愛知教育大学教育実践総合センター紀要
第5号，pp. 61～70 (March, 2002)

ジョイスティックによるキーボード反応特性の同定

清　水　秀　美
（愛知教育大学　教育実践総合センター）

Identification of Response Characteristics of Keyboards by Joysticks

Hidemi SHIMIZU
(Center for Research, Training and Guidance in Educational Practice, Aichi University of Education)

要約　心理学実験で反応パネルとしてPCキーボードを用いる場合，そのキーボード反応特性の同定は避けて通れない。そのために過去に提案された方法はPCとは独立に外部測定装置を用いてそれをチェックすることであった。しかし，多種機種の流通と，目まぐるしく変わるモデルチェンジは簡便で正確なチェック方法の開発を必要とする。ここで提案される方法は，PCに測定機能を持たすためにPCクロックの精度をμsecまで高めること，および時間精度の高いサウンド・ボードのMIDIポートに接続されたジョイスティックの活用とから成り立っている。この方法で得られたデータはキーボード・スキャン周期時間と平均キーボード遅延時間について，統計的処理により，確度の高い推定を可能とする。

Keywords：REACTION TIME, INTERRESPONSE TIME, KEYBOARDS, INSTRUMENTATION

今日，パーソナル・コンピュータ（PC）を用いて，キーボード（KB）を介して反応時間データを収集することがしばしば行われている。さらに，PCの低価格化によって，多くのコンピュータ教室が出現し，そこを準実験室として多くの人を対象に一度に実験を行うことが可能になってきている。一方，そこに設置されているPCは場所により，メーカーやモデルが異なると同時に，PCのリプレイス・サイクルも，ますます短くなってきている。

しかしKBを用いる場合，留意すべき三つの遅延要因がある。その第1の要因はキーを押し始めてキーが実際にONするまでの「物理的時間遅れ」である。第2の要因は，キーONを，スキャニングによってKB自体が反応を検出するのに要する時間的ばらつきである(Fay, 2000)。すなわち，スキャニング信号とキーON信号が同時であれば，時間遅れ0で検出されるが，その信号の直後であれば次の信号まで待たされることになる。第3の要因はキーボードによって検出された信号が文字コードに変換され，CPUに送信されるのに要する「一定遅延時間」である。この三つの要因によってKB総平均遅延時間が決定され，一方KB誤差分散は第一と第二の要因によって決定されると考えられる。

従って，心理実験データをKBを用いて収集した場合は，そのデータの信頼性を確保する上から，使用したKBの特性を明らかにする必要がある。そのために外部ユニバーサル・カウンター（UC）などで，キーボードを介して得られたデータとの同時比較によってキーボードに起因する分散と，KB総平均遅延時間の割り出しが求められてきた。Segalowitz & Graves (1990) は，キーボードを用いて信頼性のある反応データを収集するには，PCとは独立に外部の正確な測定装置を用いてそれに伴う誤差と遅延時間を測定し，論文発表に際しては測定精度とデータ修正について述べるべきであると強く勧告している。

勿論，研究者の中にはKB特性の同定は必要でないと考えるものもいる。心理実験の多くの領域では記録された反応時間の分散に比べて，KBの分散が無視できるほど大きい現象も多い（e.g., lexical decision)。しかし新しいタイプのKBが開発されているので，例えばUSBを含む全てのタイプのKBの分散が果たして記録される反応時間の分散に対して無視できるのか疑ってみる必要があろう。また他の研究者との実験結果の比較において，KB総平均遅延時間によるデータの修正は欠かせない。それとは対照的に，行動メカニズムの解明に携わる研究者は非常に高い精度を求めるので，実験室でKBに代る特殊な装置を導入することになる。しかし，本格的な実験に入る前に，コンピュータ教室で一度に多くの人を対象に予備実験を試みることがある。このような場合KB特性を知ることは助けになる。勿論，タイピングの研究は言うまでもなく，キーボードの使用が不可避な実験や，その使用が便利な場合は言を待たない（e.g., Pashler, H. 1994; Verwey, W.B. & Dronkert, Y. 1996)。あるキーの打鍵から次のキー打鍵までの時間間隔の測定値に含まれる誤差分散は，理論的には，反応時間測定のKB誤差分散の2倍になるので，特にKBスキャニングによる変動には注意が必要である。Ulrich & Giray (1989) はRT測定においてコンピュータに内在する測定時間誤差は有意味な結論を導くのに決定的な障害にならないと述べている

が，連続反応時間（IRT）測定については今後の課題である。

外部測定装置によるKB特性の同定の必要性は認識されるとしても，今日の著しいPCのハードの進化は，Graves & Bradley（1987）の同定技法を実行するのに困難をもたらしている。その第1は規格化されたPCマザーボードの集積化のために，容易に外部からプローブ端子を接合できないことである。スピーカーもまたマザーボード上に配置されて，その裏は鉄板で覆われている。第2は最良のKBを多くの中から迅速に選択し，またどのような場所でもKB反応特性を同定できなければならない。従って，できればKB反応特性の検査のためにUCやオッシロスコープや，データ記録装置の搬入と設定を回避したい。これらの問題を解決するために提案される方法は，KBの繋がれたPCそれ自体を計測装置として機能させることである。すなわち，KBと異なり時間遅れが1msec以下のジョイスティックを用いて（Graves & Bradley, 1987），PCのブザーONを始点に，ジョイスティックONを終点として，その両時刻をコンピュータ内臓のクロックとタイマーで読み取り，差を求め時間間隔を求めるのである。

本論文の第1目的は，こうして求めた値が外部のUC値とどの程度一致するかを多くのPCとKBの組み合わせについて明らかにすることにある。第2の目的は，もしUCの代わりにジョイスティックを用いることが正当化されるならば，実際に多くのPCとKBを用いて，音信号が提示された後KB反応が生じるまでの時間間隔を，ジョイスティックを介して測定された時間とを比較する（RT条件）。これによってKB総平均遅延時間とKB誤差分散を明らかにする。第3の目的は連続打鍵間隔をKBでの時間とジョイスチックでの時間とを比較し，この場合のKBによる誤差分散を明確にする（IRT条件）。

予備的作業と使用ハードウェアの概略

本方法を実現するには，予め解決すべき問題がある。

予備的作業　その第1段階はPCにUCとしての機能を持たすために，PC上にμsecオーダのクロックを実現する必要がある。従って，(a)高精度のクロックの開発と，(b)その精度評価が欠かせない。

第2段階は，KBスキャニイングによるKB誤差分散の推定を容易にするために，キーの「物理的遅延時間」の分散をできるだけ小さくすることである。できるだけ打鍵速度を一定にするという目的で簡単な道具，スプリング・ペン（P-Pen）を作成した。この道具によってキーに目標を定めるという操作が省略できるので，上限の速度でキーを打つという操作に専念できる。これによって物理的遅延に伴う分散を低く抑えることができると考えられる。勿論KBの種類に応じて平均遅延時間と分散は変わると思われるが，それほど大きな差異はないと思われる。平均時間遅れは別として，その分散はKBスキャニイングによる分散に比較して無視できるほど小さいかどうかを明らかにする。（もし一定の速度で目標キーを打鍵できるのであれば，S-Penに代えて鰐口クリップ等を用いても良い。）

ハードウエアの概略　第1段階として，ジョイスティック関連についての概略を述べる。

(a)MS-DOSで動作可能でかつPCI接続のジョイスティックとMIDI Boardを用意する。正確な時間計測は本質的にはマルチタスク用のOS（e.g., Windows, Linux, MacOS, Unix）では可能ではない。ある作業に振り向けられたコンピュータ・リソースを当該作業に振り向けるには一定時間が必要である。従って必要な時に必要な時刻が待ち時間0で得られる保証がないからである。またPCI接続は並列データ転送を実現し，KBのシークエンシャル・データ転送と異なり，転送による遅延時間の低減を実現する。

(b)S-Penでキーを打鍵する一動作がジョイスティックのボタン“A”を押すことと，キーを押すことをもたらすようにする。ジョイスティックのボタン“A”のバイパスボタンを構成して，S-Penの先端とキー上に貼られた銅箔との接触でボタン“A”が押された状態になるようにした。この結果，S-Penでのキー打鍵によってジョイスティックからの信号とKBからの信号がCPUに転送されることになる。

最後に第2段階として，ジョイスティック使用の妥当性を検証するために一組の外部測定装置と，その測定のための手続きについて述べる。

(a)その一つはジョイスティックによる時間計測の正確性の程度を調べるために，UCとそのデータをGP-IBを介して自動的に収集するためのPC，および電気信号波形モニターのためのオッシロスコープを用意する。

(b)配線に関しては，音刺激を提示する，PCのマザーボード上のスピーカーの電圧変化をピックアップする導線をプリント基板上に接続し，ここからの信号でUCをスタートさせ，一方S-Penによるキー打鍵で生じるジョイスティック電圧変化でUCをストップさせるようにした。UCで計測されたデータは自動的に記録用PCに記録される。

ジョイスティックの信頼性が確認されればこの第2段階は省略される。マザーボード背面の鉄板を切り取って導線を接続することも，高価な測定装置一式の導入も必要でなくなるであろう。それに代わって，DOS対応の約8千円のMIDI Boardと約千円のジョイスティックでKBの反応特性が同定可能になる。また，DOSのシステムはWindows 9x/Meに組み込まれているので，それをベースに実行フロッピー・ディスクを作成すればよい。以下に，以上の1と2段階をさらに詳

述し，そのあと本論文の本題に入ることにする。

Step 1-（a）高精度PCクロックの開発

KB反応特性を調べるにはmsecオーダでなく，μsecオーダの測定が可能でなければならない。すでに，Smith & Puckett（1984）は8253タイミング・チップからμsecで時間情報を得る方法を機械語で実現している（82C54のように，8253の上位互換モデルであれば問題はない）。この方法をGraves & Bradley（1987）は，BASIC言語で利用できるように書き直しmsecオーダの測定を可能にしている。

ここでも，彼等の思想を受け継いで，それをC言語で書き直し，理論値と同じ精度±（1/1193180）sec=±0.8380965μsecが得られるようにした。実際のコーディングではBorland社製の“Turbo C++ version 4.0J for DOS”を用いて，BASICでは煩雑だったアセンブリ言語との接合を簡潔に行った。プログラミングに際しては入出力割り込みを回避して，時間測定を安定に行うために，実験終了後に周辺記録装置に出力するように心掛けた（舟川，1988, p.82）。また，コーディングに際してはGraves & Bradley（1987）の他に，芦達（1995）と中島（1997）を参考にした（APPENDIX参照）。

Step 1-（b）クロックの精度評価

装置と動作環境　作成したプログラムを，キーボードFMV-KB321を備えたAT互換機の富士通社製FMV-5166T3で実行した。使用したOSはDOS version 6.2/Vである。また，PCで得られた時間測定値を評価するための装置として，岩崎通信社製のユニバーサル・カウンタSC-7203（UC）を用いた。

精度評価　実際に正確な時間測定が可能かどうかを調べるために，周期的なディスプレイの垂直同期信号間隔を測定することにより検証した。ディスプレイの1フレーム表示に要する正確な周波数特性を調べるためにフォトダイオードをセンサーとして，これをディスプレイ管面上に装着し，出力端子をUCに接続して，周波数59.662 Hzを得た。この周期間隔は16.76109 msecである。一方，実測値は16.76106 msecを最頻値として16.76190msecと16.76023 msecの3値をとり，レンジは±0.84μsecであった。これは理論的に予測される誤差と一致する。

Step 2　キー接触からスイッチONまでの経過時間

次に解決すべき問題は，キー表面に接した後，キー・スイッチがONになるまでの時間のばらつきの程度をできるだけ小さくおさえるため，可能な限り高速で正確に打鍵できるための道具を作成した。これは図2に示すようなスプリングを組込んだ簡単なペン形状を示す（S-Pen）。このS-Penを用いて，図2とは別に，同種のキーボードFMV-K321の打鍵実験を行った。キーの表面に銅箔を貼り，それとS-Penの先端とでスイッチを構成し，接触すると電圧が変化するように抵抗と電池からなる回路を構成した。一方，キー入力が有効になった時点（キー・スイッチON）で，電圧変化が生じるように同様の抵抗と電池からなる簡単な回路を別個に組んだ。これらの2つの電圧変化をUCのスタート，ストップ信号として用いることにより，キー接触からキー入力が有効になるまでの経過時間を調べることができる。図1は500回の試行で得られた経過時間を度数分布図として示したものである（M=2.007 msec, SD=0.259 msec）。従って，分散は0.067であるので，議論を簡潔にするためにこの分散は無視できるものとして取り扱う。

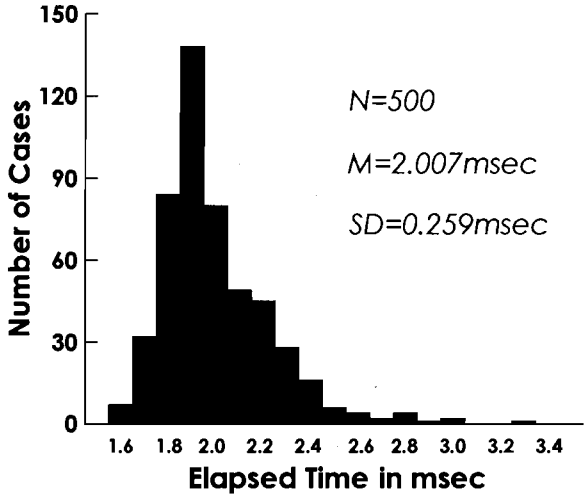

図1．S-penを用いた場合のキー表面接触からキーONに要する時間，「物理的遅延時間」

PCクロックとジョイスティックによる時間計測の信頼性の検証

PCクロックとジョイスティックによる時間計測値が外部UCの値とどの程度一致するかを，スピーカーONからジョイスチックONにいたる時間を測定することで明らかにする。PCでの時間測定ではこの2つの信号間をPCクロックでの時刻差として求められる。ただし，この場合，次の目的であるキーボード反応特性の同定を考慮した実験状況設定を想定して，一回のキー打鍵によって，ジョイスティックON，UCストップ，キーON，が可能になるように配線をおこなった。

装置とその配線の詳細

以下に，ここで用いた装置と配線についてさらに詳述する。

1. UCスタートのためのスピーカーON信号プローブ線の固定。

　この方法が一般性をもつかどうか確かめるために6台のコンピュータを対象に検査を行った。使用コンピュータは表1の通りである。6台のうちの

1台を除いてスピーカ位置に対応するマザーボードの背面の鉄板を切り取り，この穴を通してスピーカー端子に直接プローブ線を固定した。スピーカー音をマイクで電気信号に変換する方法も考えられるが約1.5msecの時間遅延を伴うので理想的ではない。

2. ジョイスティック・ボタン“A”のバイパス・ボタンの作成。

使用したジョイスティックはElecom社製のModel:JC-BD10である（15-Pin対応のものであれば種類を問わない。）導線でジョイスティック・ボタンAの一つの接点とS-Penを結んだ。もう一方の接点に新しいリード線をつなぎ，さらのその線の他端に2枚の銅箔を直列につないだ。この2枚を各々ターゲットキー“5”とダミーキー“8”の表面に両面テープで固定する（図2参照）。その結果S-Penでいずれかのキーを打鍵すると，ジョイスティックONとUCストップ信号が同時に得られ，ついで数msecのキーの物理的遅延時間の後にキーONする。

3. ジョイスティックとインターフェイス・ボードとの接続。

DOS対応のPCI接続の“MIDI Board”であれば種類を問わない。使用Boardsを表1に記載した。これにジョイスティックを接続する。

4. UCとその他の装置。

Step 1-(b)で使用したUCを用いて測定値のチェックを行う。UCデータを自動記録するために，GP-IBインターフェース・ボードSC-0111とNEC PC9821Apを使用した。また，UC入力波形のモニターはHITACHI Oscilloscope V-550で行なった。

手　続

ジョイスティックのデータを比較するUCのデータを得るために，UCのスタートをPCのスピーカ端子電圧変化により行い，UCのストップはキー“5”をS-ペンの打鍵で生じるジョイスティックの電圧変化（4.4 V[DC]から0Vへの電圧降下）で行った（Graves & Bradley, 1987）。KB1個につき，音が鳴り終わったらキー“5”を打鍵する試行100回を1試行ブロックとして，5ブロック行い，ジョイスティックによるPCデータと比較可能なデータ500個が得られた。このような手順に従って，メーカーが提供する6組のPCとKBにさらに新しく1個のKBを加えてデータ収集を行なった。調査したPCとKBの種類を表1に示す。

結果と考察

ジョイスティックから得られた時間計測値とUCから得られた値との差が小さければ小さいほど，ジョイスティックの代替使用の妥当性が高まることになる。ジョイスティックで得られた値から対応するUC値を差し引き，差異値を各KBについて500個求め，各KBについて平均値と標準偏差を求めた（表1）。PCとKBの組み合わせは購入時にメーカーから提供されたものであるが，Fujitsu社製のFMV M3/557ではUSB KBに加えてIBMのPS/2 Port対応KBについてもデータを求

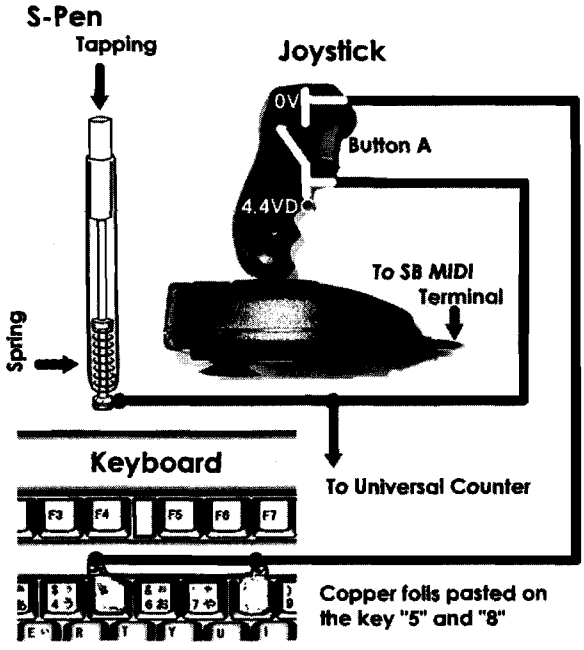

図2. ジョイスティック、S-ペンとキーボードの結線

表1. 検査されたPCとセットとして用いられた“MIDI Board”とKB。数値はこの組み合わせで得られた時間間隔計測値から外部UCの計測値を差し引いた値の平均値（*M*）と標準偏差（*SD*）。

PC Model	CPU(Speed)	PCI MIDI Board	KB model	*M* (msec)	*SD* (msec)
Fujitsu FMV					
5166T3	Pentium(166MHz)	Creative SB AWE32	FMV KB321(PS/2)	0.009	0.093
TV337	Pentium II (333MHz)	Creative SB AWE64	FMV KB321(PS/2)	0.117	0.071
M3/557	Athlon(550MHz)	Yamaha YMF744B	CP018203-02(USB)	0.134	0.062
			IBM 5576-B01(PS/2)	0.036	0.099
Dell Dimension					
XPS B800r	Pentium III (800MHz)	Creative SB Live	SK-800(PS/2)	0.053	0.084
4100	Pentium III (800MHz)	Creative SB Live	SK-800(PS/2)	0.056	0.137
8100	Pentium 4(1.5GHz)	Creative SB Live	RT7D00(PS/2)	0.119	0.073

表2. RT条件でのKBデータとジョイスティック・データの差の平均と標準偏差、及び標準偏差によるKBスキャニング時間の推定値。

PC Model	CPU(Speed)	KB model	KB Scanning Period (Period Time: msec)	*M* (msec)	*SD* (msec)	Estimated Period Time
Fujitsu FMV						
5166T3	Pentium(166MHz)	FMV KB321(PS/2)	171.6Hz(5.83)	14.418	1.709	5.92
		IBM 5576-B01(PS/2)	97.1Hz(10.30)	23.485	2.911	10.08
TV337	Pentium Ⅱ(333MHz)	FMV KB321(PS/2)	171.6Hz(5.83)	15.794	1.694	5.87
M3/557	Athlon(550MHz)	CP018203-02(USB)	100.4Hz(9.96)	72.936	5.419	18.77#
		Mets CK-006(USB)	244.2Hz(4.10)	27.553	9.453	32.75#
		Mitsumi KPQ-EA9EA(PS/2)	87.4Hz(11.44)	29.614	3.140	10.88
		RT7D00(PS/2)	338.1Hz(2.96)	23.046	0.873	3.02
		SK-800(PS/2)	233.8Hz(4.28)	12.433	1.215	4.21
		IBM 5576-B01(PS/2)	97.1Hz(10.30)	24.755	2.937	10.17
Dell Dimension						
XPS B800r	Pentium Ⅲ(800MHz)	SK-800(PS/2)	233.8Hz(4.28)	10.785	1.211	4.20
4100	Pentium Ⅲ(800MHz)	SK-800(PS/2)	233.8Hz(4.28)	10.816	1.204	4.17
8100	Pentium 4(1.5GHz)	RT7D00(PS/2)	338.1Hz(2.96)	21.230	0.836	2.90
		SK-800(PS/2)	233.8Hz(4.28)	10.614	1.221	4.23
		IBM 5576-B01(PS/2)	97.1Hz(10.30)	23.245	2.951	10.22

#USB KB であることに注意。

めた。表1から明らかなように各差異平均は約0.1 msecであり，各標準偏差も約0.1 msecである。従ってジョイスティックでの測定誤差はKBスキャニング変動誤差に比較して無視できる。ジョイスティックはDOS/VマシンにおいてUCの代替となりうることを示唆する。

RT条件でのジョイスティックによるKB反応特性の同定

スピーカONからキーONまでの時間間隔をKBで測定するとき，測定値に含まれる平均時間遅延とKBスキャニングによる変動を，ジョイスティック・データとの比較で明らかにする。具体的には，KBによる測定値とジョイスティックで得られた値との差異を検証する。

装置と手順

ジョイスティックの使用することの信頼性の検証のところで述べたのと同じである。しかし，使用したPCは同じく6台であるが，KBとPCの関係を明らかにするために，新しくKBを付加したこと，およびKBを相互に交換してデータを追加収集した。

RT条件での結果と考察

ここでも500対個のジョイスティック・データとKBデータが得られた。各KBについての，対応するジョイスティック・データ値とUCデータ値の差異についの平均値と標準偏差は表2の通りである。

表2のKBスキャニング周波数は次のような論理と方法によって記録された。スキャンパルスはキー打鍵でキー接点に現れる。従ってキーを押し続けると一群のパルス列がその接点に現れることになる。従ってこのパルス列の周波数がスキャン周波数を意味する。使用した全てのIBM KBを除いたKBは，2枚のプリント回路とそれを隔てるスイッチ接点部分を円形に切り抜いたシートから構成されている。従ってリード線の先にハンダ球を形成して短絡状態を保つように，任意のスイッチの両接点部に接するように挿入し，他方の導線の端をUCに接続した。

KBによる時間測定に伴う変動誤差がKBスキャニングによるとすれば，差の標準偏差から上記の方法で求められたKBスキャニング間隔（1000 msec/[KB Scan Period]）が推定されるはずである。KBスキャニングによる差の値の分布は理論的には一様分布であり，一方，実際のデータの散らばりも一様分布であると考えられる（図3参照）。従ってデータ値のSD_{rt}から$R_{rt}^2 = 12SD_{rt}^2$により推定値が求まる（林，1985）（R_{rt}は一様分布のレンジを意味する，すなわちKBスキャン間隔を意味する）。表2に示すように，2つのUSB KBを除いて全てのPS/2 KBsデータから求めた推定値はKBからハード的に直接測定した値とほぼ一致する。

一方，PS/2 KBの総平均遅延時間は，以前に述べたようにキーの「物理的遅延時間」，「スキャンによる遅延時間」と「CPU送信時間」の合計から成る。従ってここでもとめた平均からキーの「物理的遅延時間」に相当する約2msecと，一様分布の平均はレンジの中央にあるからスキャン周期間隔の二分の一とを差し引いて，「文字コード変換とCPU送信に要する時間」を

知ることができる。しかし，データ補正の観点から総平均遅延時間を知れば十分であろう。実際には得られたRT時間測定値からこの遅延時間分を差し引いて補正を行なうことになる。

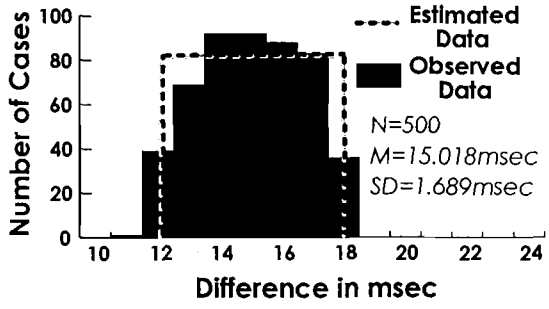

図3. PC FMV-5166T3を用いた場合のKBとジョイスティックによる反応時間（RT）測定値の差の分布

IRT条件でのジョイスティックによるKB反応特性の同定

あるキー打鍵から次ぎの打鍵までの時間間隔のKBデータは，ジョイスティック・データとどの程度食い違うのであろうか。RT条件での平均遅延時間の影響について考えると，あるキー打鍵の時刻はRT条件でのデータから明らかなように，KBを介して得られる実際の時刻はKB誤差変動を伴い平均遅延時間だけ遅れることになる。同様に次ぎの打鍵の時刻も同じくその変動を伴い一定時間遅れる。したがってIRT条件でのKBデータとジョイスティック・データの差について，平均値は0で，時間間隔の差のデータは2倍の分散部分だけが残ると想定される。またKBスキャンの分布は一様分布だから，その分布は一様分布から2標本を取りだし，その差の分布になるので三角分布になると想定される。このことを，ジョイスティック経由の時間測定で確かめる。

装置と手続

装置の詳細はRT条件と同じである。しかし，IRT条件での測定では音提示でキー“8”をS-ペンで101回連続打鍵し，キー“5”の打鍵で終了する。5ブロックで500個の測定値を得た（始めの反応はRT反応に相当するので除外）。またジョイスティックの信頼性の検証のために用いた6台のPCと7つのKBを対象に検査を行った。

IRT条件での結果と考察

キーを連続的に打鍵した場合の時間間隔を，KBを用いて測定した値からジョイスティックを用いて測定した値を差し引き，その平均値（M）と，標準偏差（SD）を求めた。表3から明らかなように平均値は0の近傍に位置している。この分布がKBスキャニングに起因する一様分布から，2つの標本を抽出して得られるとすると，それは三角分布となると想定される（PC FMV5166T3で得られた三角分布を図4に示す）。このSDからKBのスキャニング周期時間（R_{rt}）は式$R_{irt}^2=24SD_{irt}^2$:$R_{irt}=2R_{rt}$を用いて推定される。推定値と実際の周期時間はUSB KBを除いてほぼ一致する。以上によりPS/2 KBでは連続キー打鍵をKBを介して時間測定する場合に伴う分散は$SD_{irt}^2=R_{rt}^2/6$と定義される。

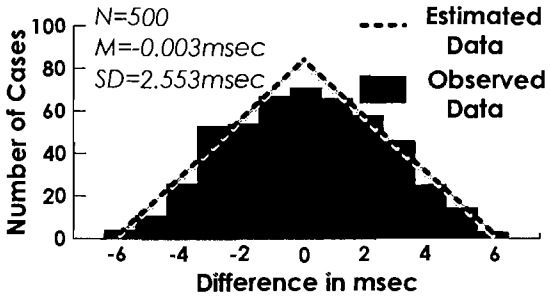

図4.. PC FMV-5166T3を用いた場合のKBとジョイスティクによる反応間隔時間（IRT）測定値の差

表3. IRT条件でのKB測定値とジョイスティック測定値の差異値の平均と標準偏差。また標準偏差によるKBスキャニング周期間隔時間の推定。

PC Model	CPU(Speed)	KB model	KB Scanning Period (Period Time: msec)	M (msec)	SD (msec)	Estimated Period Time
Fujitsu FMV						
5166T3	Pentium(166MHz)	FMV KB321(PS/2)	171.6Hz(5.83)	-0.008	2.519	6.17
TV337	Pentium II (333MHz)	FMV KB321(PS/2)	171.6Hz(5.83)	-0.306	2.599	6.37
M3/557	Athlon(550MHz)	CP018203-02(USB)	100.4Hz(9.96)	-0.254	7.930	19.43#
		IBM 5576-B01(PS/2)	97.1Hz(10.30)	-0.026	4.133	10.12
Dell Dimension						
XPS B800r	Pentium III (800MHz)	SK-800(PS/2)	233.8Hz(4.28)	0.026	1.760	4.31
4100	Pentium III (800MHz)	SK-800(PS/2)	233.8Hz(4.28)	-0.006	1.666	4.07
8100	Pentium 4(1.5GHz)	RT7D00(PS/2)	338.1Hz(2.96)	0.001	1.156	2.83

#USB KB に注意。

結　び

MIDI boardに結合されたジョイスティックは，UCに代わって，KBを反応パネルとして用いた場合に起こる，遅延時間や誤差分散を明らかにする。Segalowitzの勧告を実行するもう一つの方法を手にした。本論文ではRT条件とIRT条件でKB反応特性を同定しようと試みたが，前者の分散から後者の分散は予測できることと，平均遅延時間は後者では明らかにならないという理由で，RT条件のデータ収集でKBの反応特性は同定できるといえる。

参 考 文 献

芦達　剛（1995）DOS/Vプログラミング・リファレンス　ソフトバンク

Fay, D. Q. M.（2000） http://www.rz.uni-hohenheim.de/sys/basics/csc102/ch25.html

舟川政美　（1988　実験制御技法とマシン語サブルーチン・プログラム集－PET－
阿部純一（編）　パーソナル・コンピュータによる心理学実験プログラミング　ブレーン出版 Pp. 73-213.

Graves, R., & Bradley, R.（1987）Millisecond interval timer and auditory reaction time programs for the IBM PC.
Behavior Research Methods, Instruments, & Computers, 19, 30-35.

林知己夫（1985）確率論・統計学　日本放送出版協会

川村浩也，吉田雅秋（1995）AT互換機アーキテクチャハンドブック　ナツメ社

中島信行（1997）C & C++プログラマのためのI/O制御プログラミング入門　CQ出版

Pashler, H.（1994）Overlapping mental operations in serial performance with preview.
The Quarterly Journal of Experimental Psychology, 47A 161-191.

Ulrich, R. & Giray, M.（1989）Time resolution of clocks: Effects on reaction time measurement － Good news for bad clocks. *British Journal of Mathematical and Statistical Psychology,* 42,1-12.

Smith, B., & Puckett, T.（1984, April）
Life in the fast lane. *PCTech Journal,* 63-74.

Segalowitz, S. J. & Graves, R. E.（1990）Suitability of the IBM XT, AT, and PS/2 keyboard, mouse, and game port as response devices in reaction time paradigms. *Behavior Research Methods, Instruments, & Computers,* 22, 283-289.

Verwey, W. B., & Dronkert, Y.（1996）Practicing a structured continuous key-pressing task: motor chunking or rhythm consolidation? *Journal of Motor Behavior,* 28, 71-79.

APPENDIX

```
// Joystick によるキーボード反応特性の同定プログラム
// 関数 timerSet(), timerRead(), fnt()については Graves & Bradley(1987)を参照した。
// 比較検討できるように変数名は同じにしてある。
// 本文中の Figure 3 は RT=100, IRT=1 に設定して求めた。
// 本文中の Figure 4 は RT=1, IRT=101 に設定して求めた。
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#define   RT          100     //  反応時間(RT)の測定回数
#define   IRT                 1                       //  反応-反応時間間隔(IRT)の測定回数
#define   MCLKSEG             0x40                    //  デイリー・タイマのセグメント
#define   MCLKOFF             0x6c                    //  デイリー・タイマのオフセット
#define   TIMER_0             0x40                    //  8254 Counter 0 port
#define   TIMER_SET           0x34                    //  8254 Counter 0, set to Mode 2
#define   TIMER_LATCH         0x00                    //  8254 Command - カウンタ値のセイブ
#define   TIMER_CTL           0x43                    //  8254 Control Port
//  8254 カウンタ設定関数 ---  カウンタ 0 をモード 2,分周値を 0 に設定
#define timerSet()   outp(0x43,TIMER_SET);outp(0x40,0x00);outp(0x40,0x00)
//  デイリー・タイマ値とカウンタ値を読み込む関数
                                                    //  h=デイリー・タイマ上位 16bit の値
                                                    //  l=デイリー・タイマ下位 16bit の値
                                                    //  r=8254 カウンタ 0 の値
#define tmrRead(h,l,r)   asm mov al,TIMER_LATCH; asm cli; asm out TIMER_CTL,al;
    asm mov ax,MCLKSEG; asm mov es,ax; asm les ax,es:[MCLKOFF];asm mov word ptr l,ax;
asm mov ax,es; asm mov word ptr h,ax; ¥ // 以上 3 行を 1 行で続けてコーディングのこと
    asm in al,0x40;asm mov ah,al;asm nop; asm in al,0x40; asm nop; asm nop; asm sti; asm xchg ah,al;asm neg ax;
    asm mov word ptr r,ax   // 以上 2 行を 1 行でコーディングのこと
//  Joystick-ON の検出
#define joystick()  asm mov dx,0x201;asm joystick_sw1:;asm in al,(dx);asm test al,0x10;
    asm jnz joystick_sw1                            // 以上 2 行を 1 行でコーディングのこと
double fnt(unsigned short,unsigned short,unsigned short);  //カウンタ値を実時間に変換
int    vgaEdgeDetect(void);                         //  垂直同期信号検出
void   timeInterval(int);                           //  msec 単位で時間間隔を設定

void main(void)
{
            FILE *fp;

            doubleIRT+20],je[IRT+20],rt[RT][IRT+1],rt0[RT],jrt0[RT],jrt[RT][IRT+1],s_time;
            unsigned short h,l,r;
            int i,n,k,rep[RT];
            char c[RT][IRT+1];
            char fname[20];

        printf("¥x1b[2J¥n 記録ファイル名を入力しなさい>>" );
        scanf("%s",fname);
                                                    // 実験結果の記録ファイル設定
        if((fp=fopen(fname,"w")) == NULL){
                printf("Floppy Error!!!");
                exit(0);
        }

              timerSet();                           // カウンタ 0 をモード 2,分周値 0 で実行

              for(i=0;i<20;i++){                    // 垂直同期信号間隔の測定
                      vgaEdgeDetect();              //  同期信号の検出
                      tmrRead(h,l,r);               // その時のカウンタ値を読み取る
                      e[i]=fnt(h,l,r);              // カウンタ値を実時間に変換
        }
        printf("¥n 次にディスプレイ1フレーム表示に要する時間を msec で表示する。¥n");
        printf("誤差が±0.00084msec であれば、時間測定環境は整っているといえる。¥n");

                for(i=1;i<20;i++){                  // ディスプレイ 1 フレーム当りの時間の表示
               printf("%10.5lf¥n",e[i]-e[i-1]);
        }
                printf("次の測定に進みますか？ イエスならどれかキーを押してください。");
                getch();                            // 入力待ち
                printf("¥x1b[2J");                  // 画面クリアー
```

```
                                                    //  RT,IRT の時間測定開始
  if(IRT==1){
                printf("¥n 音が鳴り終わったらキー'5'を押せ。中断はキー'8'で可能¥n");
                printf("躊躇することなく、正確に、一気に押すこと！！！¥n");
                printf("      Keybord(KB) Joystick(JSK)  KB-JSK¥n");
        }
        else{
                printf("¥n 音が鳴り終わったらキー'8'を連続して押せ。中断はキー'5'で可能。¥n");
                printf("0.8sec ぐらいの間隔で、確実に押すこと。時間データは表示されません。");
                printf("¥n 試行回数¥n");
                }
                for(n=0;n<RT;n++){
                        if(n>0 && c[n-1][k]=='8') exit(1);   // RT 測定のブレイク
                        timeInterval(1000);     // RT 試行間隔を１秒に設定
                        k=0;                    // IRT 試行回数の０クリアー
                        c[n][k]='s';            // 初期設定
                        tmrRead(h,l,r);         // 開始時間を読む
                        sound(4000);            // 4000Hz 音刺激提示
                        timeInterval(50);       // 持続時間 50msec
                        nosound();              // 音-Off
                        e[k]=fnt(h,l,r);        // Keyboard 実時間格納
                        je[k]=e[k];             // Joystick 実時間格納

                        while(1){               // "5"が押されるまで IRT で許される
                                k++;            // 回数だけ連続記録が行われる
                                while(kbhit()){getch();}        // Keyboard Buffer clear
                                if(k==IRT && IRT>1){printf("¥nTap key '5' !!");}
                                // IRT 測定修了告知
                                if(IRT>1) printf("¥n%3d  ",k);    // IRT 試行回数の表示
                                if(k>IRT) {k-=1;break;}    // 許される IRT 試行回数で終了
                                joystick();         // Joystick-On の時刻取得
                                tmrRead(h,l,r);
                                je[k]=fnt(h,l,r);   // 時刻格納

                                c[n][k]=getche();   // Keyboard 入力文字の取得
                                tmrRead(h,l,r);
                                e[k]=fnt(h,l,r);    // 時刻格納
                                timeInterval(80);   // Joystick 入力終了待ち
                                if(c[n][k]=='5') { break;}  // IRT 試行の中断
          }
                for(i=1;i<k+1;i++){                         // 時刻から時間間隔を算出
                        rt[n][i-1]=e[i]-e[i-1];             // Keyboard での RT 時間
                        jrt[n][i-1]=je[i]-je[i-1];          // Joystick での RT 時間

                        c[n][i-1]=c[n][i];                  // Keyboard 入力文字の処理
                        rep[n]=k;                           // IRT 試行回数の格納
                      // RT 試行での結果表示; 以下の２行を１行で続けてコーディングのこと
                        printf("%3d %c %10.5lf  %10.5lf   %10.5lf¥n",n,c[n][i]
                        ,rt[n][i-1],jrt[n][i-1],rt[n][i-1]-jrt[n][i-1]);
          }
                  rt0[n]=rt[n][0];                          // Keyboard での反応時間の格納
                  jrt0[n]=jrt[n][0];                        // Joystick での反応時間の格納
        }
                                                            // 結果を外部ファイルに出力
        for(n=0;n<RT;n++){                                  // 必ず実験終了後に行うこと
                  k=rep[n];
                  for(i=1;i<k;i++)                          // IRT 試行結果のファイル出力
        fprintf(fp,"¥n%3d %c %10.5lf %10.5lf",i,c[n][i],rt[n][i],jrt[n][i]);
        }                                                   // RT 試行結果のファイル出力
        for(i=0;i<RT;i++) fprintf(fp,"¥n%3d %10.5lf %10.5lf",i,rt0[i],jrt0[i]);
        fclose(fp);
}

//  カウンタ値から実時間算出(msec)
double fnt(unsigned short h,unsigned short l,unsigned short r)
{
        static double e=65536.0,f=0.000838095;
        double dh,dl,dr,t;
        dh=(double)h;
        dl=(double)l;
        dr=(double)r;
        t=(dh*e*e+dl*e+dr)*f;
        return(t);
```

```
}

int vgaEdgeDetect()                          // ディスプレイの垂直同期信号検出
{                                            //  芦達(1995, p.233-237)を参照
       asm {
         mov dx,0x03da
   vga_vsync_11:                             // 同期信号ONを検出
         in al,(dx)
         test al,0x08
         je vga_vsync_11
   vga_vsync_12:                             // 同期信号OFFを検出
         in al,(dx)
         test al,0x08
         jnz vga_vsync_12
          }
          return(1);
}

void timeInterval(int ti)                    // 待ち時間間隔をmsecオーダで設定
{
        unsigned short h,l,r;
        double timeUp;
        tmrRead(h,l,r);
        timeUp=fnt(h,l,r)+(double)ti;
        while(1){
                 tmrRead(h,l,r);
              if(fnt(h,l,r) >= timeUp) break;
        }
         return;
}
```