論　文

動的なパラメータ指定機能をもつあるデバッグシステム*

萩原兼一** 細見輝政**

Abstract

A debugging system (DS-11) that provides a new approach to the process of debugging has been implemented on PDP-11, and in use.

DS-11 has various kinds of commands and requests which enable a programmer to make effective use of the advanced debugging techniques as well as the usual ones such as post-mortem dumps, snapshots, trace and breakpoints. That is, not only the traditional AT-timing but also the newly introduced ON-timing is available in each request. Both of them specify a request point when the request should be served. The former specifies the request point in terms of only an instruction address, whereas the latter does it in terms of a program state. Thus the ON-timing can dynamically specity it, therefore the programmer is able to get the pertinent information about possible bugs and carry out the debugging effectively and efficiently.

1. まえがき

従来からよく使用されているデバッグ手法には，主につぎに挙げる4種類のものがある[1),2)].

D1. 死後ダンプ (Post-Mortem Dumps)
D2. スナップショット (Snapshots)
D3. トレース (Trace)
D4. 区切り点 (Breakpoints)

D1は，プログラムの異常終了時に自動的に，あるいは，プログラム中での呼び出しにより起動され，主記憶のすべての内容などを8進か16進数として出力するものである．D2は，予め指定しておいた番地の命令にプログラムの制御が来たときに，予め指定しておいた情報を出力するものである．D3は，各命令実行ごとにプログラムカウンタ（以下PCと略す.）や各種レジスタの値を出力するものである．D4は，予め指定された番地の命令を，一種の割り込み命令で置き換えておき，そこに制御が来たときにプログラムは中断状態となり，種々の処理を行った後にプログラムを続行できるものである．

これらはデバッグ手法として基本的なものと思われるが，D1やD3は莫大な出力量に比べて有用な情報が必ずしも多くはない．D2は情報を出力する時点としてある番地の命令を実行した後あるいは前という指定法をとるが，この指定法だけではデバッグに有効な出力時点を指定できるとは思われない．また，D4による中断時に，コンソールパネルやタイプライタからの入力による種々の処理を行うことも場合によっては時間効率上好ましいとは思われない．そこで，これらの基本的な手法をさらに有効に使用するための一つの方法と思われる動的な指定機能を持つデバッグシステム (Debugging System-11; DS-11) を作成したので報告する．

DS-11は，ミニコンPDP-11用のデバッグシステムであるが，その概念は種々のシステムに応用され得るものと思われる***.

---

* A debugging system with the functions in which parameters can be dynamically specified by Kenichi HAGIHARA and Terumasa HOSOMI (Faculty of Engineering Science, Osaka University)

** 大阪大学基礎工学部情報工学科

*** DS-11は，pdp-11のアーキテクチャに依存はするが，DS-11により導入された機能は，ほとんどのミニコン上で実現可能である．

2. デバッグシステムに対して望まれる機能

デバッグシステムがどのような機能を備えていればD1～D4の手法をさらに効果的に使用でき，効率的なデバッグを行えるかを分析し，それらを列挙する．

F1. デバッグシステムの備えている種々のサービス(ダンプなど)を行う時点(以下タイミングと呼ぶ)の指定法に融通性を持たせること．デバッグの際に，テストされているプログラム(以下TPと略す)において，その実行の各ステップごとの情報を出力することが必要なことは稀であり，ある特定のタイミングでの情報だけで十分であることが多い．このタイミング指定として有用と思われるものをつぎに挙げる．

T1. 指定した番地にある命令を実行したとき，この場合でも，その番地に複数回制御が来るならば，実行の最初から，指定した回数の間だけとか，あるいは，指定した回数の後だけとかの指定ができれば，ループに入った直後や脱出時とかの指定も可能となる．

T2. 指定した種類の命令を実行したとき，たとえば，サブルーチンコール／リターン命令を実行したときを指定できれば，サブルーチンごとの確認などに有効であろう．

T3. 割り込みが発生したとき．

T4. 指定したステップ数だけ進行したとき．

T5. 指定したレジスタや主記憶の内容が，指定した条件を満たしたとき，たとえば，デバッグ中に，ある領域の値が不適当な値に変更されていることを見出したとき，“その領域の値が不適当な値になる”という条件を何らかの形で表現でき，この条件が成立したときというタイミングを指定できれば効果的であろう．

従来のデバッグシステムではT1のみが許されるものが多いが，T2～T5，あるいはこれらを組み合わせた指定ができれば，ダンプなどの機能をより有効に使用できよう．

F2. デバッグシステムが会話型であること．一般に，プログラマは“虫”の原因を大まかにしか予測できないので，TPを適当な所まで実行させて中断し，種々の情報を収集し，より正確に“虫”の原因を予測するということの繰り返しでデバッグを行う．したがって，デバッグシステムは，指定したタイミング(**F1** 参照)でTPを中断し，任意の個所の情報を収集し，また，その内容を任意の値に変更する(**F3** 参照)ことなどができ，その後TPを続行できる会話型であることが望ましい．

ただし，現在の計算機の使用形態から，F2は小型計算機やTSSでのデバッグシステムにのみ実現可能なものと考えられる．

F3. TPの命令やデータ領域あるいはレジスタの値を変更できること．たとえば，あるモジュールが誤っていることを見出したとき，そのモジュールが仕様通りに動いたとした場合の出力などに整え，残りの部分のテストを続行できれば，ある“虫”のためにそれから先のTPのテストができないことを防ぐことができる．

F4. デバッグのためにTPを変更しなくてもよいこと．たとえば，各種の値を出力するためにWRITE文をTPに挿入するなどということを行わなくてもよい．つまり，デバッグシステムの諸機能をサービスするときに，少なくともユーザレベルでは，TPとデバッグシステムとが独立していること．特にアセンブラのような低級言語でのこのような変更は，多分に二次的な“虫”の原因に成り易いようである．

F5. 入出力の形式が多様であること．入出力する情報が数値であれば8進あるいは10進表示で，また，ASCII文字であればASCII文字のまま入出力できる方が一般に効果的である．番地の値を入出力する場合も，絶対番地と相対番地との選択ができる方がよい．

F6. 使い易いシステムであること．特に，ユーザとデバッグシステムとの通信手段としてのコマンドやリクエスト(**3.2** 参照)などは簡潔で全機能を活用できるものであること．

F5やF6の機能は，デバッグの際に基本的に必要とされるものではないが，効率よくデバッグするためには非常に重要なものと思われる．

3. DS-11 の概略

本章では，DS-11の概略を説明するが，その前に，PDP-11/20のデバッグ機能について簡単に述べる．

3.1 PDP-11/20 のデバッグ機能と DS-11 のデバッグモード

PDP-11/20には区切り点命令(BPT)とトレーストラップ機能との二つのデバッグ機能がある．前者はBPTを実行することにより起きるトラップであり，後者はプログラム状態語(Program Status Word; PSW)のトレースビット(Tビット)がセットされて

いるとき，各命令の実行ごとに起きるトラップであって，それぞれの指定番地に制御が移る．

DS-11 では，これら二つの機能を利用して，DS-11 と TP と制御を交換している．BPT を TP に埋め込んでおいてトラップする方法をランモード (R モード)，T ビットをセットして各命令ごとにトラップする方法をトレースモード (T モード) と呼ぶ．前者は，埋め込んだ BPT に制御が来たときのみ DS-11 に制御が移るので実行時間の損失が僅かなのに比べて，後者は，各命令実行ごとに DS-11 に制御が移るので実行時間が長くなるという欠点はあるが，綿密な追跡ができるという長所がある．これら両者の長所を配分よく利用すれば，つまりデバッグに必要な個所は T モードで，不必要な個所は R モードで追跡すれば，実行時間の損失は比較的少なく，かつ必要な個所は綿密に追跡できるので時間効率がよい．DS-11 では，これらのモードの切り換えを動的にも行うことができる．

(3.12 参照)．

3.2 コマンドとリクエスト

コマンドは，コンソールタイプライタから入力される DS-11 に対する指令であって直ちにサービスされる．主なコマンドを **Table 1** に示す．コマンド入力可能な DS-11 の状態をコマンド受付け状態 (*C* 状態) と呼ぶ．

リクエストは，LIST コマンド (**3.3** 参照) により登録される DS-11 に対する指令であって，TP が T あるいは R モードで追跡されているとき (DS-11 のこのような状態をモニタ状態 (*M* 状態) と呼ぶ) に，指定された条件 (**3.4** 参照) が満たされればサービスされる．主なリクエストを **Table 2** (次頁参照) に示す．DS-11 では，従来のデバッグシステムがそのサービスをコマンドレベルでしか提供しなかったものもリクエストとして使用可能にしている．このことにより，デバッグの時間効率を上げることができる．

Table 1 Commands of DS-11

コマンド名	機　能	形　式
DUMP	指定された領域を出力する	DUMP FROM addr1 TO addr2 [m][c]
MODIFY	指定された1語を変更する	MODIFY addr←exp [c]
GET	指定された TP をロードする	GET program-name
SAVE PROGRAM	プログラムの状態の退避を行う	SAVE PROGRAM program-name
SAVE REQUEST	リストやシンボルの退避を行う	SAVE REQUEST file-name
UNSAVE PROGRAM	退避したプログラムの状態を復元する	UNSAVE PROGRAM program-name
UNSAVE REQUEST	退避したリストやシンボルを復元する	UNSAVE REQUEST file-name
TRACE	Tモードで TP を追跡する	TRACE [pc][,ps][/step][: pc']
RUN	Rモードで TP を追跡する	RUN [pc][,ps][: pc']
JUMP HISTORY	JUMP HISTORY を出力する	JUMP HISTORY
FLOW HISTORY	FLOW HISTORY を出力する	FLOW HISTORY
CONFIRM	リストとシンボルを確認のため出力する	CONFIRM
OUTPUT	コマンドあるいはリクエストの出力装置を指定する	OUTPUT {1,2} device
KIND	コンソールタイプライタのエコーモードの切り換えを行う	KIND
SYMBOL		
CREATION	シンボルをベースあるいは参照シンボルとして定義する	(symbol-char)=exp{BASE,〈CR〉}
UPDATE	シンボルの値を変更する	(symbol-char)=exp←exp{BASE,〈CR〉}
DELETION	シンボルを消去する	(symbol-char)=exp DELETE〈CR〉
CONFIRM	シンボルの値を確認のため出力する	(symbol-char)=exp〈CR〉
LIST		
CREATION	新しいリストを定義する	LIST list-char list-char 0: request 0 list-char 1: request 1 ⋮ list-char n: 〈CR〉 END
UPDATE	リストを変更する	LIST list-char UPDATE list-char m: request m list-char m: request m', 〈CR〉 END
ELIMINATION	リストを消去する	LIST list-char ELIMINATE 〈CR〉
ATTACH LIST	指定されたリストをA属性にする	LIST list-char ATTACH
DETACH LIST	指定されたリストをD属性にする	LIST list-char DETACH
CONFIRM	リストを確認のため出力する	LIST list-char CONFIRM

(注)　ユーザは下線部を入力する．[ ] は省略可能なもの，{ } は二者のうちどちらかを選択する．
〈CR〉 は特に意味のある場合のみ記入してある．
[m] は出力様式の指定，[c] はコメントであり；ではじまる．

Table 2 Requests of DS-11

リクエスト名	機能	形式
JUMP TABLE	JUMP TABLE を出力する	JUMP TABLE
FLOW TABLE	FLOW TABLE を出力する	FLOW TABLE
INTERRUPT SURVEILLANCE	指定された割り込み処理ルーチンも追跡する	INTERRUPT VECTOR=addr
DUMP	指定された領域を出力する	[d] timing DUMP FROM addr TO addr2 [m][c]
MODIFY	指定された1語を変更する	[d] timing MODIFY addr←exp [c]
PAUSE	DS-11 を C 状態にする	[d] timing PAUSE [c]
SWITCH	TとRモードの切り換えを指定する	[d] timing SWITCH [c]
ATTACH LIST	指定されたリストをA属性にする	[d] timing LIST list-char ATTACH [c]
DETACH LIST	指定されたリストをD属性にする	[d] timing LIST list-char DETACH [c]
COMMENT	コメントを出力する	[d] timing ; comment

(注) ユーザは下線部を入力する. [ ] は省略可能なもの.
[d] は出力装置の指定, [m] は出力様式の指定, [c] はコメントであり; ではじまる.
最初の3つのリクエストは, それが登録されているリストがA属性のときはいつでもサービスされる.

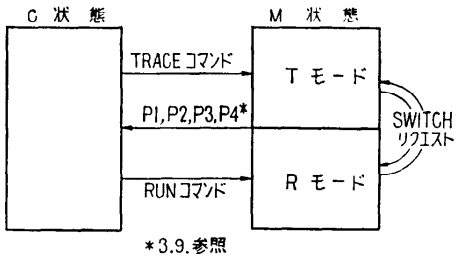


Fig. 1 State transition diagram of DS-11

C 状態から *M* 状態への遷移は TRACE あるいは RUN コマンドにより, 逆の遷移は P1～P4 (**3.9** 参照) の要因により起こる. その遷移図を **Fig. 1** に示す.

3.3 リ ス ト

リストはリクエストの集合である. リストは複数個定義でき, それぞれは1字の英文字により識別される. 各リストには, そのリストに登録されているリクエストがサービスの対象となるA属性と, 対象とならないD属性という二つの属性がある. 新しく定義されたリストはA属性である. リストの二つの属性の切り換えは, コマンドで行う. また, リクエストにより DS-11 の *M* 状態でも行える (**Fig. 2** 参照) ので, 本来リクエストの持っている動的な能力をさらに高めることができる. たとえば, リスト A (**Fig. 3**) が定義されており, また TP 実行前にリスト A, B, C がそれぞれ A, A, D 属性であったとする. TP が0番地から順に実行されて行くものとすれば, TP の制御が 0～200 番地または 300 番地以降にあるときのみリスト B は A 属性であり, リスト A は 0～400 番地のときに A 属性である. ところで, TP の制御が 500 番地に来たときにはリスト A は D 属性であり, したがってリスト A に登録されているリクエスト A3 は無視され, リスト C は 100 番地以降 A 属性のままである. また, リスト D (**Fig. 4**) を定義しておくと, D0 と D1 のリクエストにより 100 番地の内容が 0 で

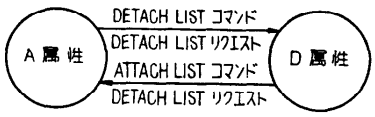
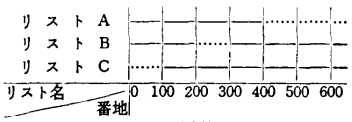


Fig. 2 Attribute transition diagram of a LIST



*LIST A
A0: AT 200 LIST B DETACH
A1: AT 300 LIST B ATTACH
A2: AT 100 LIST C ATTACH
A3: AT 500 LIST C DETACH
A4: AT 400 LIST A DETACH
(注) AT…については **3.4** 参照

Fig. 3 Attribute transition of LISTs by the At-timing

*LIST D
D0: ON 100 .NE #0 LIST E ATTACH
D1: ON 100 .EQ #0 LIST E DETACH
D2: ON %2 .HI #100 & %2 .LOS #200 LIST F ATTACH
D3: ON %2 .LOS #77 LIST F DETACH
D4: ON %2 .HI #201 LIST F DETACH
(注) 数値は8進表示されている.
印の付いている数は, その数自身の値を
単なる数は, コア上でその数で示される番地の内容を表わすものとする. .NE, .EQ, .HI, および .LOS は, それぞれ ≠, =, > および ≤ を意味する.
ON…については **3.4** 参照

Fig. 4 Attribute transition of LISTs by the ON-timing

ないときのみリストEはA属性，D2，D3およびD4のリクエストにより %2* の値が 100 より大きくかつ 200 以下のときのみリスト F は A 属性である．

3.4 タイミング

タイミングは，各リクエストが M 状態のどの時点でサービスされるかを指定するものである．DS-11には，オンタイミングとアトタイミングと呼ばれる2種類のタイミングがある**．制御が DS-11 に戻った時点でタイミングは真か偽かのどちらかの値をとる．リクエストは，それがA属性のリストに登録されており，かつそのタイミングが真の値をとるときのみサービスされる***．

オンタイミングは，2.で述べたT2やT5を主に実現することができ，その様式はつぎの通りである．

ON cond**** [& cond]***** [& cond]

オンタイミングにより，T2 を実現できる．たとえば，サブルーチンからの戻り命令****** (RTS) が行われるときを指定するには，つぎのようなオンタイミングを使えばよい．

ON ♯200 .LOS @%7 & @%7 .LOS ♯205

ここで @%7 は，%7******* の示している番地の内容，つまりつぎに実行される命令を意味し，このオンタイミングは @%7 の値が 200 以上 205 以下のとき，つまりつぎに実行される命令が RTS のときに真となる．この方法を適用すれば，任意の種類の命令が行われるときを指定するオンタイミングを設定することができる．

オンタイミングを使用すればT1も実現できるが，これは使用頻度が高いのでアトタイミングとして独立に実現した．その様式はつぎの通りである．

AT ADDR* [/COUNT*]

ここで，ADDR は主記憶上の番地を，COUNT はアトタイミングが真になるための回数に関する条件，いわゆる通過回数指定を表わしている．たとえば，

AT 20000/5

は，このアトタイミングで指定されるリクエストの属しているリストがA属性になってから，20000 番地の命令が何度か実行されるとき，最初の5回だけこのアトタイミングが真となり，また，

AT 20000/-5

は，最初の5回は偽で，6回目以降は真となる．COUNT は，省略されればそのアトタイミングは ADDR で示される番地の命令を実行するたびに真となる．このように，COUNT の指定により，ループ中での必要な回数目にリクエストをサービスしたり，回帰的プログラムの追跡にも利用できる．

3.5 割り込み処理追跡機能

PDP-11 では，割り込み処理ルーチンの開始番地が，割り込み要因に対しそれぞれ固有の番地（ベクトルと呼ばれる）に格納されている．

DS-11 では，INTERRUPT SURVEILLANCE リクエストによりベクトルを指定すれば，そのベクトルに対応する割り込み処理ルーチンの中も追跡できる．このリクエストにより，T3 が実現されている．

3.6 ステップ数に関するデバッグ手法

DS-11は，Tモードで実行されているTPの実行ステップ数を数える 16 ビットのカウンタ (STEPCT) と，STEPCT の桁あふれの回数を数える 16 ビットのカウンタ (EXTSTEP) を持っている．この二つのカウンタをオンタインミングを使用して監視することにより，たとえば，疑似的に“タイマを用いたデバッグ”[3] などに利用でき，種々のタイミング指定に使用できる．

TRACE コマンドは，Table 1 に示す様式であるが，PC で指定する番地から，PS で指定する値を PSW の内容として**，STEP で指定するステップ数だけ，PC′ で指定する番地まで，Tモードで TP を実

---

* PDP-11 には8つの汎用レジスタがあり，それらをそれぞれ %0, %1, …, %7 と表わす．

** オンタイミングは，TP が T モードで追跡される場合に使用されることを想定している．

*** A 属性のリストと D 属性のリスト中に同一リクエストがあった場合，前者のリクエストのタイミングが真の値をとれば，そのリクエストはサービスされる．

**** cond は，(オペランド1) .RE (オペランド2) という様式であり，RE には大小関係などを示す文字列が入る．この cond により，PDP-11 のアセンブリ言語での比較命令で判定できる条件を表わすことができる．

***** 以降 [ ] で囲まれたものは省略可能である．したがって，一つのオンタイミングで三つまでの cond の論理積をタイミングとして指定できる．たとえば，

ON ♯200 .LOS %3 & %3 .LOS ♯300

というオンタイミングは，%3 の値が 200 以上 300 以下のときのみ真となる．また，ATTACH LIST および DETACH LIST リクエストを併用することにより，任意個の cond の論理積の値をもつオンタイミングを設定することが可能である．

****** PDP-11 では戻り先をどの汎用レジスタに退避さすかにより，RTS の命令コードは 200〜205 となる．

******* PDP-11 では，%7 をプログラムカウンタ PC として使用している．

---

* ADDR と COUNT とは，リテラルやシンボルにより数式表現されそれぞれ整数をその値として持つ．

** PSW は 16 ビットで構成されており，それを2進数とみたときの値により指定する．ただし，第4ビットはTビットにあたり，そのビットがオフになるような値を指定して，TRACE コマンドを発することは無意味である．

行せよというコマンドである*. STEP を指定しなければ，無限のステップ数を指定したものとみなされ，PC あるいは PS の指定がなければ，それぞれそのままの状態で TP は続行される．

3.7 FLOW TABLE と JUMP TABLE リクエスト

FLOW TABLE リクエストは，T モードで TP の各ステップごとにその PC の値をラインプリンタに出力させるものである．

JUMP TABLE リクエストは，*M* 状態で TP のブランチ発生を監視し，そのブランチ命令の番地とブランチ先との番地を対にしてラインプリンタに出力させるものである**.

一般に，TP の制御の流れを追跡するためには後者で十分であるし，前者に比べてラインプリンタへの出力量もかなり少なくなる．

3.8 FLOW HISTORY と JUMP HISTORY コマンド

FLOW HISTORY および JUMP HISTORY コマンドにより与えられる情報は，それぞれ前述の FLOW TABLE および JUMP TABLE リクエストと同様のものであるが，コマンドが発せられた時点における有限長（DS-11 作成時に指定する）の過去の制御の情報を出力させるものである．

これらのコマンドにより，*C* 状態でその直前の TP の制御の流れが有限ではあるが与えられることは，デバッグの際に有益であると思われる．

3.9 会話型システム

DS-11 が *C* 状態になる原因としてつぎの 4 つがある．

P1. TRACE コマンドの STEP または PC′ の指定（**3.6** 参照）が満たされたとき．

P2. PAUSE リクエストのタイミングが真になったとき，

P3. ユーザにより，コンソールパネルから TP 実行の中断を指示されたとき．

P4. TP が未定義命令 (illegal instruction) などを実行しようとしたとき．

P1〜P3 は，DS-11 に対するユーザの自発的な *C* 状態への遷移指示によるものであるが，P4 は DS-11 自身の指示によるものである．

P3 はコンソールパネルのスイッチレジスタの最下位のビットを反転させることにより行われる．これは，たとえば TP が無限ループに突入したと思われるときなどで，P1 や P2 による方法が効力を発揮しないときに使用される．

M から *C* 状態へ遷移するときは，遷移した原因 (P1〜P4)，STEPCT の値，直前に実行された命令の番地，つぎに実行されようとしている命令の番地，および PSW の条件ビットの状態をコンソールタイプライタに打ち出す．*C* 状態では，種々のコマンド処理を行え，TRACE あるいは RUN コマンドにより *M* 状態に遷移できる（**3.2** 参照）．このように，DS-11 は *C* 状態と *M* 状態との間の遷移が自由に行える会話型システムである．特に，P2 の PAUSE リクエストで，従来から行われている区切り点の機能をアトタイミングを用いて行えるほか，オンタイミングを用いて種々の状況に応じて *C* 状態への遷移が行える点がデバッグの際に有用であろう．

3.10 MODIFY コマンドとリクエスト

MODIFY コマンドあるいはリクエストにより F3 の機能が行える．それぞれ *C* 状態あるいは *M* 状態で，アドレス空間上の任意の語，つまり命令や作業用領域あるいは各種レジスタの値を任意の値に変更することができる．たとえば，**Fig. 5** のリクエスト G0 と G1 により，指定したタイミングが真になったとき，そのときのスタックの先頭にそのときの汎用レジスタ %3 の値をプッシュすることができる．

3.11 入出力形式指定

コマンドあるいはリクエストにより情報を出力する場合，その値を 8 進，10 進，あるいは 16 進表現で出力するように指定できる．また，その情報をバイトごとの数値として，あるいは ASCII コードとみなしその ASCII 文字として出力するようにも指定できる．さらに，番地の値を出力する場合，絶対番地として，あるいはある番地をベース* としてそこからの相対番

```
*LIST G
  G0: (タイミング) MODIFY %6←%6-2
  G1: (タイミング) MODIFY @%6←%3
(注) G0 と G1 のタイミングは同一のもの．
     PDP-11 では %6をスタックポインタと
     して用いている．
```

Fig. 5 Dynamic stack modification by the MODIFY-requests

* DS-11 は，TP が STEP で指定したステップ数だけ進行するか，PC′ で指定した番地まで進行するか，そのうちのどちらか先に成立した条件で *C* 状態となる．

** 条件ブランチの際に，その条件が成立せずにつぎの番地の命令が実行されるときはブランチが起きたとはみなさない．

* SYMBOL コマンドにより，このようなベースを設定できる．

地として出力するように指定できる．また，コマンドやリクエストのパラメータなども同様に絶対番地や相対番地で，また8進や10進表現で入力することができる．

この入出力形式指定機能は，**2.** で述べたF5を満足し，ユーザが煩雑な変換をしなくても望みのデータ形式で情報を処理できる (Table 1, Table 2 参照)．

3.12 SWITCH リクエスト

このリクエストは，TモードとRモードとの切り換えを行う．

一般に，主プログラムはいくつかのサブルーチンを呼ぶが，ボトムアップにシステムを開発して行く場合にはそれらのサブルーチンがすでに正しく動くと分かっていることがしばしばであるし，また，トップダウンに開発して行く場合にはそれらのサブルーチンがまだコーディングさえされていないことが多い．

前者の場合，それらのサブルーチンをテストする必要がないのでRモードで，主プログラムだけをTモードでテストする方が時間効率がよい (**3.1** 参照)．たとえば，**Fig. 6** に示すように X 番地にサブルーチンコール命令があり，Y 番地がその帰り先とすると，リスト H のリクエストでこのことが行える．

後者の場合は，X 番地の命令を実行後に MODIFY コマンドあるいはリクエストによりあたかもそのサブルーチンを仕様通りに行って帰って来たようにレジスタやデータ領域を変更し，Y 番地の命令からテストを続行すればまだ実現されていないサブルーチンを呼んでいるプログラムの追跡も行える．

3.13 SAVE PROGRAM および UNSAVE PROGRAM コマンド

SAVE PROGRAM および UNSAVE PROGRAM コマンドを利用すれば，いわゆる“チェックポイントリスタート (checkpoint restart)”が行え，何度か“虫”を再現させるのに，毎回TPの先頭から実行させる必要がなく時間効率がよい．

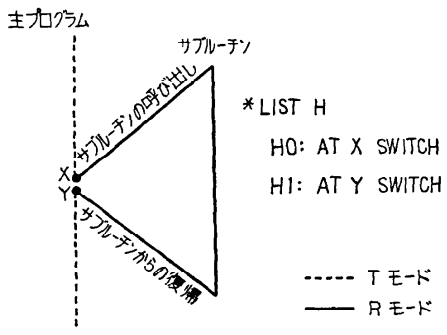


Fig. 6 Change of the program mode in the M-state by the SWITCH-requests

4. むすび

本稿では，現在多くのプログラマが行っているデバッグ手法をより効率的なものとする方法として，オンタイミングという巾広いタイミングの指定法と，動的な指定機能をもつリストを備えたデバッグシステムDS-11について述べた．文献 6) には，シミュレータアナライザとして望ましい機能がいくつか挙げられているが，DS-11はそれらのほとんどを含んでおり，我々の知る範囲では，リクエストのサービス時点の指定を，DS-11ほど柔軟かつきめ細かく動的に行えるデバッグシステムはない．

DS-11 は，1973 年末から稼動しており，現在までに数回の改訂がなされ，枯れたシステムとなってきている．そして様々なソフトウェアシステムの開発に寄与してきた．特に，共通のデータファイルを処理する複数個のモジュールを複数人で作成したときに，DS-11 が使用されたが，データファイルの変化をオンタイミングで容易に検知でき，そのことがこのようなシステム開発には非常に有効であったことを記しておく．

DS-11に残された課題は，オンタイミングやリクエストのマクロ的表記法およびシンボリックな番地指定法の実現，さらに高級言語で書かれたプログラムのデバッグへの応用である．また，プログラム開発支援システムに埋め込むことが計画されている．

最後に，日頃御指導賜わる嵩忠雄教授，数々の御助言を頂いた都倉信樹助教授，藤井護助教授，ならびに熱心に御討論頂いた嵩研究室諸氏に深謝いたします．

参考文献

1) P.C. Poole: “Debugging and testing” in Advanced Course on Software Engineering, F.L. Bauer, Ed. New York: Springer-Verlag, pp. 278～318 (1973)
2) W.C. Hetzel: Program Test Methods, Englewood Cliffs, N.J., Prentice-Hall (1973)
3) 古川，岡田：タイマを利用したデバッキングシステムについて，情報処理学会第 15 回大会論文集，pp. 111～112 (1974)
4) 萩原，細見：動的な指令機能を持つデバッグシステムについて，信学会全大論文集，p. 6-176

(1976)
5) DS-11 使用手引書, 嵩研資料 (1973)
6) E. C. Berkeley: "Computer-Assisted Documentation of Working Binary Computer Programs with Unknown Documentation" in Software Engineering/Vol. 2, J. T. Tou, Ed. New York: Academic Press, pp. 1～17 (1971)

(昭和51年4月12日受付)
(昭和52年5月9日再受付)