

## 16ビットマイコン68000用クロス型

### テストデバッグ支援システム HITS

中所武司、田中厚、岡本恵里、本田明德、黒崎徹  
(日立製作所システム開発研究所) (同左 戸塚工場)

#### 1. まえざき

ソフトウェアの生産性と信頼性の向上は、最近のマイクロコンピュータをはじめとする計算機応用分野の広がりに伴い、ますます重要な課題となってきた<sup>1)</sup>。なかでもテスト工程は、ソフトウェア開発費用の半分以上を占め、生産性向上に重要であるばかりでなく、品質保証に不可欠の重要工程である<sup>2)</sup>。しかしながら、テストデータを用いてプログラムを実行し、その結果を調べるというプログラムの検証方法は、実用面での<sup>3)</sup>ような問題がある。

- (1) 効果的テスト項目の効率的作成方法がない。
- (2) テストの準備、結果の確認作業に手間取る。

これらのテストの問題は、最近の16ビットマイクロコンピュータの普及に伴うソフトウェアの大規模化によって、この分野でも重要になってきた。そこで、我々は、これらのマイクロコンピュータ用ソフトウェアを汎用大型計算機を用いて効率良く開発するための支援システムの一つとして、16ビットマイクロコンピュータ68000用クロス型テストデバッグ支援システムHITS<sup>4,5)</sup> (Highly Interactive Testing-and-debugging System)を開発している。

従来、この種のツールとしては、データ値の表示や変更のための実行中断機能やトレース機能を主体としたデバッグツールが多い。汎用計算機の分野では、単体テストを支援するツールとして、MSP社のMTS<sup>6)</sup>やGE社のTPL<sup>7)</sup>などがあるが、前者は対象言語を限定しない方式などのため、また後者はFortranサブルーチンの引数に基づいたテストに限られることなどのため、先に述べたテストの問題はあまり解決していない。その他、プログラムのフロー解析や記号実行による誤りの自動検出、パス解析による構造テスト支援などのツールの研究も多々<sup>8)</sup>、デバッグも含め、テスト工程全体を系統的に支援するものは少ない。

HITSは、このような現状把握に基づいて設計されたシステムであり、16ビットマイコンHMCS 68000用のアセンブラおよび高級言語Super-PL/H<sup>9)</sup>で記述されたプログラムの機械語オブジェクトを汎用計算機HITAC Mシリーズ上でシミュレーション実行しながら、その効果的かつ効率的なテスト、デバッグを行うものである。本文では、その設計思想、機能、処理方式について述べる。

#### 2. 設計思想

本システムの設計は次のような基本方針に基づいて行った。

- (1) 汎用大型機によるクロスシステム化。
- (2) 系統的テストと効率的デバッグ支援機能の一元化。
- (3) テスト作業の極小化。
- (4) 高級言語とアセンブラの複数言語支援。

オイ項については、16ビットマイコン用大規模ソフトウェアの開発を実機で行

う場合、

- ・開発用大容量ファイルが使いにくい、
- ・多数の開発担当者間で実機の同時使用ができない、

などの欠点がある。そこで、本システムはクロスコンパイラ、クロスアセンブラなどと共に用いるクロス型システムとし、汎用計算機による一貫開発システムを実現する。そのシステム構成を図1に示す。

オ2項については、テストヤプログラムの誤りの検出を目的とするのに対し、デバグは誤りの原因を除くのが目的であることから、従来のデバグツールではテスト機能は不十分である一方、テストツールはデバグ機能を有しないなど、お互い独立に開発されてきた。しかしながら、言うまでもなくテストとデバグは表裏一体の関

係にあるものであり、誤りを検出したテスト手続きに一時的なデバグ用コマンドを付加しながらもう一度実行できれば、デバグ効率が良い。そこでHITSでは、特定のテスト手続きを会話型実行しながら、任意の位置での中断点指定、命令やデータのトレース、データ値の表示や変更などができるようにして、臨機応変なデバグを可能にした。

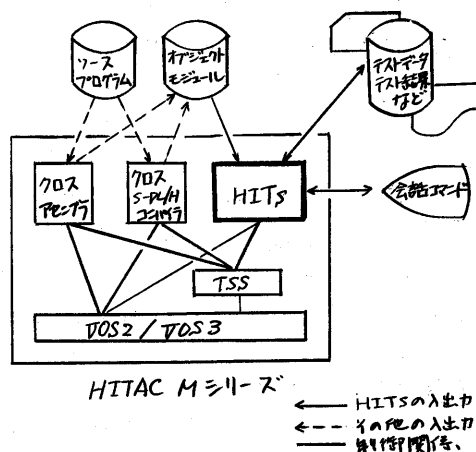
オ3項については、テスト支援ツールの主な目的の1つはテスト環境の作成やテスト結果の確認のための作業を軽減することなので、ツールの使用のために必要となる作業は極力少なくしなければならない。そのため、テスト手続き記述言語をコマンド言語として単純化すると共にテスト環境作成用コマンドを充実させた。各コマンドのオペランドも概テストプログラム記述言語と同一構文とし、利用者の負担を減じた。

オ4項については、本システムが主たる対象とするプログラムは、大部分をPL/Iサブセットを母体にマイコン特有の機能を追加した高級言語Super-PL/Hで記述しているが、一部をアセンブラで記述している。そこで、単体テストに限らず、連動テスト、システムテストにも一貫して本システムを使えるように、両言語を各々のソースレベルで支援する。

### 3. 系統的テスト

プログラムのテストは、まず個々のモジュールを対象にした単体テスト、次にそれらを幾つか結合した連動テスト、最後に全体を結合したシステムテストと段階的に行われる。特に単体/連動テストの方法としては、そのシステムの特성에応じて、下位モジュールから順に上位モジュールを結合しながら行うボトムアップテスト法やその逆に最上位モジュールから行うトップダウンテスト法などがある。テスト支援システムは各段階のテストをそのテスト法に依らず系統的に支援することを目指す。本文ではこの系統的テストを次の3要件で定義する。

- ・単体、連動、システムテストの各段階を支援する。



- ・各段階で同一のテストデータが使える。
- ・効果的なテスト項目作成を支援する。

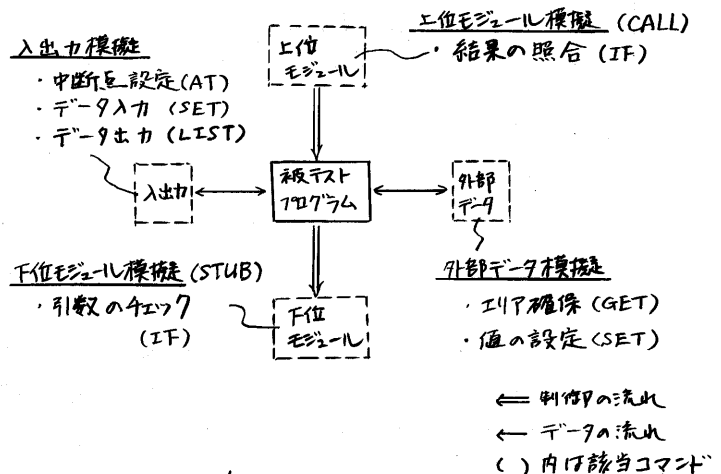
以下、(1)～(3)でこれらの要件について述べ、(4)でこれに基づくテスト手順を示す。

### (1) 単体/連動テスト

単体/連動テストは次の理由でソフトウェアの信頼性と生産性の向上に重要である。

- ・誤り修正費用は $\gamma$ の検出時期が遅いほど増大する。
- ・大規模ソフトウェアの構造テストの徹底。
- ・平行開発の促進による工程短縮。

さらに、最近定着しつつある構造化設計などのモジュール化技法との相乗効果も大きいと思われる。 $\gamma$ ここで、本システムでは、単体/連動テストを容易に行えるように、図2に示すような上位モジュール、下位モジュール、外部データ、入出力処理の模擬機能を備えた。



### (2) テスト手続き

単体/連動テストでは、被テストプログラムの入

図2 単体/連動テスト支援機能

力データの他に(1)で述べたような環境設定が必要である。 $\gamma$ ここでこれらをまとめた上に結果の確認を含めてテスト手続きとして記述する。そして、結合するモジュールを次第に増やしてテストする時には、不要になった部分を削除すれば前のテスト手続きを再利用できるようにする。

### (3) テスト充分性評価

プログラム仕様を満たすことを検証する方法としては、形式的に記述された仕様とプログラムの等価性を自動検証するのが理想的であるが、未だ実用には程遠い。 $\gamma$ ここで、実際には本文で対象としているようなテストデータを用いた動的な検証方法に頼らざるを得ない。従って、(2)で述べたテスト手続きはプログラムの機能仕様の代りとなる重要なものであり、テスト手続きに反映されたテスト項目の充分性がそのままプログラムの品質に反映されることになる。 $\gamma$ ここで、テスト手続きの充分性評価の1つの方法として、準備したすべてのテスト手続きを実行した時にプログラムのパスなどの経度網羅されたかを測定する。

### (4) テスト工程の定式化

以上のような系統的テストを支援するHITSを用いた場合のテスト手順を図3に示す。テスト工程とプログラミング工程は仕様が与えられた段階から独立に開始



るテスト手続きは、図4に示すように、PROCコマンドから始まり、END PROCコマンドで終る一連のコマンド列で構成される。通常テストケースは複数個なので、各テストケースに固有のコマンドはCASEコマンドとEND CASEコマンドで囲む。このテスト手続きは、ライブラリ化しておいてEXECコマンドで実行できる他、直接端末に入力して実行することもできる。

本章の以下の節では、2章の設計方針を満たすために設けたコマンドの機能について述べ、構文等の言語仕様の詳細は次章に譲る。なお、アセンブラ用と高級言語用の相違部分についてはその都度述べる。

#### 4.2 環境模擬機能

##### (1) ドライブ定義

上位モジュールの模擬は、SETコマンドによる入カパラメータの値設定、G Oコマンドによる被テストプログラムの起動、IFコマンドによる出カパラメータの照合によって行うが、さらに高級言語用としてCALLコマンドを設けた。

##### (2) スタブ定義

下位モジュールの模擬は、STUBコマンドのサブコマンドにドライブ定義と同様のSET、IFコマンドを用いて行う。

##### (3) 外部データ模擬

被テストプログラム内で宣言されていない外部データは、GETコマンドでそのエリアを確保しておけば他のデータと同様に値の設定や表示ができる。

##### (4) 入出力模擬

被テストプログラムが入出力処理を含む場合は、ATコマンドでその位置に中断点を設定し、SETコマンドによる値の設定、LISTコマンドによる値の表示を行う。

#### 4.3 環境設定作業削減機能

HITSのような開発支援ツールは、その使用のための作業が多くなると普及が難しくなるため、次のような作業削減機能を充実させた。

(1) 繰返して用いるコマンド列をマクロ化しておいてそのマクロ名で引用したり、QUALLYFYコマンドで修飾名の省略時解釈を指定しておいて修飾付きアドレスの指定を簡略にすることによりテスト手続きの記述量を軽減できる。さらにアセンブラ用として、複雑なアドレス式の代りとなる名称をEQUATEコマンドで定義できる。

(2) 値の設定や照合に用いる定数をそのコマンドの実行毎に変えたい場合は、その定数リストをDATAコマンドで定義しておけば、以後その定数名を定数の代りに用いることができる。

(3) 上位/下位モジュールや外部データで既にオブジェクトが存在するものは、それをLINKコマンドのサブコマンドINCLUDEを用いて結合できるため、その模擬は不要になる。

#### 4. 4 デバッグ機能

(1) ATコマンドを用いて中断点を設定でき、LISTコマンドでデータの値をチェックできる。その際、中断点で入力するコマンドが決まっている場合はそれらをサブコマンドとしてあらかじめ指定できる。また、中断点の解除はOFFコマンドで行う。

(2) 分岐、副プログラムの引用など実行順を表示する制御トレースや、データの値変更の時点を検出するデータトレースおぼびりの解除はTRACEコマンドで行う。その際、BREAKオプションを指定しておけばトレース対象の事象生起毎に中断するため、トレース情報の出力を必要最小限にできる。

(3) 会話型処理時にLIST、TRACEコマンドを用いて大量の情報を出力する場合、OUTオプションを指定すれば出力先をTSS端末やユーザファイルに変更できるので、後でオフライン出力すれば良い。

(4) 誤りを検出したテスト手続きをデバッグのために再実行する時、BREAKオプションを指定しておけば被テストプログラムの実行開始直前に一時的コマンドを追加できるため、テスト手続きをデバッグ用に変更する必要がない。

(5) テスト実行中にシミュレータが不当命令コードやアドレスエラーなどの異常を検出した時のダンプ出力や連トレースなどの指示をSTUBコマンドのINTERRUPTオプションを用いてできるため、デバッグのための再実行回数や減らせる。また、無限ループ対策として、バッチ用には実行命令数の最大値指定機能、会話型では割込キーによるコマンド入力機能がある。

#### 5. コマンド言語仕様

HITSのコマンドの構文は表1に示すが、通常のコマンド言語と異なる特徴として、

(1) 手続き的概念の導入

(2) 対象言語依存性

(3) コマンド名などの短縮記法

などがある。まず(1)については、コマンドは本来個々に独立したものであり、コマンド間にわたるお互いの制約は少ない方がよいが、本言語はテスト手続きの記述に用いられるために、最小限の手続き的規則を導入した。その主なものは次の7種類のブロック構造である。

- ・テスト手続きブロック (PROC ~ END)
- ・テストケースブロック (CASE ~ END)
- ・コマンドリスト定義ブロック (CLIST ~ END)

表1 コマンドの構文と機能

| コマンド名   | オペランド 構文                                                                     | 機 能                          |
|---------|------------------------------------------------------------------------------|------------------------------|
| AT      | <実行アドレス指定並び> [ <u>コマンド</u> ]                                                 | 中断点設定とサブコマンド指定               |
| CALL    | <エントリ名> [ ( { <データ指定> { <定数> } ... } ) ]                                     | 被テストプログラムの実行開始               |
| CASE    | <テストケース名>                                                                    | テストケースの開始                    |
| CLIST   | <コマンドリスト名>                                                                   | コマンド列のマクロ定義の開始               |
| DATA    | <定数名> <定数> ...                                                               | 定数列の定義                       |
| DO      |                                                                              | サブコマンドの開始                    |
| END     | [ <種別キー> ]                                                                   | システム, 各種ブロックの終了              |
| EQUATE  | <名前> <実行アドレス指定>                                                              | 新しい記号名称の追加                   |
| EXEC    | <テスト年続き名> [ ( <テストケース名> ) ]                                                  | ライブラリ化されたテスト年続きの実行           |
| GET     | { <名前> [ <長さ指定> ] [ <絶対アドレス定数> ] } ...                                       | データエリアの確保                    |
| GO      | <実行アドレス指定> [ <実行アドレス指定> ]                                                    | 被テストプログラムの実行開始               |
| IF      | <条件式> <コマンド>                                                                 | 条件判定とサブコマンド指定                |
| INCLUDE | <モジュール名> ...                                                                 | 被テストプログラムのローディング             |
| LINK    | [ <セクション番号指定> ... <絶対アドレス定数> ] ... <コマンド>                                    | 各セクションのローディングアドレスとサブコマンド指定   |
| LIST    | <拡張データ指定> ...                                                                | 主メモリ, レジスタの値, メッセージの表示       |
| OFF     | [ *<br><実行アドレス指定並び> ]                                                        | 中断点の解除                       |
| OPTION  | <オプション指定> ...                                                                | 各種オプションの指定                   |
| PROC    | <テスト年続き名>                                                                    | テスト年続きの開始                    |
| QUALIFY | <修飾名>                                                                        | 修飾名の省略時解釈指定                  |
| RESUME  |                                                                              | 中断点からの実行再開                   |
| SET     | <代入文> ...                                                                    | 主メモリ, レジスタへの値の設定             |
| STOP    | [ <種別キー> ]                                                                   | テスト年続き, テストケースの強制終了          |
| STUB    | { <エントリ名> { [ <コマンド> ]<br>INTERRUPT }                                        | スタブ又は異常処理の定義開始               |
| TRACE   | { PROC<br>BRANCH<br>DATA ( <データ指定> ... )<br>BACK [ <整数定数> ]<br>PROC<br>OFF } | 制御, データのトレースと制御の逆トレーススキップの解除 |

(注) 記法の説明: [ ] は省略可, { } はいずれか1つ選択, ... は繰り返し

- ・リンクブロック (LINK ~ END)
- ・スタブブロック (STUB ~ END)
- ・条件ブロック (IF ~ END)
- ・中断点ブロック (AT ~ END)

これらのブロックを構成する最初のコマンドは、いずれも複数のサブコマンドの指定が可能なものであるが、これらのサブコマンド列を主コマンドのオペラントに記述させるような構文はテスト手続きとしての読み易さに欠け、TSS 端末からのキー入力も複雑になる。そこで、このようなブロック構造の導入により、1行1コマンドの簡潔なコマンド構文とした。但し、後の4コマンドについては、サブコマンドが1個の場合に限りオペラントに直接書けるようにして、記述の簡便さも考慮した。

オマの特徴については、HITS ギャアセンブラおよび高級言語 Super PL/H を対象とするため、コマンドの構文はこれらの言語とは独立した共通の構文にすることが考えられた。しかし、アセンブラと高級言語では言語構文が非常に異なること、およびコマンドの構文が対象言語と異なると本システムの使用者にその習得のための余分の労力を強いることなどの理由から、各コマンドのオペラントの構文はできるだけ対象言語に合せた。

例えば表1の構文規則において、〈実行アドレス指定〉は、高級言語用ではテスト用としては手続き名、デバッグ用としては変数を用いるがアセンブラ用ではロケーションにつけた記号名称および16進表現の相対/絶対アドレス指定を用い、さらにそれらの変位として16進定数の加減算もできる。また、〈データ指定〉は、前者では変数名、後者では〈実行アドレス指定〉と同形式の他、レジスタ指定や間接アドレス修飾、インデックス修飾形式も可能である。

オマの特徴として、まずコマンド名は2文字目以降の任

```

PROC TP21
  LINK 7:1000H;9:3000H, DO
  INCLUDE SUB1,SUB2
  END
  GET COMDATA,0A000H
  SET STATE=10110000B
  STUB SUB3, DO
  LIST P
  IF STATE=0F0H, SET P=40
  END
CASE C01

}

CASE C07
  CALL SUB1(5,V)
  LIST V,STATE
  IF V=0, LIST 'ERROR:V=0'
  END CASE

}

END PROC

```

(a) テスト手続き

[Ready 状態]

```

OPTION BREAK
EXEC TP21(C07)
[テスト-C07の開始後 手続き SUB1の]
[実行開始直後に中断。コマンド待ち。]
QUALIFY SUB1
AT 17, LIST STATE
TRACE DATA(STATE)
RESUME

```

[最初に STATE の値を変えて] 所で  
[中断。コマンド待ち。]

STOP PROC

[テスト手続きの強制終了。Ready 状態]

(b) 会話型デバッグ

図5 HITS の使用例

意の位置から後を省略できるようにして、会話型でデバッグする時に端末から入力する文字数を減らせるようにした。但し、例えばSのように3コマンドのいずれが識別できない場合は、あらかじめシステム側で使用頻度の多いもの(SET)に決めている。

最後に、高級言語用のテスト手続きの記述例とその会話型デバッグ時の使用例を図5に示す。

## 6. 処理方式

### 6.1 システム構成

本システムは、図1に示したように汎用計算機HITAC Mシリーズのオペレーティングシステム上で稼動するが、会話型で使用する時はそのTSSシステム上で用いる。被テストプログラムは、クロスアセンブラまたはクロスコンパイラの出カするオブジェクトモジュールを対象とするため、16ビットマイクロコンピュータHMCS 68000用ソフトウェアはプログラミングからテストまで一貫して汎用計算機上でのクロス処理可能になる。なお、ソースプログラム内で用いた名前やロケーションに関する情報は、クロスアセンブラおよびクロスコンパイラからオブジェクトモジュールの付加情報として受け取る。

### 6.2 プログラム構成

プログラムの静的構造は、論理的には最上位にモニタ、その下にコマンド処理、最下位にシミュレータという階層構造を基本としているが、コマンドによってはその解析と実行の間に時間的にずれがあるため、実際の動的制御構造は、図6に示すような状態遷移図になる。中断点処理は、後で実行するコマンドの制御のためにコマンド処理から分離したもので、各コマンドの実行ルーチンはコマンド処理と共有している。

## 7. おわりに

以上、16ビットマイクロコンピュータ用ソフトウェアのテストを汎用大型計算機上で効率良く行うための支援システムHITSについて述べた。

機能面では、単体、連動、システムテストの各段階を支援し、サフトテスト手続きを統一的に記述するなど、テストを系統的行えるようにした。そして、このテスト手続きは、コマンド言語として簡潔な文法にすると共に、テスト作業の極小化のためのコマンドを充実させ、使い勝手の良いシステムとした。

本システムは、高級言語Super PL/H用エーセンブラ用との結合版として開発中であり、現在、そのテストデバッグ段階である。

なお、適用段階において、HITSによる

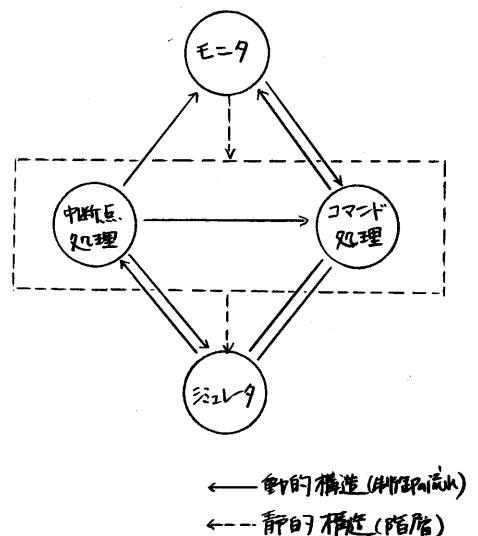


図6 HITsのプログラム構造

単件/連鎖テストをより効果的に行うために、設計工程においても、モジュール間インターフェイスを簡潔にするような構造設計を行うことや、各モジュールの仕様を明確にドキュメント化することが望ましい。

HITSは、テスト作業の効率化を実現したが、ソフトウェアテストのもう一つの課題である効果的テスト項目作成のための機能については、テスト網羅性評価機能だけでなく、今後、これらの機能を充実させていきたい。

最後に、本研究の機会を与えていただいた日立製作所戸塚工場道家浩太郎部長ならびに、有益な御討論をいただいた同工場山本利昭、同社システム開発研究所渡辺坦、山野統一の諸氏に感謝します。

### 参考文献

- 1) Boehm, B.W.: Software Engineering, IEEE Trans. Computers, Vol. C-25, No. 12, pp. 1226-1241 (1976).
- 2) 中所武司: ソフトウェアのテスト技法、信孚屋誌、Vol. 64, No. 5, pp. 549-552 (1981).
- 3) Goodenough, J. B. and Gerhart, S. L.: Toward A Theory of Testing: Data Selection Criteria, Current Trends in Programming Methodology, Vol. 2, Prentice-Hall, New Jersey, pp. 44-79 (1977).
- 4) 中所, 渡辺, 黒崎, 山本: マイコン用クロス型テストデバッグ支援システム HITS の設計思想、情報学会オズ3回全国大会, pp. 419-420 (1981).
- 5) 田中, 中所, 岡本, 本田: マイコン用クロス型テストデバッグ支援システム HITS の機能と処理方式、同上, pp. 421-422 (1981).
- 6) Myers, G. J.: Software Reliability, p. 360, A Wiley-Interscience, New York (1976).
- 7) Panzl, D. J.: Automatic Software Test Drivers, Computer, Vol. 11, No. 4, pp. 44-50 (1978).
- 8) Miller, E. F.: Program Testing Technology in the 1980s, Proc. Conf. on Computing in the 1980s, IEEE Computer Society, pp. 72-79 (1978).
- 9) 宮本勉: フログラム・テスト支援ツール: サーベイ, 情報処理, Vol. 20, No. 8, pp. 688-693 (1979).
- 10) 神野, 西野, 加藤, 横畑, 渡辺, 尾石: 16ビット・マイクロコンピュータ68000用高級言語S-DL/Hの開発、情報学会マイクロコンピュータ研究会資料 17-3, pp. 1-8 (1981).
- 11) 迫田行介: テストプログラム記述言語, 情報処理, Vol. 22, No. 6, pp. 520-524 (1981).