文書処理とヒューマンインタフェース 14－2
（１９８７．９．７）

時間情報を利用した制御を可能にするＭＳ－ＤＯＳの機能拡張について

森川　治

製品科学研究所

（梗概）　人間どうしの対話では対話内容だけでなく、対話の間合いもスムーズな対話を実現する意味において重要な役割を持っていると考えられる。また人間は、たとえ相手が機械であろうと、タイミングによって何等かの情報を相手（システム）に送っていると考えられる。従って、使いやすい人間-機械間の対話の実現には、対話におけるタイミング情報の利用方法の研究が必要であると考えられる。

この研究を進めて行くためには、時間情報を利用した制御の可能なコンピュータシステムが必要である。そこで、パーソナルコンピュータPC-9801シリーズのMS-DOS上でこれらの機能を実現する方法を開発したので報告する。

Extended functions on MS-DOS for operating timing information.

Osamu MORIKAWA

Human Factors Research Department, Industrial Products Research Institute, AIST, MITI, Ibaraki, 305 Japan

In human conversation, not only its content but also utterance timing plays an important role in achieving smooth communication. Humans also convey some kind of information through timing, whether the opponent is a human or machine. Thus, in order to realize a user-friendly man-machine interface, it is necessary to study the use of timing information in conversations. For this kind of study, it is necessary to have a computer system that can be controlled utilizing timing information. In this paper, the development of a method to introduce such capabilities on a PC-9801 with MS-DOS is reported.

1、はじめに

人間どうしの対話では対話内容だけでなく、対話の間合いもスムーズな対話を実現する意味において重要な役割を持っていると考えられる。例えば、対話中相手の反応が遅い場合、「自分の言った言葉が聞こえていなかったのではないか」、「内容が理解できていないのではないか」、「興味がないのではないか」等と推測をし、それを手がかりに「再度話しかける」、「使用する言葉を変えて言い直す」、「話題を変える」等、対策をたてて我々人間はスムーズな対話を実現している。

対話の間合いが重要であることは、人間-機械間の対話においても同様で、ユーザの入力に対するタイムリーな応答はシステムをユーザに受け入れやすくすると同時にユーザのエラーを少なくさせると考えられる[3][4]。逆に、入力に対する応答の過度の遅れはユーザに「入力されていない」、「入力が正しく行われていない」あるいは、「システムが何か間違えている」などと思わせ、システムを使いにくくする原因になっていると考えられる。逆に人間はたとえ相手が機械であろうと、タイミングによって何等かの情報を相手（システム）に送っていると考えられる[6][7]。ただ、現在のシステムでは、この情報をどの様に扱ってよいのか分からないので、無視しているだけだと考えられる。例えば現在の人間-機械間の対話でも、タイムリーでないユーザの応答から「実行結果が希望した結果でなかったのではないか」「次に行うべき処理を思考中なのか」「システムのコマンド名を忘れてしまったのか」等の情報が推測可能と考えられる。

人間-機械間の対話を人間-人間間の対話と全く同じ方式にすべきとは考えないが、人間-人間間の対話に学ばなくてはならない要素も多くあると考えられる。その中の１つとして、対話における人間のタイミング情報の利用方法の研究があると考えられる。

この研究を進めて行くためには、コンピュータシステムが適切な応答時間特性を実現できることと同時に、人間の応答時間を計測し、それをアプリケーションプログラム内で使用できることが必要である。

この様なシステムの実現方法として、計測・実験用の特殊なハードウエアをつくり、それを利用する方法があるが、設備が大掛かりになり、実験に使用できるソフトウエアの種類が限られるといった欠点がある。そこで、実験に使用できそうな多くのソフトウエアが市販されているPC-9801シリーズのパーソナルコンピュータ上のMS-DOSに可能な限りソフトウエアでこれらの機能を実現する方法を今回採用した。

2　拡張機能の詳細

本システムを起動することにより、PC-9801シリーズのMS-DOS上で利用可能になる拡張機能は、

①　高精度（１／１００秒）での経過時間の測定、

②　指定時間後の制御の変更、

③　タイムアウト機能付きのキーボード入力、

④　キー操作、各種外部装置の状態をタイムスタンプ付きで記録する機能、

⑤　仮想キー打鍵、

⑥　異なるキー配置のキーボードをエミュレートする機能、

の６点である。アプリケーションプログラムは、PC-9801のBIOSやMS-DOSのシステムコールと同様な手順で本システムを呼び出すことにより、拡張された機能のサービスを受けることができる。これらの基本的使用方法は、AHレジスタに機能番号をセットし、割り込みタイプ28（16進数の1C）のソフトウエア割り込みを発生させることである。また、各サービスルーチンへ渡すためのパラメータは、8086のレジスタを介して行う。なお、Ｃ言語（OPTC86）からこれらを使用するのに便利な様に、専用のライブラリを作成した。以下の説明では、主にＣ言語からの呼び出しを中心に述べていく。

①　高精度（１／１００秒）での経過時間の測定、

本システムを起動すると、1/100秒の分解能の時計が起動を始める。この時計を基に各種のサービスを提供している。この時計は４バイトから成り、最大約５００日まで計測できる。この時計の値を読み出す関数 get0time() と時計の値を強制的に０にする関数 tm_clr() がある。なお、関数 get0time() の値は４バイトの符合なし整数である。さらに経過時間を計測するのに便利な様に、関数 gettime() を用意してある。つまり、経過時間を計測したい処理の前後で gettime() を実行すれば、目的の処理の経過時間が後者 gettime() の関数値として得られる。

```
/*** gettime() 使用例 ***/
main()
{
    long gettime(),time;
    gettime();
    sub();                  /* 時間測定したい処理 */
    time=gettime();
    printf("SUBの処理時間は  %d0 msec です. ",time);
}
```

②　指定時間後の制御の変更、

指定した経過時間後（time）に指定した処理（sub）に制御を移すサービス rqtint(time,&sub) と、それをキャンセルする rstint() がある。機械語レベルでのサポートはこ

の２種類だけであるが、このサービスを裸のままＣ言語（OPTC 86）で使用するにはいろいろな点で問題がある。例えばＣ言語では、8086のセグメントレジスタDSとSSの値は等しく、２つのポインタレジスタBPとSPによって指されている領域のメモリは局所変数としてアクセスすると仮定してプログラムされている。また、MS-DOSがシングルユーザ、シングルタスクを前提に作られているため、MS-DOSのサービスルーチンがリエントラントな構造になっている保証はなく、割り込み先でシステムサービスを要求すると動作がおかしくなる可能性があること等である。そこで、本システムのＣ言語用ライブラリでは、関数 rqtint() の時間精度を犠牲にし、アプリケーションプログラムを動作中に限り、制御移動のサービスを受け付ける様に改造した物 rqltint(),rsltint() を用意している。このため、例えば、1.00秒後に割り込みを設定したものが、実際には1.01秒後以降になる場合もあるが、この誤差は人間にとって問題にならない時間と考えている。

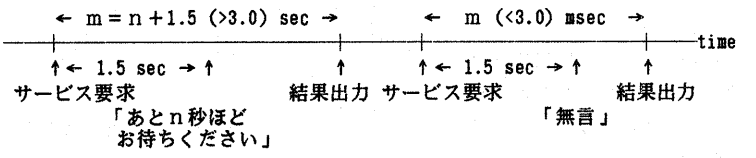

図１　メッセージ出力タイミング

本機能により、「処理が長引く（例えば３秒を越える）様な場合、約1.5秒経過した時点で、あとどの位待つべきかをユーザに知らせる」といったことがアプリケーションプログラムで実現可能になる（図１）。具体的なプログラムは、次の様にすればよい。

ｱ) 作業の進行状況を関数 msg_sub() が推定できるように、適当な変数を外部変数として宣言する。

ｲ) 処理を行う直前に、rqltint(150,&msg_sub); を実行し、1.5秒後に制御を msg_sub()に変更するように本システムに要求を出す。

ｳ) 処理が終了したなら、rsltint(); を実行し制御変更の要求をキャンセルする。

一方、メッセージを出力する関数 msg_sub() では、次の様にすればよい。

ｴ) ここに制御が移るのは、本体の処理が1.5秒経過した時点である。そこで、ｱ)で定義した外部変数を参照して作業の進行状況を計算し、あと1.5秒以上処理時間が必要と判断された場合には、それなりのメッセージを出力する。1.5秒かからないと判断された場合にはなにもせずに終了する。

```
/***  制御の変更  rqtlint 使用例 ***/
int i,n;                              /* ｱ */
main()
{
    int k;
    for(n=10; n<80; n+=10) {
      rqtlint(150,&msg_sub);  /* ｲ */
      printf("¥nn=%d",n);
      for(i=k=0; i<n; i++) {  /*全体でｎヶ処理して*/
          subl(n);            /*困難さはｎの関数で*/
          k+=i;               /*現在はｉ番目を処理*/
          }                   /*中であるという例  */
      rstlint();              /* ｳ */
      printf(" k=%d",k);
      }
}
msg_sub()
{
    int t;
    t=(n-i)*150/i;            /*t＝残り時間（n,i）*/
    if(t > 150) {             /* ｴ */
      t /= 100;
      printf("¥nただ今 %d 番目を処理中です．¥
        あと %d 秒程お待ち下さい ¥n",i,t);
      }
}
```

③　タイムアウト機能付きのキーボード入力、

ユーザが全く何を行なったら良いのか分からずに困惑している状態を、システムが「十分長い沈黙」により関知し、状況打開のアドバイスを自動的に起動するといったことが、特に初心者のユーザを対象にしたアプリケーションプログラムには必要な機能と考えられる。しかし、通常のキー入力ルーチンでは、文字が入力されるまでは、制御がアプリケーションプログラムに戻って来ないので、この様なサービスを実現することは困難である。そこで、本システムでは、時間制限付きのキー入力関数 getct(limit)、kwait(limit) と時間制限の設定関数 tlimit_set(limit) を用意した。指定できる時間は1/100秒単位で最大655.35秒（約１１分）までである。getct(0),kwait(0) のときは、直前に tlimit_set() で設定された時刻までが制限時間になり、０以外の値をパラメタにした場合には、その値が制限時間になる（図２）。これらの関数はシステムのキーバッファを更新しないので、ＡＴＯＫの様なフロントエンドプロセッサに影響を与えることがない。２つの関数 getct() kwait() は同じ様な機能であるが、getct() はBIOSのキーバッファに、kwait() はMS-DOSのキーバッファに入力されるまで

待つところが異なっている。例えば、フロントエンドプロセッサを使用中、文字を確定するのが遅れると、getct() ではタイムオーバーにならないが、kwait()ではタイムオーバーになることがある（図６）。

```
/*** getct 使用例 ***/
#define TIME_LIMIT 300
main()
{
    int class;

    printf("あなたの腕前は（1-5 級）？");
    if(getct(TIME_LIMIT)) scanf("%d",&class);
    else                  class=5;

    printf("¥nシステムはあなたを ¥
          %d 級と判断しました．",class);
}
```

④　キー操作、各種外部装置の状態をタイムスタンプ付きで記録する機能

この記録（以下これを打鍵データと呼ぶ）によりアプリケーションプログラムは、ユーザの操作状態をモニタリングすることができる。打鍵データは、プログラム内ですぐ利用できることは勿論、作業終了後に別のプログラムで採取し保存することも可能である。そのため、解析時に便利なようにユーザの打鍵データ以外に、表１のような情報を採取する指定ができるようになっている。記録の形式は事象コード１バイトとタイムスタンプ２バイトの計３バイトが基本になっている。

キーのメーク／ブレーク時の事象コードは、入力されたアスキーコードでなく、キーボードとほぼ１対１に対応するＰＣ９８０１固有の物理キーコードである。ただし、SHIFTキーとリターンキーはキーボード上に２つずつあるがどちらを押しても同一の事象コードが記録される。事象コードがＲＳ２３２Ｃの場合、直後にＥＳＣマークが記録され、ＥＳＣマークのタイムスタンプ部には受信したコード１バイトと、ステータス１バイトが記録される。事象コードの「タイムオーバー」は、本システムの時計の下２バイトがオーバーフローする毎に（655.36秒＝約１１分）に記録され、このタイムスタンプ部には、時計の上位２バイトが記録される。このデータ形式は木村ら[2]の「打鍵データ収集システム」と同じ形式となっている。記録データは、本システム内部のリングバッファに保存される。もしバッファのサイズ以上のデータが来たときは、最初のデータから捨てられていく。

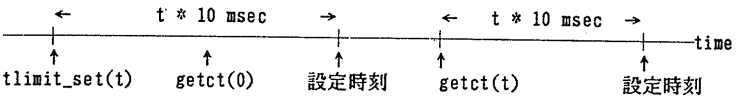

図２　設定時刻の決まり方

ここでは、リングバッファから記録を読み出す ks_get() と ks0_get()（ks0_get()はリングバッファの更新は行わない）、記録を書き込む ks_put(key_code) と ks0_put(key_code,time)（ks_put()ではサービスを起動した時刻のタイムスタンプが自動的に押される）、更にリングバッファを初期化して空にする ks_init()、リングバッファの状態を調査する ks_stat()を用意した。記録内容を指定する関数として ks_sw_set(ks_sw)、現在の指定状況を知る為の ks_sw_get()を用意した。 ks_sw の値の各ビットの意味は以下の通りでビットが１の事象は記録を行わない。初期値は０である。

```
play_mode  = 01h      （システム予約）
key_mask   = 02h
f8_mask    = 04h
hd_mask    = 08h
f5_mask    = 10h
RS232C_mask= 20h
mouse_mask = 40h      （予定）
CRT_mask   = 80h      （予定）
```

本システムではリングバッファをメインメモリ上に配置してあり、標準で256事象を記録できる容量にしてある。バッファ容量は、システム立ち上げ時に指定することにより、最大約２万事象まで拡大できる。なお、２万事象以上の打鍵データを収集する場合にそなえ、バンク切り替えを利用した大容量ＲＡＭボード上に打鍵データを記録するシステム（BMKS）も用意してある。BMKSシステムを動作させるに

```
(1) キーのメーク     0 ≦code＜0x77
(2) キーのブレーク   0x80≦code＜0xF7
(3) ２ＨＤ           code=0x78
(4) ハードディスク   code=0x79
(5) ２ＤＤ           code=0x7A
(6) ＲＳ２３２Ｃ     code=0x7C  +  ESC data stat
(7) タイムオーバー   code=0xFF  (*)
(8) ＥＳＣマーク     code=0xFE  (*)
（*タイムスタンプ部の２バイトは別の用途に使用される。）

  これ以外の事象コードとして以下のものを予定してる。
(9) プログラムマーク  code=0x7B  +  ESC n1 n2
(10)マウス            code=0x7D  +  ESC x1 x2 ESC y1 y2
(11)画面更新          code=0x7E
(12)再生モード終了マーク code=0xFD
```

表１、操作データとして記録される事象

はＩ・Ｏデータ社のＲＡＭディスクドライバＩＯＳ－１０が起動している必要がある。最大記録事象数は増設しているＲＡＭボードの容量、ＩＯＳ－１０の設定条件によって異なるが、BMKSシステムでは最大130万事象（４MByte）までサポートしている。またBMKSシステムではリングバッファをサポートしていない。従って、途中で記録されている打鍵データを読みだしても以後記録できる容量に変化はなく、メモリの最後まで記録されると以後のデータは記録されずに捨てられる。

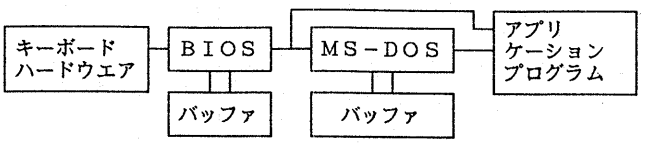

図３、ＰＣ－９８０１　ＭＳ－ＤＯＳのキー入力の流れ

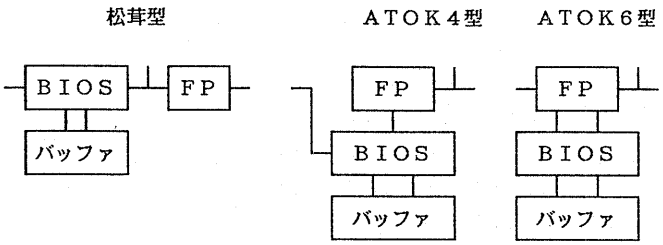

図４、フロントエンドプロセッサ（ＦＰ）の組み込まれ方

⑤　仮想キー打鍵、

PC-9801のMS-DOSでは、キーボードから入力された打鍵データは、先ずBIOSに入りそこで、物理キーコードから文字コードに変更されBIOSのバッファに貯えられる。つぎにそれは、MS-DOSシステムによってファンクションキーやCTRL-C等の特殊キーの部分に解釈処理が施されてMS-DOSのバッファに貯えられる。最後に、そこからアプリケーションプログラムにデータが渡される（図３）。さらにATOK等のフロントエンドプロセッサがシステムに組み込まれている場合には、BIOSとMS-DOSとの間にもう１段変換処理が入ることになる（図４）。

本システムでは、ユーザがキーボードを打鍵したのと同様な効果をシステムに与える、つまりBIOSのキーバッファにデータを書き込む関数 vkey_put(code) を用意してある。この関数は、アプリケーションプログラムで、ATOKの様なフロントエンドプロセッサを制御したい時に利用する。本機能により、例えば表計算プログラムにおいて、日本語を入力する項目、数字を入力する項目をそれに適した入力モードにセットすることがアプリケーションプログラムの側でできるようになる。

```
/*** 仮想キー打鍵　vkey_put 使用例 ***/
/* ATOK が組み込まれている事を仮定しています */
#define CTRL_XFER 0xB500
main()
{
        char name[20];
        printf("あなたの名前は？");
        vkey_put(CTRL_XFER);  /* 漢字モード */
        scanf("%s",name);
        vkey_put(CTRL_XFER);  /* 英字モード */
        printf("¥nあなたは %s さんですね．",name);
}
```

拡張機能④のタイムスタンプ付きの打鍵データ記録を解析するために、それを実時間で再生する ks_play(time) を用意してある。この関数を実行すると、timeで指定した時間後にリングバッファ内にある打鍵データの実時間再生を開始する。再生は、リングバッファ内のデータが空になるか、再生終了マークのデータをリングバッファから読み出すまで続けられる。再生モードであることを確認するには関数ks_sw_get()を、強制的に再生モードを終了させるには関数ks_sw_put(ks_sw)を使用する。この再生モード動作中は、リングバッファに打鍵データの記録は行われない。しかし、キーボードからのハードウエア割り込みは受け付けるので、シフトキーやカナロックキー等の特殊キー（キーボードBIOSの内部状態を変更させるキー）を打鍵すると、完全な再生動作は保証できなくなる。また再生モード開始状態と再生データの開始時点でのキーボードの状態（カナロックのon/off等）が異なると正しい再生は得られない。この様に、ks_play()は動作条件が厳しく扱いにくいものであるが、その機能は強力で、例えばオンラインマニュアルを一歩進めた「実時間動作提示型マニュアル」といったアプリケーションの実現を可能にする。つまり、文書でユーザに知識を伝達するのでなく、実際に動作させてデモンストレーションにより、直接、知識を伝達する方式を可能にする。

⑥ 異なるキー配置のキーボードをエミュレートする機能

本システムを使用して、「課題作業を行っている時の人間の反応時間の測定」を行ったところ、ある被験者から特異な結果が得られた。その原因は、被験者の使い慣れたキーボードの文字配置とPC-9801のそれが微妙に異なっていたことのようであった。具体的には、「（）＝＋」等の特殊キーの配置が異なっていることが、心理的ストレスを引き起こす原因と推測された。実験の目的は、「作業内容の心理的困難さ」を反応時間により測定する事であったため、キーボードの操作性に依存する反応時間の遅れは極力減少しなければならない。上記問題を解決するために、本システムでは各種のキーボードをエミュレートする機能をサポートすることにした。

本システムでは、PC-9801のキーボードBIOSに代わって、本システムのキーボードBIOSがキーボードから送られてくる物理キーコードを文字コードに変更する処理を行うことで上記機能を実現している（図５）。本システムでは、「SONYのNEWSのキー配列」、「DVORAC配列」及び「拡張ＳＫＹ配列」（暫定版）を標準で用意してある。これらのキーボードのエミュレーションの起動法は標準状態（CAPS、ｶﾅ がロックされていない状態）で、CTRL－SHIFT－GRPHの３つのキーを同時に打鍵することによりおこなう。一回このモードに入ると、標準状態が「SONYのNEWSのキー配列」に、カナロック状態が「DVORAC配列」に、カナ＋CAPSキーロック状態で「拡張ＳＫＹ配列」のキーボードになる。再度、通常状態にCTRL－SHIFT－GRPHの３つのキーを同時に打鍵するとPC-9801標準の「QWERTY配列」のキーボードにもどる。

「標準ＳＫＹ配列」では文字ｃ、ｊ、ｌ、ｑ、＠等が入力できないので、本システムで採用した「拡張ＳＫＹ配列」（暫定版）ではＶのキー（QWERTY配列の¥）を２ストローク用の特殊キーとし、２ストローク目に打鍵したキーのQWERTY配置の文字コードを入力できるようにしてある。例えば、「ｃ」を入力したい場合には「標準ＳＫＹ配列」の「ｖｚ」（QWERTY配列の「¥ｃ」）と打鍵する。大文字の「Ｃ」を入力したい場合には、シフトキーを押したまま「ＶＺ」と打鍵する。「ｖＺ」「Ｖｚ」の様に途中でシフトキーを変化させると特殊キー「ｖ」「Ｖ」が無効になり２ストローク目の「Ｚ」「ｚ」がそのまま入力文字となる。

これ以外のキー配列をエミュレーションする場合には、アプリケーションプログラムで「キーコード-文字列」の対応表を作成しシステムにその対応表を使用するように指示を与えることにより可能になる。対応表の作成法、使用方法の詳細は省略する。

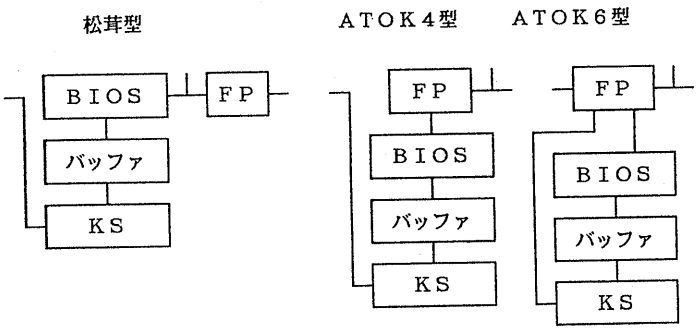

図５、本システム（ＫＳ）の組み込まれ方

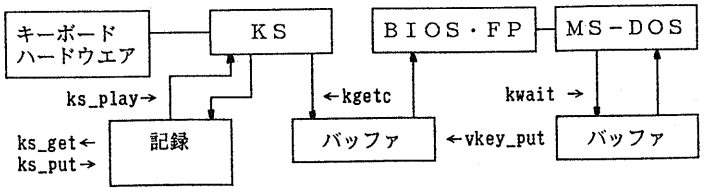

図６、本システムの拡張機能の作用する位置

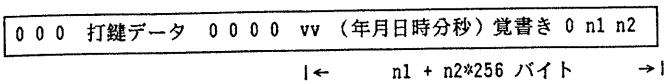

図７、ＫＳ．ＤＡＴの記録形式

３、打鍵データ収集／表示ソフトウエア

当然の事ながら、市販のソフトウエアは本システムを意識して作られていないので、市販のソフトウエアが動作中に本システムで得られるタイミング情報を利用した制御はできない。しかし、市販のソフトウエアが動作しているときの様子を後から収集して、解析することはできる。そのため本システムが収集した打鍵データをMS-DOSのファイルに記録するソフトウエアKSSAVEと、その記録を表示するソフトウエアKSTYPE、実時間で再生するKSPLAYを作成した。なお、ここで扱う記録ファイル名は、カレントドライブのKS.DATとし、使用者が必要に応じてrenameコマンド等でファイル名を変更して保存するものとする。KS.DATの記録形式は、図７の通りで、「ヘッダ、打鍵データ、トレーラ、

```
(1)     RET          63    63
(2)     m           390   327
(3)     i           400    10
(4)     f           409     9
(5)     e           418     9
(6)     s           426     8
(7)     space       441    15
(8)     g           475    34
(9)     e           486    11
(10)    t           498    12
(11)    t           510    12
(12)    i           530    20
(13)    m           553    23
(14)    e           564    11
(15)    .           585    21
(16)    c           601    16
(17)    BS          676    75
(18)    BS          769    93
(19)    j           917   148
(20)    x           932    15
(21)    w           948    16
(22)    RET         967    19
 ---  以下省略  ---
```

図８．２　ＫＳＯＴＹＰＥによる出力例

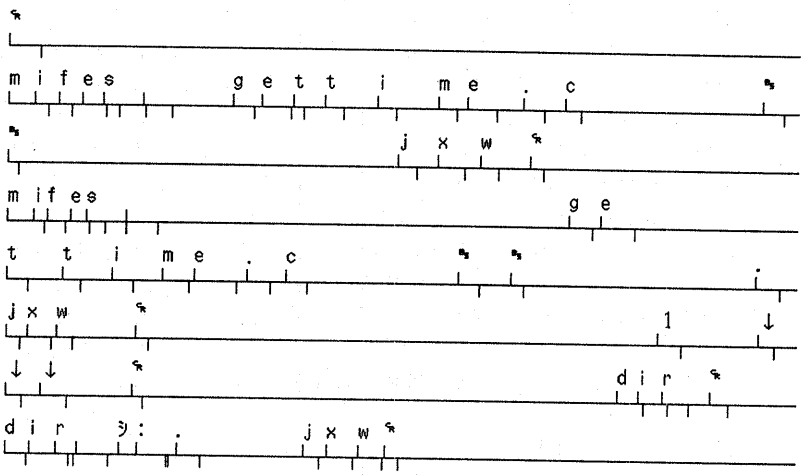

図８．１　ＫＳＴＹＰＥの出力画面例

```
   0: 63 」
  63: 327 mifes gettime.c,75 BS,93 BS,148 jxw」
 967: 406 mifes ,168 ge,113 ttime.c,65 BSBS,93 .jxw」
2084: 198 f1↓↓↓」
2383: 184 dir」
2602: 248 dir ｼ:ｼ/.jxw」
2994: 169 」
3163: 64 kssave」
total 72 key-strokes.
```

図８．３　ＫＳ１ＴＹＰＥによる出力例

```
   0: 63 」
  63: 327 m,10 i,9 f,9 e,8 s,15 ' ',
      34 g,11 e,12 t,12 t,20 i,23 m,11 e,21 .,16 c,
      75 BS,93 BS,148 j,15 x,16 w,19 」
 967: 406 m,10 i,5 f,9 e,6 s,15 ' ',168 g,12 e,
      113 t,21 t,19 i,19 m,12 e,20 .,15 c,
      65 BS,20 BS,93 .,40 j,8 x,11 w,30 」
2084: 198 f1,38 ↓,15 ↓,13 ↓,35 」
2383: 184 d,8 i,9 r,18 」
2602: 248 d,9 i,10 r,8 ' ',16 ｼ,7 :,11 ｼ/,4 .,48 j,9 x,12 w,10 」
2994: 169 」
3163: 64 k,12 s,24 s,9 a,14 v,20 e,16 」
total 72 key-strokes.
```

図８．４　ＫＳ１ＴＹＰＥによる出力例（連続打鍵時の時間間隔表示）

```
 0    0 1C   3F 9C   4B 2F  186 17  190 AF  195 20  199 97  19E
12  1A2 1E  1AA A0  1AB 92  1B0 34  1B9 9E  1BA B4  1C4 21  1DB
A1  1E3 12  1E6 92  1F1 14  1F2 94  1F6 14  1FE 94  205 17  212
97  219 2F  229 AF  230 12  234 92  23F 31  249 B1  251 2B  259
AB  25F  E  2A4 8E  2AC  E  301 8E  305 23  395 A3  39C 2A  3A4
AA  3AE 11  3B4 91  3BB 1C  3C7 9C  3CC 2F  55D 17  567 AF  56B
  ---  以下省略  ---
```

図８．５　ＫＳＤＶＯの出力例（ＫＳＥＶＯの入力例）

コメント、コメント長」の５つの部分から成っている。これは、木村ら[2]の記録形式と第４要素であるコメント部の使用法を除けば互換性がある。また「コメント、コメント長」は省略してもかまわない。本システムで使用しているコメント部は、先頭２バイトにシステムの種類、不定長の覚書の順である。なお、システムの種類がバンクメモリ使用システム(BMKS)の場合には覚書の直前に６バイトの作業開始年月日時分秒を記録してある。

表示するソフトウエアKSTYPEには、グラフィック画面に表示するKSTYPE（図８．１）、１事象１行にして文字列として表示するKSOTYPE（図８．２）、複数事象を１行にして表示するKS1TYPEがある。KS1TYPEでは、解析のしやすさ、見やすさの為に、改行キーの直後、長時間（2.5秒）以上の沈黙、RS-232C回線からのESCマークの直前で改行をするようになっている。更に、連続打鍵（打鍵間隔0.5秒以下）の場合は打鍵間隔時間の表示を省略して表示する（図８．３）。勿論、指定により、表示させることも可能である（図８．４）。

実時間再生ソフトウエアKSPLAYは、副作用を伴わない再生、つまり打鍵したのと同じタイミングで表示するモードと、副作用を伴う再生モードを選択できる。副作用を伴う再生モードを選択すると、キー入力に応じた反応

をコンピュータが示すので、色々な応用が考えられる。例えば、アプリケーションプログラムのデモンストレーションを行なったり、オンラインマニュアルを一歩進めた「実時間動作提示型マニュアル」、アニュメーション、音楽の自動演奏等が可能になる。

KS.DATの一部分をKSPLAYで再生させたい場合や、加工したKS.DATを再生させたい場合にそなえ、KS.DATをテキストエディタで扱える文字列情報に変更するソフトウエアKSDV0（Decorder V-0.0）とそれを元のKS.DATの形式のデータに戻すKSEV0（Encoder V-0.0）を用意した（図８．５）。

4　応用例　--大型計算機のＴＳＳ利用時の応答特性の測定--

本システムはMS-DOS上で動作するソフトウエアならほぼ制限なしに使用でき、そのソフトウエア使用時の人間、機械双方の応答特性を測定できる。ここではPC-9801を端末エミュレータとして働かせることにより、大型計算機の応答特性を測定する応用例を示す。これは、「遅くて使いにくい」と一部で指摘されている状況を確かめる意味と、その原因がどこにあるかを明らかにする意味がある。

測定に際し特別に必要なものは無く、使用するソフトはMS-DOS上で動作する端末エミュレータだけである。今回は、VT100エミュレータを使用した。測定は、次の様な手順で行う。MS-DOS立ち上げ後、本システムを起動して機能拡張を行う。端末エミュレータを起動させ、そのエミュレータの上で測定したい大型計算機を使用する。作業終了後、端末エミュレータを終了させる。この時点で、大型計算機から通信回線を通して送られてきたデータ全てと、直前に行ったキー操作データがPC-9801のメモリ上にタイムスタンプ付きで記録されている。それをデータ収集ソフトKSSAVEを利用して、MS-DOSのファイルKS.DATに記録する。図９にこの記録の一部を表示ソフトKS1TYPEにより出力した結果を示す。これによると、この大型計算機の応答時間（キーボードからの入力に対してそれが受理された合図を送るまでの時間）は0.07秒程で満足できる水準と考えられる。しかし、表示画面の更新（ページ送り）時間は、コマンド受理後表示開始まで約0.7秒位であるが通信回線のスピードがネックになっていて一画面表示が終了するまで約９秒（終了時刻279.98-コマンド起動時刻271.16）かかっている。そのため、ユーザは画面表示が終了し、次のコマンドの入力ができるまで「機械によって待たされている」状態が続いていると考えられる。このことは、表示終了後カーソルがコマンドラインに来た時刻270.53において、ユーザが待たずして次のコマンドを起動している様子から観察される。

原子力規制委員会の「制御室設計レビューの指針」[3]の勧告によると、「画面の更新のスピードは、0.5－１秒以内が限度である」とある。もし、ユーザが画面内容を先頭から読むのであれば、「読み始める行動」がこの勧告の時間以内に開始できれば良いと考えられるが、ユーザが目的のページを検索するのであれば、「画面全体の更新」が勧告の時間以内で終了すること、それも0.5秒以内であることが望ましいと考えられる。つまり、ユーザの行動が前者の場合、今回の大型計算機の応答時間は表示開始までの時間0.7秒と先頭の行（空行でなく文字の含まれる行）の表示終了時間0.4秒の和1.1秒はほぼ勧告の範囲内と考えられるが、後者の場合は勧告の範囲内とは考えられない。つまり、この大型計算機が「遅くて使いにくい」と評価されたのはユーザが後者の使い方していた時であろうと推測できる。もし、この大型計算機で勧告を満足する為には、表示開始後0.3秒以内に全画面を表示できる通信回線が必要となる。おおまかな計算では、1200bpsで約９秒かかっていたデータを0.3秒で転送するのであるから、通信容量は30倍の36Ｋｂｐｓが必要と計算される。あるいは、一回の応答で転送するデータ量が現在の約1/30である必要がある。そのためには、現在の約1/30のデータ量の転送で作業できる新たなソフトウエアの開発が必要になる。

```
27053: 1 ^[[02;15H              /* カーソルをコマンド行に移動 */
       38 ESC,18 8               /* 次のページ表示要求（ユーザの入力） */
27116: 7 ^[[24;80H               /* コマンド受理の合図 */
27129: 62 ^[[01;65H220           /* 内容表示開始 */
27200: 0 ^[[03;06H37
27208: 1 ^[[03;10H177  ALLOCATED TO FT07F001
       -----  省略  ----
27970: 1 ^[24;10H177  ALLOCATED TO FT07F001
27998: 1 ^[[02;15H,49 ESC,18 8
       -----  省略  ----
28971: 1 ^[[02;15H               /* カーソルをコマンド行に移動 */
28977: 662 m,4 M,103 ESC,47 8    /* 最終ページ表示要求（ユーザの入力） */
29793: 6 ^[[24;80H               /* コマンド受理の合図 */
29805: 68 ^[[01;65H371           /* 内容表示開始 */
29881: 1 ^[[02;15H
29889: 1 ^[[03;02H
29895: 1 ^[[KI6403.UGHOURCH.DATA(CH356)
29920: 0 ^[[04;02H
       -----  以下省略  ----
```

図９、大型計算機の応答特性

5　終わりに

使いやすいシステムを作るには、タイミング情報を制御できる機能が不可欠である。人間の発するタイミング情報の有効利用はこれからの研究課題であるが、システムの反応への有効利用は現在でも可能と考えられる。

勿論、シングルタスクでしかもシングルユーザのコンピュータであれば、本システムで拡張した機能を利用しなくても、適切な応答時間をアプリケーションプログラムの側で実現することは可能である。つまり、速すぎる処理を適切なタイミングにするための「時間待ち」の処理を随所に入れるたり、時間がかかりそうな処理の場合には、あらかじめ適当な所にメッセージを出力する命令を埋め込む事により対処する方法である。しかしこの手法は機械依存性が高いため、コンピュータの処理能力が異なれば「適切な応答時間」を示さなくなってしまう。パーソナルコンピュータのカタログに「クロック周波数 XX ＭＨｚで高速な処理を実現すると共に、従来の機種で開発されたプログラムの移植性を高めるために、クロック周波数 YY ＭＨｚでも動作可能です。」が堂々と特徴として上げられていることからも、このことは明かである。さらにこの手法は、マルチタスクやマルチユーザのシステムでは当然通用しない。

つまり、時間情報のようにコンピュータのハードウエアの性能に影響される度合の高い機能は、アプリケーションプログラムの内部で解決するのでなく、本システムのようにコンピュータシステムのＯＳレベルでサポートすべき機能であると考えられる。

本システムによって拡張された機能を使用して、多くのアプリケーションプログラムがフレンドリーなインターフェースを実現することを望む。

参考文献

[1]　川村清：ＰＣ－９８０１解析マニュアル「第０巻」、秀和システムトレーディング、（1983）

[2]　粕川、木村：パーソナルコンピュータ用打鍵データ収集プログラムとその応用、情報処理学会第３０回全国大会、３Ｇ－８（1985.3）、pp.1645-1646

[3]　「制御室設計レビューの指針」（原子力規制委員会資料）、ＩＳＯ／ＴＣ１５９人間工学国内委員会資料３－８、（1986.7）

[4]　「人間と機械の相互作用」、ＩＳＯ／ＴＣ１５９人間工学国内委員会資料３－５－２、（1986.7）

[5]　ＩＯＳ－１０　ＲＡＭディスク＜ＭＳ－ＤＯＳ版＞取扱説明書（E-01版）、（1986.8）

[6]　森川　治：「ユーザフレンドリーなマンマシン対話についてに一考察」情報処理学会第３３回全国大会、２Ｍ－３（1986.10）、pp.1313-1314

[7]　森川　治：「人間とコンピュータの相互作用過程」、テレビジョン学会技術報告、ｖｖ１８０－４（1986.11）、pp.19-24