U.D.C. 681.32.069:681.32.06.001.41

効率よいプログラム テストを実現するプログラム テストツール“HMTS”

Efficient Computer Program Testing Package: Hitachi Module Testing System

松本良治* *Matsumoto Yoshiharu*
鳥居哲郎** *Torii Tetsurô*

ＨＭＴＳは，コンピュータ プログラムのテストを効率よく行なうための支援プログラムである。ＨＭＴＳを使用することにより，プログラムをモジュールごとにテストできるだけでなく，未完成モジュールをも含めたプログラム内モジュールの動作テストを，簡単な制御文を用いて行なうことが可能となる。また，どのようなテストを，どのようにして行なうかのテスト条件は，被テストプログラムを変更しなくても設定でき，テスト結果をシミュレートしながら，テスト条件を自動的に変更できるので，効率の良いプログラム テストが行なえる。

本稿は，ＨＭＴＳの特徴と，その使い方について紹介するものである。

1 緒　言

作成したプログラムが，決められた仕様どおりに動作するかどうかのテストは，コンピュータ プログラムの開発において非常に重要な作業になる。プログラム テストをいかに効率よく行なうかどうかが，プログラム開発の生産性に大きな影響を与えるからである。Hitachi Module Testing System（以下，ＨＭＴＳと略す）は，このような状況に応ずるため開発したプログラム テスト支援のためのプログラムで，HITAC M-170/M-180のVirtual-Storage Operating System 1，2及び3（以下，ＶＯＳ１，同２及び同３と略す）の下で動作する。ＨＭＴＳを使用することにより，プログラムをモジュールごとに確実にテストするだけでなく，まだ完成されていないモジュールをも含めたプログラム内モジュール間の動作テストを，簡単な手段で行なえる。すなわち，テストのために特別のプログラムを別に用意する必要がなく，ＰＬ/1, COBOL, FORTRANなど，どのような言語で記述されたプログラムに対しても，同様な制御文を与えるだけでテストが行なえる。またどのようなテストを，どのようにして行なうかなどのテスト条件は，被テスト プログラムを変更しなくても設定でき，テスト結果をシミュレートしながらテスト条件を自動的に変更することができるので，効率よいプログラム テストを実現することができる。

2 プログラム テストの重要性

開発したプログラムが，問題なく動作するかどうかを確認することは，後々のバグの発生による影響の大きさを考えれば極めて重要なことであるにもかかわらず，一般的なプログラム テストの方法がまだ確立されていないために，問題を難しいものにしている。通常，プログラムのテスト方法として，そのプログラムが実際に動作するであろう環境を想定して，必要なデータを与えてみるとか，あるいはプログラム自身の中に，あらかじめテスト機能をもたせておいてモニタリングする方法など，そのプログラムの性格に合うと思われる方法でテストが行なわれる。いずれの場合もテスト計画を立て，何をテストするかの項目の選択が行なわれるが，密度の濃いテストを行なおうとすればするほど，そのプログラムを開発したときに要したのと同程度の労力が必要になる。仮に相当の労力をかけて，密度の濃いテスト項目を設定しても，それが直ちに品質の保証に結びつくことにはならない。更に大きな問題は，そうして実施したテスト方法が，多くの場合，他のプログラムのテストには不向きになっている。このようなことから最近では，テストしやすいプログラム，あるいはバグの発生しにくいプログラムを，最初から開発しようというプログラミング技術での改善が活発になってきた。

2.1 ＨＭＴＳの特徴

ＨＭＴＳは以上のような観点から，プログラムのテストをできるだけ確実に行ない，どのようなプログラムにも同じような方法でテストでき，そしてプログラム自身もある程度テストしやすい形になっている（すなわち，モジュール化されている）ことを前提にして，次のような特徴をもたせている。

(1) モジュール単位にテストする。但し，未完成のモジュールがあっても関連モジュールのテストを行なう。

(2) テストのために被テスト プログラムの変更を不要とする。

(3) どのような言語で記述されたプログラムに対しても，同じ使い方ができるようにする。

(4) 分かりやすく，使いやすくする。

3 ＨＭＴＳの構成

3.1 ＨＭＴＳと被テスト プログラムの関係

ＨＭＴＳを使用して，どのようにテストするかを被テストプログラムとの関係で示したのが**図1**である。被テスト プログラムは，これからテストしようとするプログラムを表わしている。ＨＭＴＳは基本的に四つの部分から構成されている。同図中で丸で囲んだ番号を付けてあるところが，実際のプログラム テストの前の準備段階になる。

(1) ①では，被テスト プログラムの属性を，ＨＭＴＳ連絡マクロを介してＨＭＴＳ側に与える。プログラムの属性とは，テストするモジュールの名前，モジュール間で受け渡しが行

* 日立ソフトウェアエンジニアリング株式会社　** 日立製作所ソフトウェア工場

なわれるパラメータの数，使用されている記述言語の種類などをいう。ＨＭＴＳ連絡マクロをアセンブルすることにより，ＨＭＴＳ連絡モジュールを生成する。

(2) ②は，被テスト プログラム自身のコンパイル，又はアセンブルを示す。手順としては，①と②が逆になってもよい。

(3) ③の段階で被テスト プログラムとＨＭＴＳをリンケージ エディタによって結合し，実行プログラムを作成する。ここで①で生成したＨＭＴＳ連絡モジュールと，ＨＭＴＳの制御部を合わせて取り込む。

①から③までては，具体的なテスト方法の指示は何も与えていない。どのようなデータを設定して，どのような条件でテストするかは①から③までで作成したプログラムを実行する時点，すなわちテスト実行時に指示する。このときのテスト指示に応じてＨＭＴＳ実行部を呼び出し，テストの流れを制御する。

3.2 テスト制御方式

ＨＭＴＳによるテストの制御方式は，基本的に対象となる被テスト プログラムがモジュール化されていることを前提にしている。モジュール化されているということは，プログラムの実行の流れが，モジュールからモジュールへと移っていくことである。従って，ＨＭＴＳはモジュール間インタフェースに一定の規則があることを前提としてテスト制御を行なう。**図2**はモジュールからモジュールへ制御が移るときに必要な基本情報を示している。Ｒ15，Ｒ14などの番号は，汎用レジスタを意味し，モジュールＭＯＤＡからモジュールＭＯＤＢ，ＭＯＤＣへ制御が移る場合，４個の汎用レジスタを使用してモジュール間インタフェースにおける基本情報を与える。汎用レジスタＲ15には，次に制御が移るモジュールＭＯＤＢのアドレスを，Ｒ14にはモジュールＭＯＤＢの処理が終了したときの戻りアドレスとしてＭＯＤＣのアドレスを，Ｒ13には各汎用レジスタの内容などを含む退避領域のアドレスを，そしてＲ１にはモジュールＭＯＤＢの処理に必要なデータをもつパラメータ リストのアドレスをそれぞれセットする。テスト実行時に，ＨＭＴＳに対してこれらの汎用レジスタの内容の変更を特に指示しない限り，ＨＭＴＳは以上の内容をもとに各モジュールのテストを制御する。すなわち，ＨＭＴＳがモジュールごとにテストを制御するとき，被テスト モジュールに必要なデータやテスト条件の設定はＨＭＴＳ側で管理するが，個々のモジュール内の処理についてはいっさい関与しない。

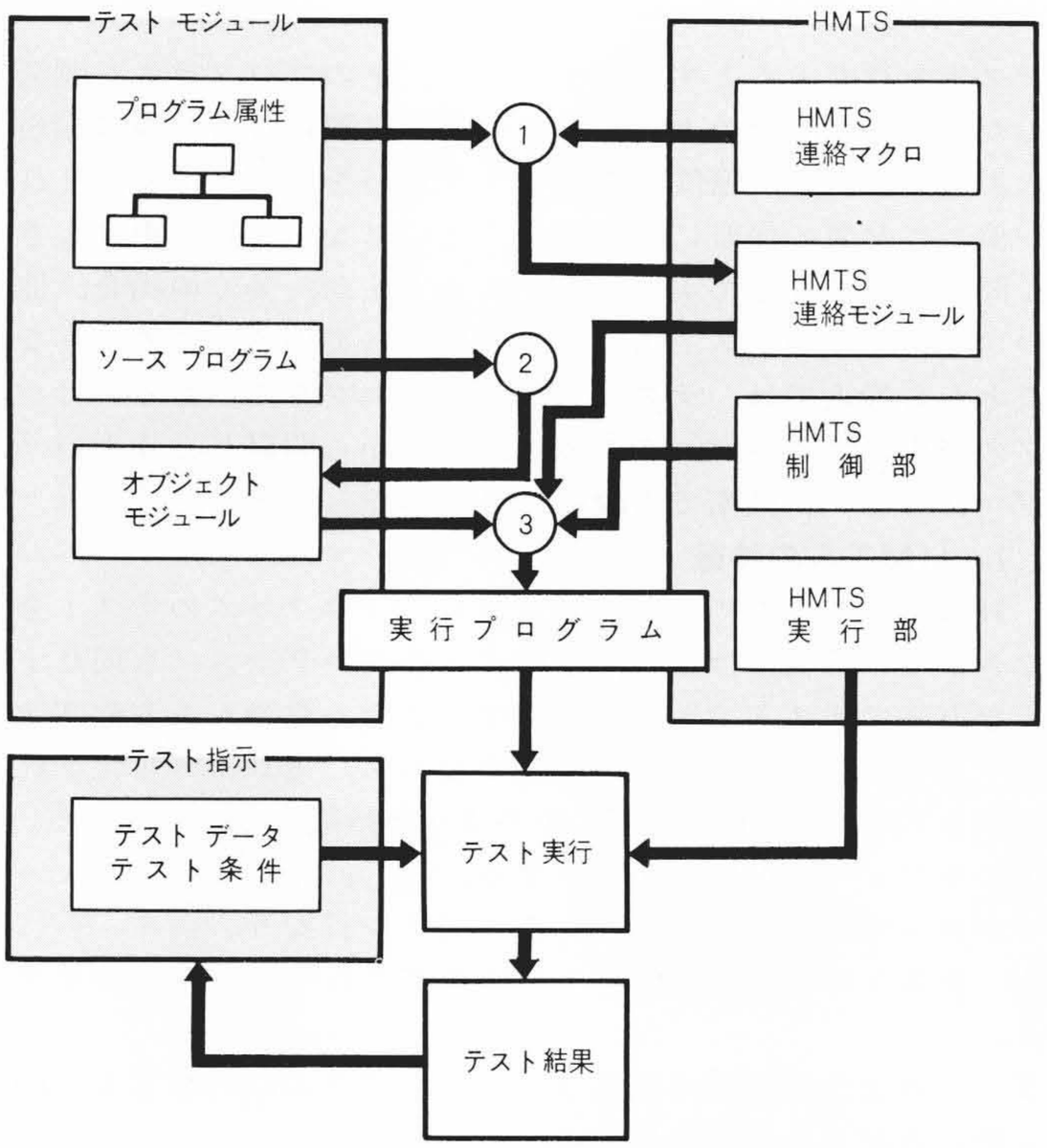

図１ HMTSの構成と処理方式 HMTSは基本的に四つの部分より構成されている。

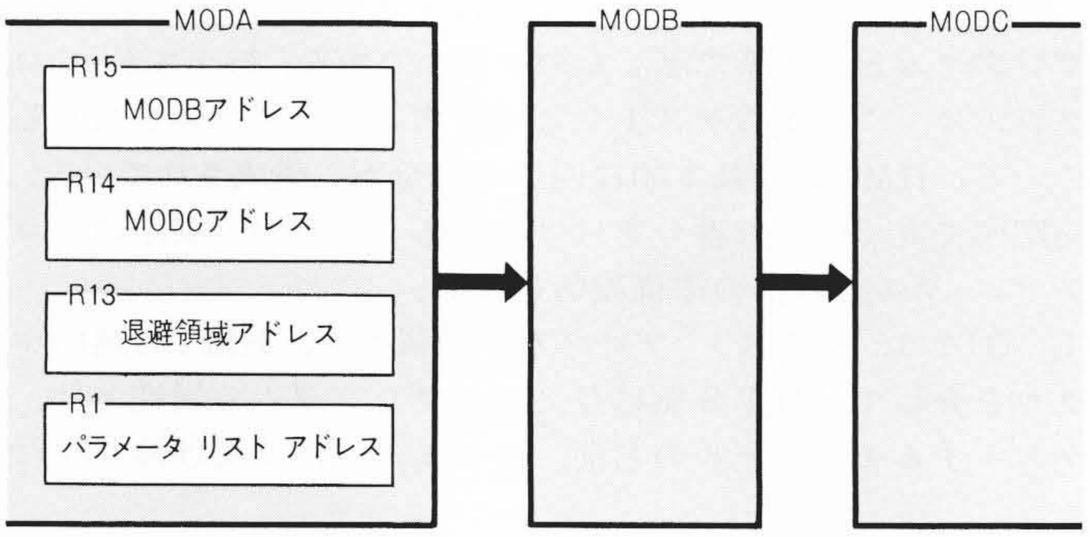

図２ モジュール間インタフェースにおける基本情報 MODAからMODBに制御が移るとき，四つの汎用レジスタに基本情報を与える。

4 ＨＭＴＳの機能

個々のモジュールに着目してテストを行なうとき，次のような要領でテスト計画を立てることになる。

(1) テスト モジュールに必要な入力データをセットする。

(2) 未完成モジュールの結果を，どのようにシミュレートするか決定する。

(3) テスト モジュールのプログラム不良を，どのように処置するかを決めておく。

(4) テスト結果の良否を判定する。

これらのテスト計画に対応させて，ＨＭＴＳがもつ具体的な機能について次に説明する。

4.1 入力データのセット

4.1.1 領域の確保

モジュールが動作するためには，動作するテスト モジュール以外に種々の領域が必要となる。ＨＭＴＳを使用して，テストを行なう際に必要なパラメータ領域，外部領域，そしてテスト モジュールとの関連を**図3**に示す。また**図4**はＣＯＢＯＬを用いた場合の領域確保の例を示している。

4.1.2 データのセット

テスト モジュールをテストするうえで必要な環境の設定，入力データのセットは，メモリ領域を利用することにより行なわれる。このメモリ領域を任意のデータで編集することにより，種々のテスト条件を作成する。ＨＭＴＳはこのようなデータのセットに対して，任意のデータを，任意の位置に，そして任意の長さでセットすることを可能にしている。代表的なデータの形式は，16進，文字，２進，10進，浮動小数点及びアドレスの各定数である。**図5**はデータの設定の例である。

4.1.3 テスト用データ ファイルの作成

テスト モジュール中で参照するデータ ファイルの個々のレコードも重要なテスト データであり，通常のテスト方法の場合は前もってデータ ファイルを作成しておくことになる。ＨＭＴＳでは，テスト実行時にこの種のテスト データを作成し，入・出力系統のモジュール テストを同時に行なえるよう

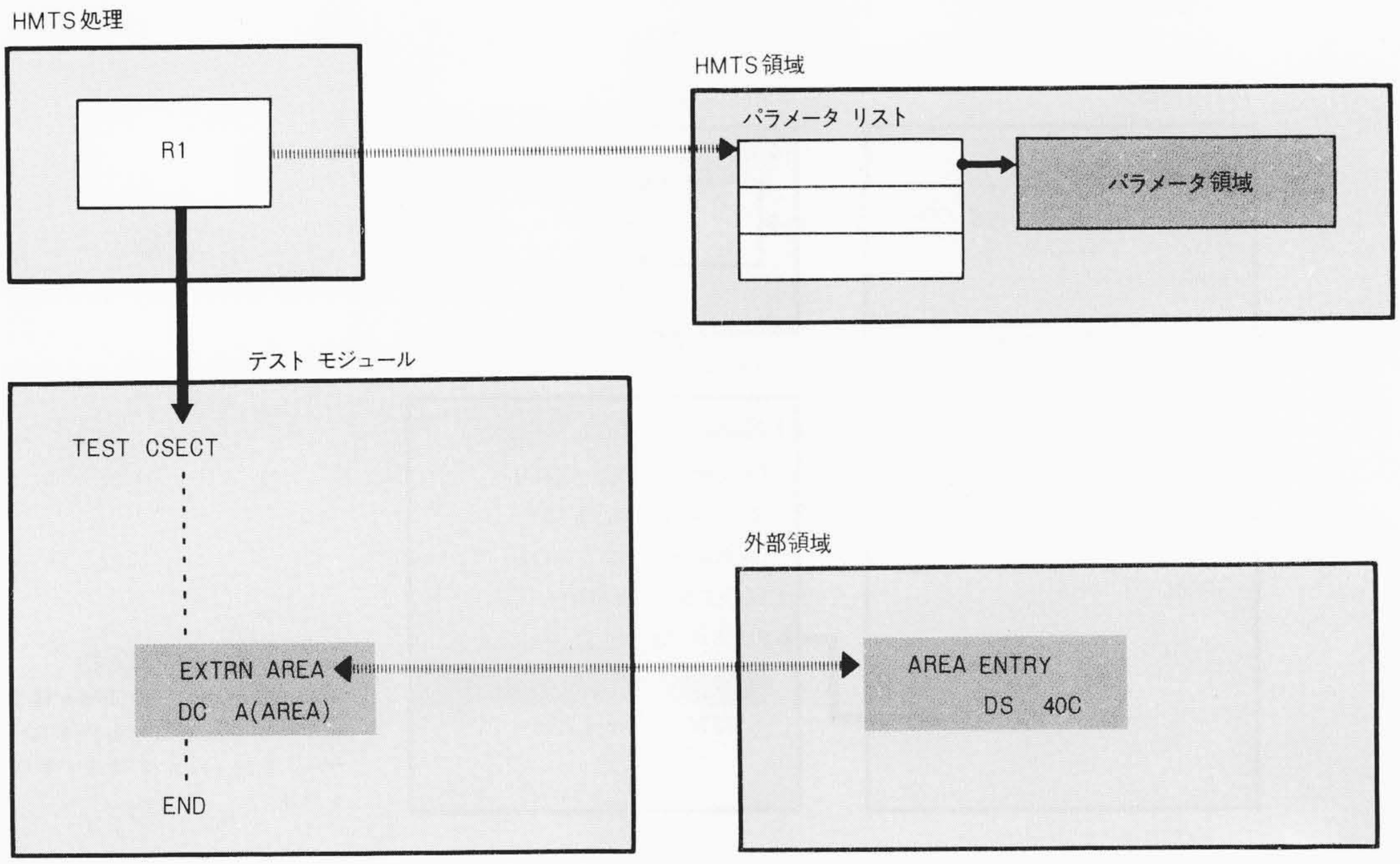

図3　パラメータ領域，外部領域とテスト モジュール　テスト モジュールが参照する領域は，HMTSが渡すパラメータ領域と外部領域とで構成されている。

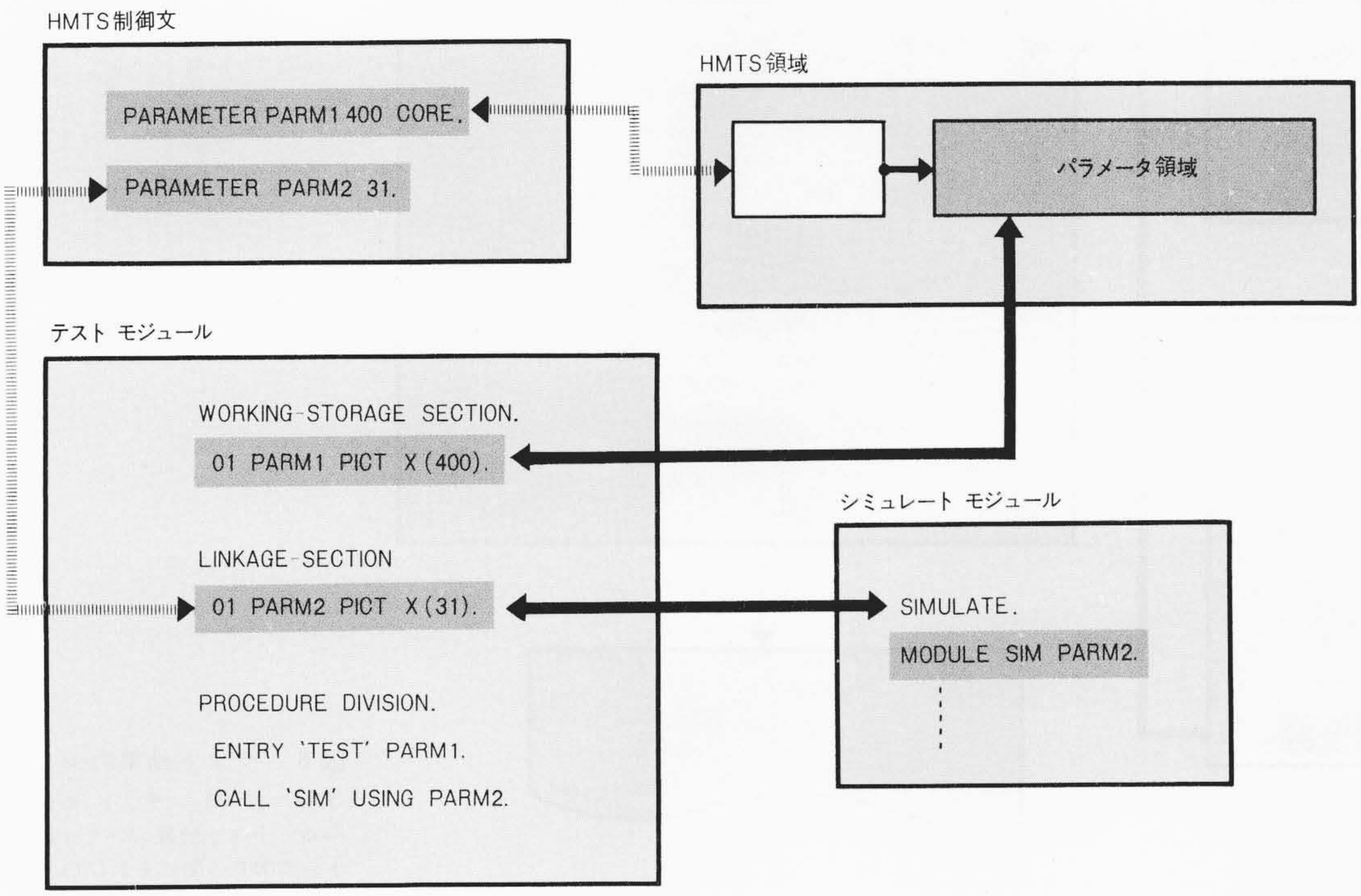

図4　HMTSとユーザー モジュールの参照領域　HMTS及びユーザー モジュールで確保した領域が，HMTS，ユーザー モジュール，未完成モジュールに渡される状態を示している。

にしてある。このことは，出力データ ファイルの作成に対しても同様で，テスト モジュール中で作成したデータ ファイルの内容を，同じテスト ランの中でラインプリンタなどに印刷表示する。

4.2　テスト結果のシミュレート

一般にプログラムの処理は，あるモジュールが他のモジュールを呼び出して処理を続けていく。あるモジュールをテストしようとしたとき，そのモジュールに関連する他のモジュールがすべて作成済みとは限らない。

このような場合，ＨＭＴＳは未完成モジュールのテスト結果をシミュレートする。ＨＭＴＳはモジュールの動作自体をシミュレートするのではなく，モジュールのテスト結果をシミュレートする。

ＨＭＴＳのテスト結果のシミュレートの方法には次のものがある。

(1)　常に同一結果をシミュレートする。
(2)　複数個の結果をサイクリックにシミュレートする。
(3)　未完成モジュールに渡される入力データに従ってシミュレートする。

図6はモジュールのテスト結果のシミュレート例である。この例では，ＴＥＳＴというモジュールがシミュレートされるモジュールで，入力データとしてＰＡＲＭ１，ＰＡＲＭ２が与えられる。シミュレートの方法は，入力データのＰＡＲＭ１が，文字列ＡＢＣと一致する場合にはＰＡＲＭ１とＰＡＲＭ２

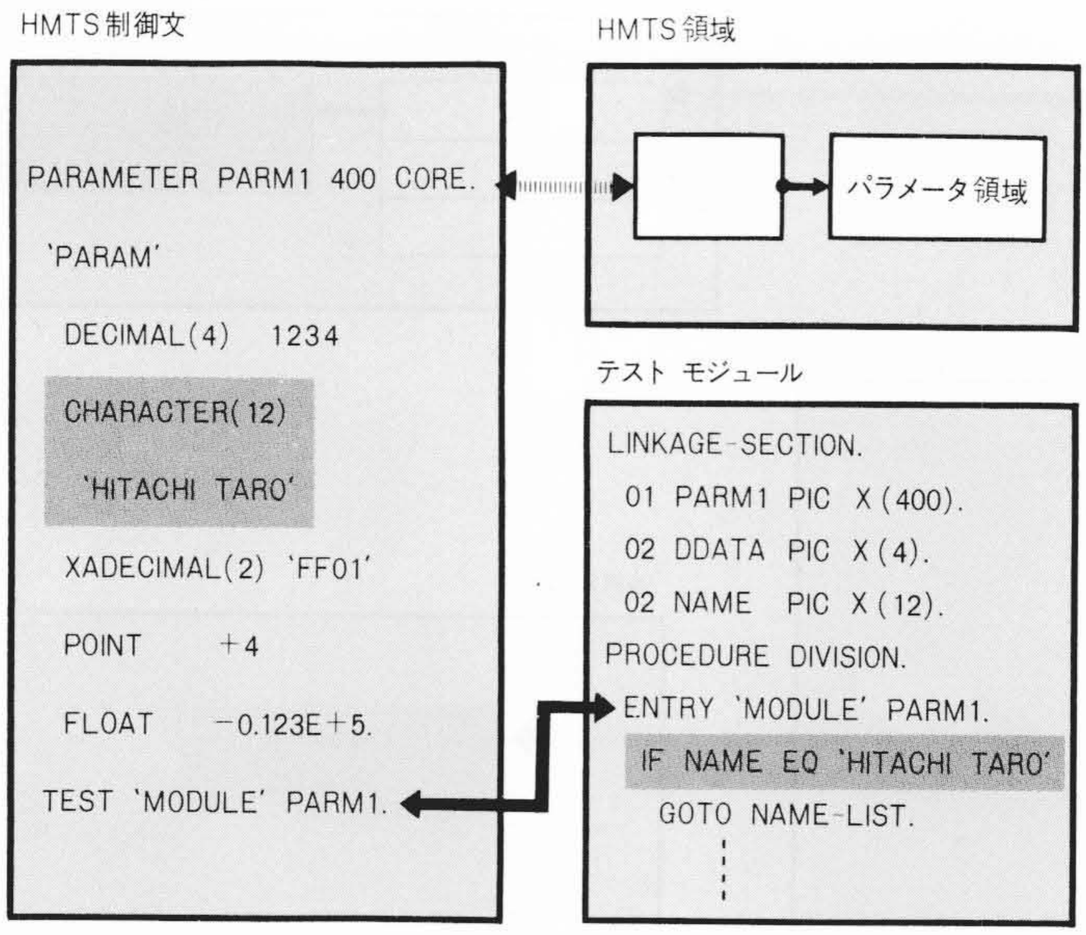

図5 データの設定
HMTSでパラメータ領域を確保しデータを設定したものをユーザー モジュールで参照する例を示す。

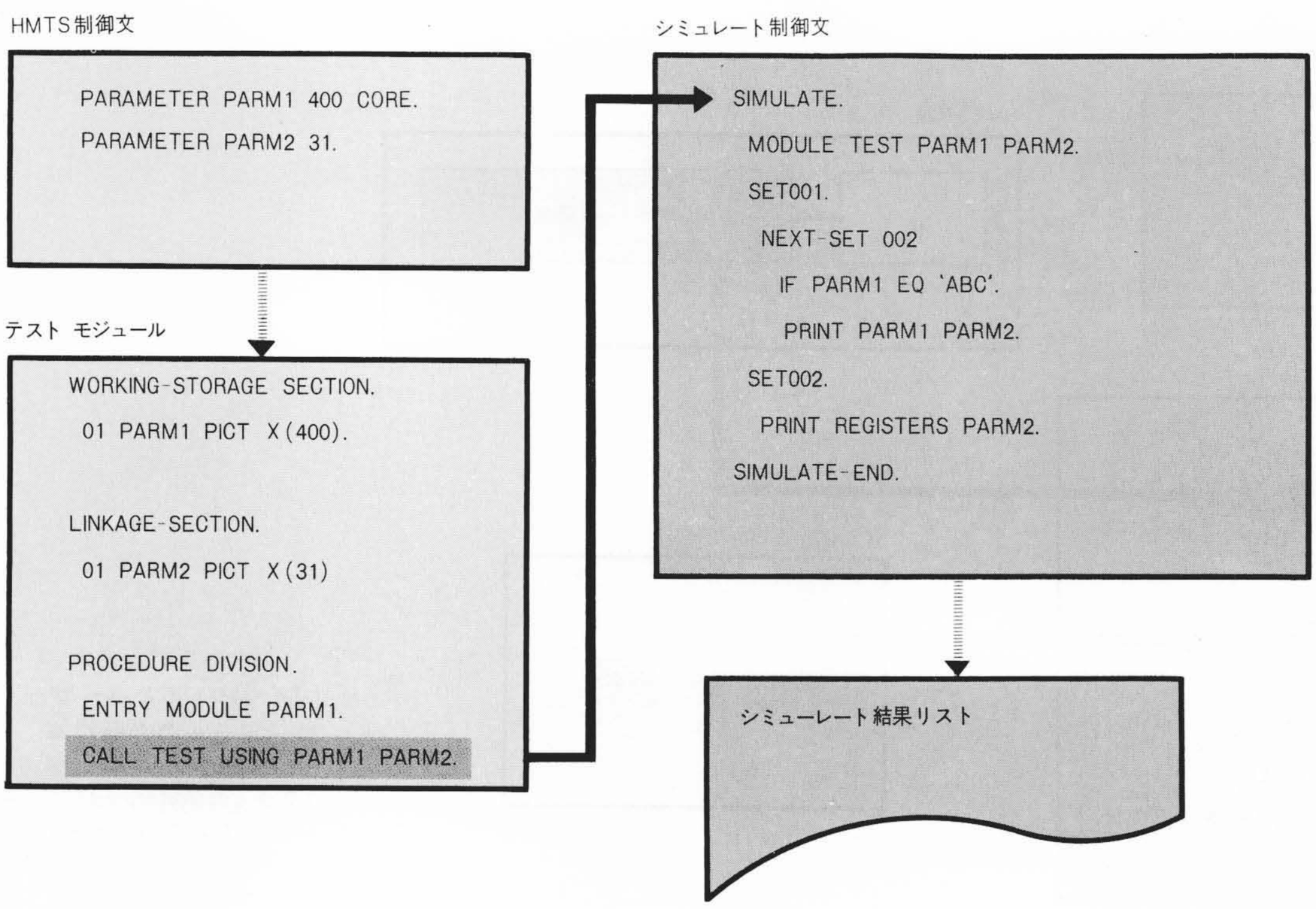

図6 テスト結果のシミュレート例 テスト モジュールとテスト結果のシミュレートの制御文の関係を示している。

を印刷し，一致しない場合は汎用レジスタとＰＡＲＭ１の内容を印刷することである。

4.3 プログラム不良の処置

モジュールをテストする際に，プログラムの不良によりプログラム エラーとなったり，無限ループに入ることがある。これらの不良のために，モジュールのテストが中断しないように，あらかじめ有効な手段を考えておく必要がある。

4.3.1 プログラム割込みの処置

不当命令やアドレス エラーによりプログラム割込みが発生すると，通常はテストを中断し，プログラム不良の原因を取り除かない限り，以降のテストは不可能となる。ＨＭＴＳはこのような状況をできるだけ回避し，テスト効率の向上を図るために，次のような機能を用意している。

(1) プログラム割込みに関する診断情報を出力し，次の命令を続けて実行する。

(2) 不当命令のデータを自動修正し，処理を続行する。

(3) 指定したプログラム割込みの回数に達するまで，処理を続行する。

4.3.2 ループ チェック

プログラムの論理不良による無限ループにより，コンピュータ時間のむだな使用を避けるため，ＨＭＴＳ側にあらかじめテストに要する実行時間を指示しておく。指定時間を超えると，プログラム不良として，必要な診断情報を出力してテストを中止する。

4.3.3 経過時間の測定

テストに要した実行時間をテストごとに報告する。これは，テスト段階から性能評価に関する資料を提供することになる。

4.4 テスト結果の判定

テスト結果の良否の判定は，通常多量の出力結果リストによって行なうことが多く，十分な検証を行なうだけでも相当の労力が必要になる。これを軽減するために，ＨＭＴＳは次のような機能を用意している。

4.4.1 結果の予想

ＨＭＴＳでは，あらかじめ予想されるテスト結果と，実際のテストによる結果とを比較し，結果が一致するときはその旨を表示し，不一致のときは予想結果と実際のテスト結果とを表示する。これにより，テスト結果の自動検証とテスト結果の標準化されたドキュメントの取得が可能となる。

4.4.2 結果の自動印刷と変更時印刷

テスト時に被テスト モジュールが使用したメモリ領域を，テスト終了時に自動的に印刷する機能がある。また指定したメモリ領域がテスト中に変更されたときだけ印刷する機能もある。これらの機能により，テスト結果の資料を必要最小限にできるばかりか，テスト結果の良否が容易に判定できることになる。

5 ＨＭＴＳの利点

以上述べてきたことから，ＨＭＴＳを使用することによって得られる利点として次のことが挙げられる。

(1) 個々のモジュールのテストを確実に行なうことができる。
(2) テストの進捗管理が正確になる。
(3) テスト作業の標準化が容易になる。
(4) コンピュータの使用効率が向上する。
(5) テストに要する費用を少なくすることができる。

6 結　　言

プログラムのテストをいかに効率よく行なうかは，テスト方法だけの問題ではない。被テスト プログラムがテストしやすくなっているかどうかにもよる。これは，ＨＭＴＳを使用するために特別のプログラミング上の配慮が必要という意味ではない。なぜなら，ＨＭＴＳ自身は，被テスト プログラム（厳密に言えばモジュール）の処理内容そのものについてはブラック ボックスとして扱うからである。ＨＭＴＳによって事前にテスト データや条件を設定してから，被テスト モジュールに制御が移るので，個々のモジュールの入・出力が論理的なプログラムの仕様に対応して，明解になっていることが望ましい訳である。

またプログラムのテスト作業以降の，プログラムの不良の検出とその修正の反映ということを考えれば，これらの作業手順の仕組みも重要な問題になる。すなわち，プログラムのテスト，不良の検出，プログラムの修正，そして再びテストという一連の作業が，できるだけ人手の介入を少なくする方法で行なえれば，相当の効率が期待できる。ＨＭＴＳがもつ機能から考えれば，修正とテストの作業の一貫した作業方法も不可能ではないからである。

我々は，プログラムのテストという観点から，今後ともプログラムの生産性向上，そして品質向上に寄与できる分野を追求していくと同時に，関係各位の御教示を仰ぎ，ＨＭＴＳ自身をよりいっそう良いものにしていきたいと念願する次第である。

論文抄録

仮想メモリ システムにおけるプログラムの局所性とその最大化

日立製作所　益田隆司・塩田博行

情報処理　16―12，1055（昭50-12）

仮想メモリ方式による計算機システムの性能を向上させるための方策が種々考察されている。仮想メモリ方式の実験，研究段階の時期から，ページ・サイズ，アドレス変換方式などのハードウェア面，オペレーティング システム内のタスク，メモリのスケジューリング方式などのソフトウェア面に関する報告が数多く発表されている。その実用化に伴い，利用方式の側面からの性能向上策が検討されている。これには，

(1) 開発済みのプログラムの再構成可能なプログラム単位での再構成法
(2) 仮想メモリ システム向きの数値計算法，特に行列の取扱い法
(3) 仮想メモリ システムのもとでプログラムを作成する際の留意すべき点

などが含まれる。

プログラムが一定時間内に利用する異なったページの集合をワーキング セットと呼び，時間に対してこの大きさが小さいプログラムを局所性の良いプログラムと呼ぶならば，局所性の良さがシステムの性能に大きな影響を与えることが知られている。

(1)~(3)はいずれもプログラムの局所性を上げることをねらっている。

本論文は，我々がこれまでに行なってきた(1)による試みを更に発展させたものである。(1)では，プログラムを再構成可能な最小要素であるセクタを単位として，実行時に時間的に近いところで利用されるセクタは，できるだけ同一のページに集めるように並べ替えることにより，プログラムの局所性を上げることを目的としている。そこでは，各セクタ内のメモリ使用効率には全く触れていない。本論文では，これを更に発展させ，各セクタごとのメモリ使用効率を定義し，プログラムの局所性をとらえる新しい考え方を提案した。そして，これに基づいて，開発済みのプログラムの各セクタのメモリ使用効率を計算し，メモリ使用効率の低いセクタについては，その構造の詳細な分析を行なっている。同時に，仮想メモリ システム下で動作するプログラムのメモリ使用効率に関する基本的な性質を幾つかの側面から検討した結果を報告する。更に，これらのメモリ使用効率に関するプログラムの動作特性の分析結果から，新しいプログラムを開発する場合に，その局所性を上げるために留意すべき点（セクタの大きさ，プログラムの書き方など）に対して一つの定量的な指針を与える。

本論文の内容は，特に繰り返し多く利用され，また，プロシージャ部のウエイトが大きいプログラム（システム プログラム，専用のアプリケーション プログラムなど）の局所性を上げるために有効である。また本論文で選んだ二つのモデル プログラムからの結果については，それらが経験あるシステム設計者により設計され，また，分析の対象とした約600個のセクタは，大勢のプログラマによって，設計，開発されていることから，その一般性は十分保証できると考えられる。