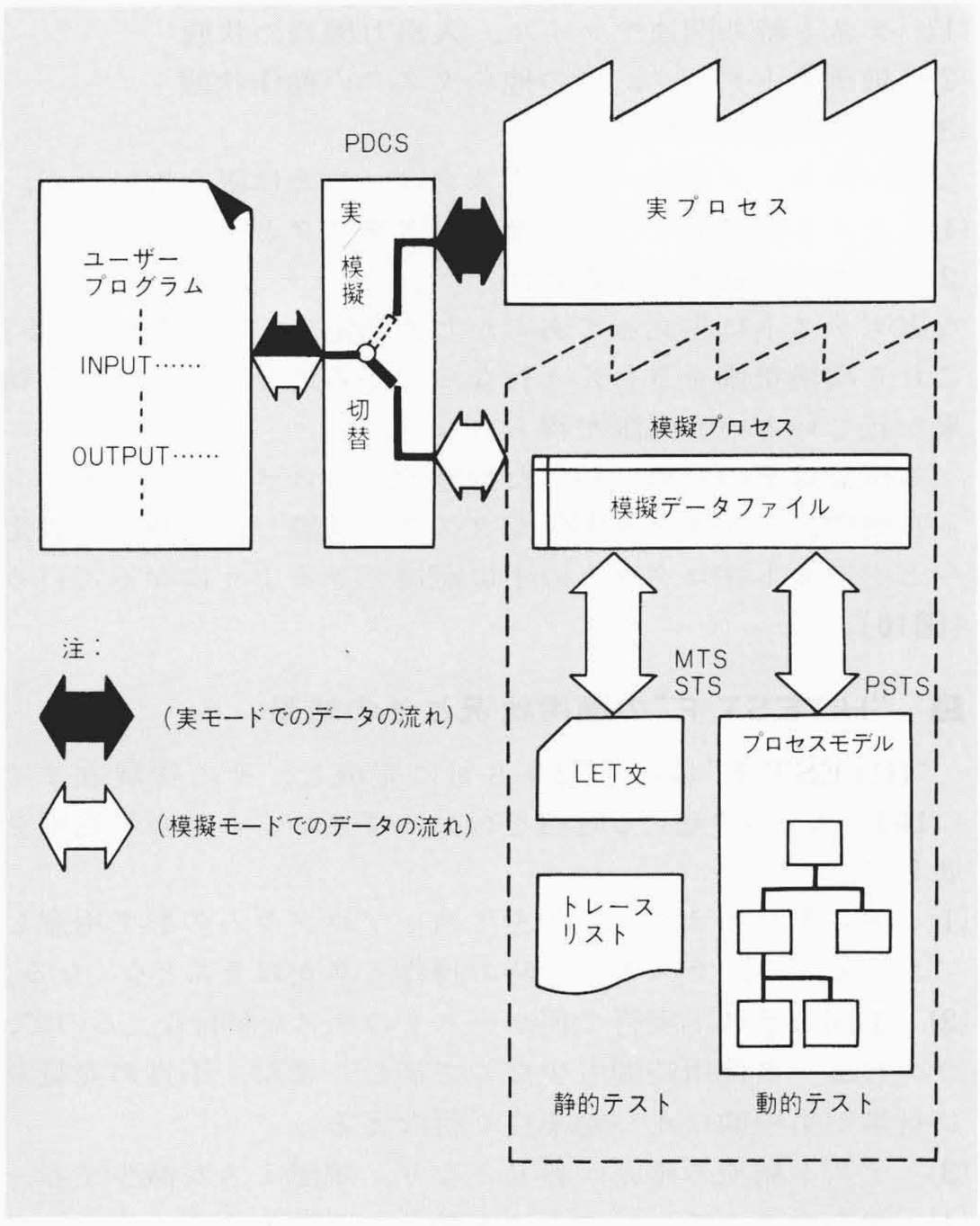


図8　プロセスシミュレーション機能　プロセスシミュレーションは，PDCS(基本プロセス入出力システム)の中にシミュレーション用の切換スイッチを設けることにより実現されている。MTS，STSによる静的テスト，PSTSによる動的テストが可能である。

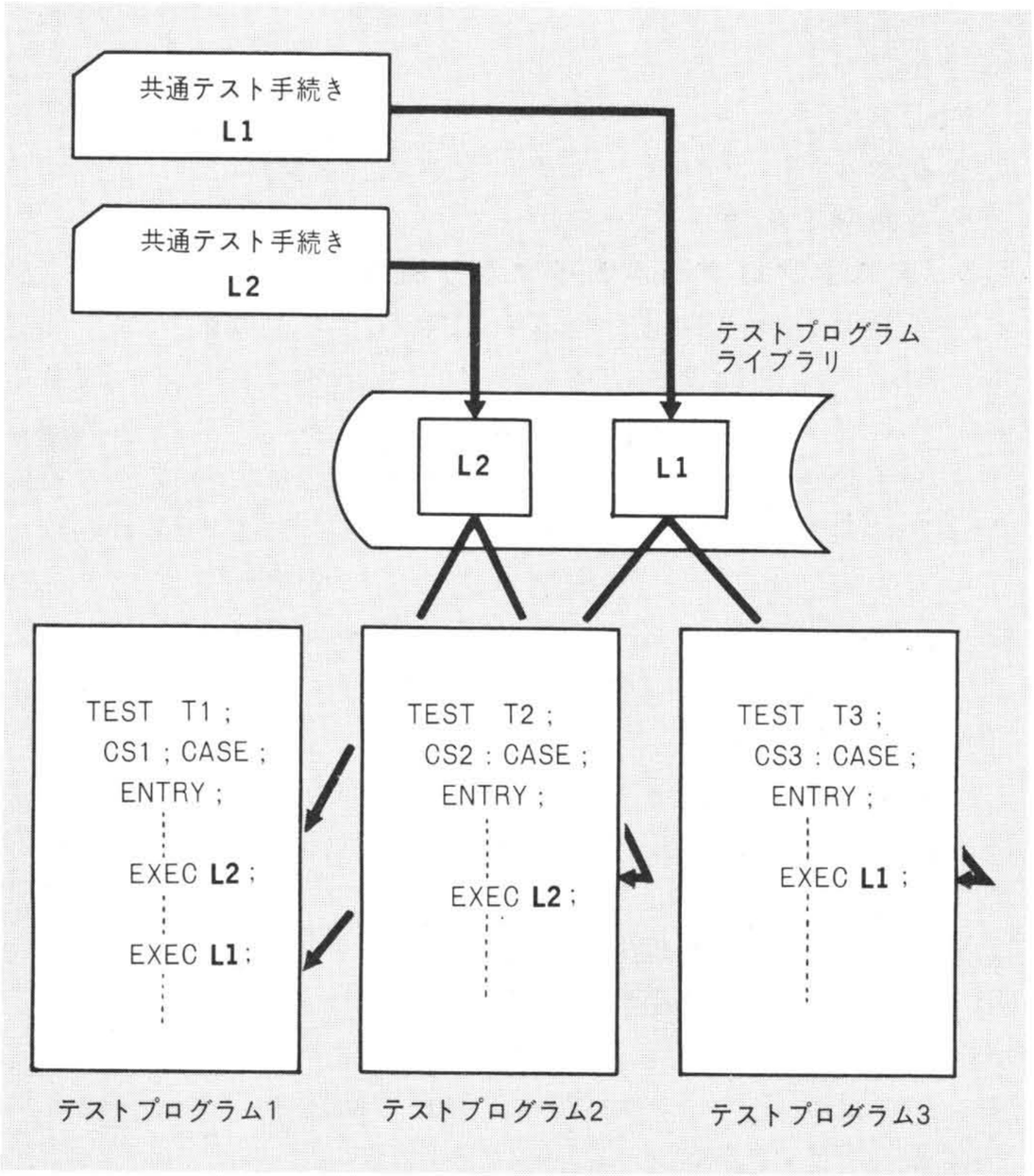


図9 テストプログラムライブラリ機能 テスト間で共通なテスト手続き(例えば，グローバル，バルクエリアの初期化など)は，テストプログラムライブラリに格納しておけば，各々のテストプログラムで任意に参照できる。

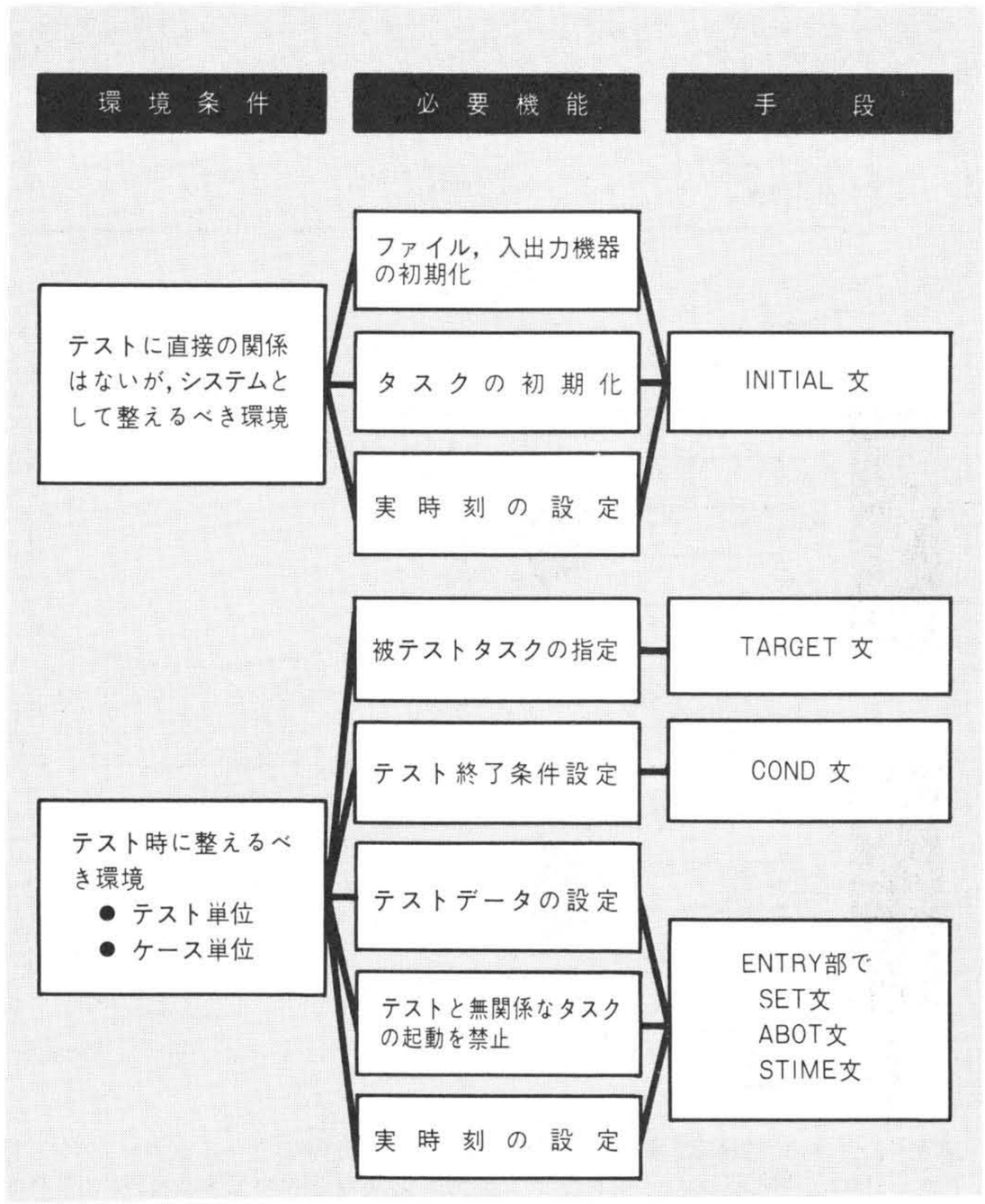


図10 テスト環境設定機能 システムテストでは，テストに先立って上記のようなテスト環境整備を行なっておくことにより，テスト結果の正しさを保証することができる。

例えば，
(1) テスト時の関連ファイル，入出力機器の状態
(2) 被テストタスク，その他のタスクの動作状態
(3) 実時刻の設定値
などである。更に被テストタスクは一つとは限らないため，
(1) テストに際して実行させるべきタスクとその順序
(2) テストを終了させる条件
などもテストに先立ってあらかじめ指定しておく必要がある。これら環境整備をきちんと行なってからでないと，テスト結果が正しいという保証が得られない。

本機能はそのために設けた機能で，システム全体のイニシャル処理，テストと無関係なタスクの切離し，実時刻の設定などをテストプログラムの中に記述できるようになっている(図10)。

5 “HITEST/F”の適用状況とその結果

“HITEST/F”は昭和53年8月に完成し，その後現在までに100システムを超える適用を行なってきた。これらの適用を通じて，
(1) あらかじめテスト手順をテストプログラムの形で用意しておけるので，テスト実行時の操作ミスがほとんどなくなる。
(2) 1回のテスト実行で何ケースものテストが行なえるので，コンピュータ使用時間も少なくて済む。また，不良の発見及び対策が集中的にかつ効率良く行なえる。
(3) テスト結果の確認が容易となり，確認ミスが減少する。
(4) 総合テストで，プロセスの動きと同期したダイナミックな機能テストが十分に行なえる。また，システムの性能上の問題を早期に摘出し対策を立てることができる。
といった良好な結果を得ている。

6 結 言

HIDIC 80シリーズの計算機制御用ソフトウェアの機能一貫テストシステムとして開発した“HITEST/F”について，その開発思想，方針，機能上の特長及び適用状況について述べた。現在“HITEST/F”は，計算機制御用ソフトウェアのテストツールとして必要不可欠のものとなっており，所期のねらいを十分達成している。

今後は，更に適用経験を重ね，使いやすさの向上を図るとともに，新しい技術を取り入れて，テストシステムとしてより完全なものに近づけていく必要があると考えている。

このテストシステムが，ユーザーのソフトウェア開発の高信頼化と生産性向上に寄与でき，また世界のソフトウェア工学の進歩に対して，いささかでも貢献できるとすればこれに勝る喜びはない。

終わりに，この“HITEST/F”の開発と適用に御協力をいただいた関係各位に対し，深く感謝の意を表わす次第である。

参考文献

1) 林，外：HIDIC 80シリーズ ソフトウェア開発支援システム，日立評論，61，603～606(昭54-8)
2) 林，外：制御用トップダウン・ストラクチャードプログラミング言語—SPL—，日立評論，60，235～240(昭53-3)
3) 迫田，外：プログラムテストシステムTPL—テストプログラム記述言語—，情報処理学会第15回全国大会(昭49-12)
4) 平井，外：HIDIC 80シリーズ基本制御ソフトウェア，日立評論，61，599～602(昭54-8)