社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－CE－85 (1)
2006／7／8

Full BASIC 言語処理系開発の現状と課題

白石 和夫
文教大学教育学部

教育用に用いることを目的としてISO Full BASIC準拠のBASIC処理系を開発してきた。Windows版とLinux版がある。最新版ではFull BASICの図形機能単位のほぼ全機能を実現しつつある。本稿では，規格処理系の存在意義と，処理系開発に残された課題について述べる。

Present Status and Problems on Developing Full BASIC Language System

Shiraishi Kazuo
Faculty of Education, Bunkyo University

We have been developing ISO Full BASIC language processing system, which works on Windows and Linux OS. The recent versions actualize almost all the graphics module of ISO Full BASIC. In this report, we denote the present state and confronting problems on developing the system.

1. はじめに

教育用言語は，アルゴリズムを考え，また，それを他者に伝達するための人間が使う言葉としての性格，そして，数理的な事象を処理するための道具としての実用性の２面性を持つことが要求される。Full BASIC[1], [2], [3]はその観点からよく検討された文法を持つ言語である。しかし，現実には，改変された文法を持つ方言 BASIC が教育用に用いられることが少なくない。

また，規格処理系は，言語仕様の詳細が規格で定められているため，論理に従って組み立てていったプログラムが「処理系の仕様」のために意図の通りに動作しないなどの問題を起こす心配がなく，その観点からも教育用として用いるのに適しているといえる。

筆者は，ISO Full BASIC 規格に準拠したBASIC 言語処理系を開発し，配布してきた。内部構造の詳細は，[4], [5]で，開発の意図などは[6], [7], [8]において述べた。本稿では，その後の進展を含め，Full BASIC 言語処理系開発の現状と課題について述べる。

なお，Full BASICは，True BASICを土台にして定められた規格であるが，True BASIC社が配布しているWindows demo版を試す限りにおいて，True BASICは規格との非互換を多く残している。その他，少なくとも筆者が知りうる限りにおいて，筆者の処理系以外にFull BASIC規格をほぼ実現したものは存在しないように思われる。

2. Full BASICの特徴

2.1. 数値

Full BASICは，計算結果の正確さを規定する。これは，コンピュータを計算機として使う観点からは不可欠なことである。

Full BASICでは，十進小数の正確な計算が義務付けられている。計算結果の正しい値が数値変数の精度としてあらかじめ定めた精度の十進小数で表現できるときには，計算結果は真の値であることが要求される。現時点でこれはFull BASIC規格の象徴ともいえる規定であり，筆者が配布する処理系の名称「(仮称)十進BASIC」の由来でもある。

Full BASICでは，組込み関数も規格で定められている。そして，組込み関数やべき乗演算の計算結果の正確さは，加減乗除の演算と同程度に要求されている。

コンピュータを計算の道具として使うとき，誤差の範囲を確定することは重要なことであり，初心者に対する教育において，そのことは強調されなければならない。しかし，誤差の見積もりは概して複雑な仕事であり，多くの場合，初学者には大きな負担となる。十進数に基づき，しかも，可能なかぎり正しい値を返すことを保証するFull BASICの規定は，誤差解析の敷居を下げ，初学者に誤差に注目した活動を行わせるのに有用である。

方言BASICの多くが数値に複数の型を用意するが，Full BASICには数値の型の概念がない。そのため，Full BASICでは数値定数の記述の仕方によってプログラムの意味が変わってしまうなどの問題は生じない。

2.2. 初学者向きに定められた構文規則

Full BASICが構造化プログラミングに対応した言語であることはよく知られているが，初心者が確実なプログラムを書くことができるように工夫された文法を採用した言語であることはあまり知られていないように思われる。それらは処理系を作成する側からすれば文法を複雑なものにする仕様であり，処理系を作る側にとってありがたくないものである。したがって，多くの場合，この特質は，方言BASICとFull BASICとを分けるものとなっている。

BASICでは，比較演算の結果は数値ではない。一方，マイクロソフトBASICでは，比較演算の結果は数値であるから，

```
IF 1<A<3 THEN ···
```

のような文を書くことができるが，初心者はこの文の正しい意味を理解せずに書くであろうし，誤った理解で書いたとき，自らの誤解に気付くことは困難である。

Full BASICは，基本BASICから文法を拡張し，IF文のTHENに続けて単純実行文を書くことを認めている。Full BASICの文法では，IF文は単純実行文ではない。これは，次のように意味の曖昧さが存在するIF文が生成されないようにするための工夫である。

```
IF ～ THEN IF ～ THEN ···ELSE ···
```

Full BASICにはマイクロソフトBASICのマルチステートメントに対応する構文は存在しないが，それも，次の文が正しい意味を理解せずに書かれる可能性が高いことを考えれば当然であろう。

```
IF A=1 THEN GOTO 10:IF A=2 THEN GOTO 20
```

-A-Bのような数値式を書いたとき，先頭の負号は単項演算であり，2個目の負号は2項演算である。Full BASIC（基本BASICも）では，それらは区別されることなく，加算と同順位の演算として処理されるが，処理系を作る側に立つと，これは文法を複雑なものとする仕様である。マイクロソフトBASICをはじめとして（BASIC以外の言語を含め）多くの言語が，単項マイナスを2項演算の減算と別個の演算として処理することによってその複雑さを回避している。それによって，2^-3や2*-3のように負数を括弧で囲まずに式を書くことができるが，BASICではそれは望ましいことではない。というのは，BASICではべき乗の演算は左から順に行われることにな

っているのにも関わらず，2^-3^2は2^(-3^2)の意味になってしまうからである。

2.3. 論理演算

マイクロソフトBASICでは，AND，OR，NOTなどの論理演算は，2進数値に対するビット演算である。比較演算の結果が数値であることとあわせて，（処理系を作成する側にとっての）シンプルな文法を実現している。

それに対して，Full BASICのAND，OR，NOTは制御構造の一部である。Full BASICでは，p AND q でpが偽であるときqは評価されない。同様に，p OR q でpが真であるときqは評価されない。それによって，たとえば，配列aの添え字の上限が10である場合に

```
IF i<=10 AND a[i]>0 THEN
```

のような簡潔な表現が可能になる。マイクロソフトBASICで同様の条件判断を行うためにはIF文をネストして書く必要がある。

2.4. 構造化例外状態処理

Full BASICの特徴は，例外状態処理まで含めて構造化されていることである。数値計算を行うプログラムでは，桁あふれなどの例外に対処することが本質的に不可欠であり，例外状態処理を見通しよく記述できる構造化例外状態処理の機能は，応用上，欠かせないものである。

2.5. 組込み関数

Full BASICでは，基本BASICに対し，組込み関数が強化されている。

基本BASICでは逆正接のみであった逆三角関数に，逆余弦，逆正弦が追加されている。また，角の大きさの単位を度に変える手段も用意されている。したがって，高校数学Iで余弦定理を学習すれば，3辺の長さから頂角の大きさを求めるプログラムを書くことができる。また，対数関数には常用対数と2を底とする対数が追加されているから，自然対数を学習する前の生徒でも利用できる。

2.6. 配列入出力と行列演算

BASICには，MAT文による行列演算が可能という特徴がある。Full BASICでは，配列入出力，行列の和，差，積，逆行列，転置行列，スカラー倍などのMAT文が用意されていて，配列，行列関係の処理を簡略に記述できる。

2.7. 問題座標系に基づくグラフィックス

グラフィックス機能は，計算結果を見やすく表示するために欠かせない。Full BASICのグラフィックス機能は問題座標系を基礎におくものであり，計算された数値を直接指定してグラフ化できる。

直線を描く命令は，XYプロッタの動作を模したものとなっていて，直線を描く命令に最初に点を指定したときには，線を描かずに描点が移動する。そのため，マイクロソフト系BASICの描画コマンドよりも簡単なロジックでグラフや曲線を描くことができる。

2.8. 予約語

Full BASICには，部分集合中核と呼ばれる規格と，（「部分集合」を前置しない）中核と呼ばれる規格とが並存する。「部分集合」を前置する規格は，現在の多くのBASICを名乗る処理系がそうであるように，機能語を予約語としている。もう一方の，「部分集合」を前置しない規格は，予約語を規格で定めた18個に限定する。「部分集合」でないほうの規格を採用すれば，利用する処理系によって予約語が異なり，一方で動作するプログラムが他方では動作しないというような問題が回避できる。また，「部分集合」規格も含め，利用者定義関数の識別名をFNで始める必要がなくなっているから，利用者定義関数にf(x)，g(x)などの自然な名前を付けることができる。

2.9. 変数名の宣言

Full BASICでは，単純変数を含め，すべての変数名を宣言することができる。この機能を利用すると，コンパイラは宣言されていない変数名を検出して警告を出すことが可能であり，大きなプログラムを安全に作るのに役立つ。

3. 処理系開発の現状

3.1. 概要

現在，開発中の Full BASIC 言語処理系は，Microsoft Windows 環境で動作する。Linux 環境にも移植を進めているが，一部，機能制限がある。本稿では，最新の Win32 版について述べる。

JIS Full BASIC は，幾つかの機能単位から構成されている。現バージョンが目指すのは図形機能単位である。図形機能単位は中核機能単位を含むから，必然的に，中核機能単位にも対応する。中核機能単位は，構造化された制御構造や局所変数の概念が存在し，また，再帰呼び出しにも対応した副プログラム，関数定義，それに構造化された例外状態処理機能などのほかに，順編成表示形式，順編成内部形式ファイル，流れ編成内部形式の 3 種のファイルに対応している。

一方，実時間機能や拡充ファイルなどには対応していない。実時間機能は，科学技術分野には必須であるので，今後の課題といえる。

Full BASIC の図形機能には，図形変形という特徴的な機能が存在する。副プログラムとほぼ同様の構文で絵定義と呼ばれる描画手続きを書くことができる。絵定義は CALL 文の代わりに DRAW 文で呼び出す。DRAW 文には，SHIFT, SCALTE, ROTATE, SHEAR などの図形変形を指定することができ，それらを，複数個，連結して指定することも可能である。組込みの図形変形では平面上のアフィン変換しか実現できないけれども，それらの代わりに 4×4 行列を指定すれば射影変換を行わせることも可能である。本 BASIC の最新版は，ほぼ完全な形で射影変換に対応している。

4. 開発された処理系の詳細と課題

ここでは，JIS Full BASIC の章構成に従って，図形機能単位との非互換を述べる。なお，行番号の省略を許すこと，また，扱う文字集合の範囲を漢字にまで拡張したことによって生じる非互換については省く。

ここでの問題は，JIS の誤りかも知れないと考えられる規定にまで対応させるべきか，あるいは，使い勝手が悪化するような場合でも規格を尊重すべきか，という点である。

4.1. 規格合致プログラムを受け入れる

本 BASIC には，規格準拠であっても翻訳できないプログラムが存在する。

その一つは，

```
10 PRINT 1e1000000
```

のように極端に大きな数値定数を書くと，翻訳時にエラーになってしまうことである。本来は実行時にエラーとなるべきものである。

また，配列添え字を 32 ビット整数で管理する関係で，

```
10 DIM A(30000000000 TO 30000000010)
```

のような配列宣言を含むプログラムは規格合致ではあるが本 BASIC では翻訳できない。

4.2. 意図的な非互換

4.2.1. 続行可能例外を生成する文

Full BASIC のグラフィックスは，計算結果を図示するという目的に適した機能を提供している。とはいうものの，その設計は完璧とはいえない。初心者には使いづらい点も幾つかある。

その一つが，グラフィックスコマンドに多い続行可能例外を生成する文である。これらの文は，例外状態となったとき，その文が when 本体になければ，エラーを報告せずに適当に処理して先に進む。たとえば，SET WINDOW 文では，幅，または，高さが 0 の窓を指定すると，その文を実行しなかったことにして先に進む。この動作は，初心者には，誤りに気付く機会を奪う不利益しかもたらさない。

本 BASIC では，JIS 本来の動作を行わせるオプション設定を設けた上で，通常の動作ではこれらの文で例外が起きた場合には続行

不能例外を生成するようにしている。

4.2.2. 互換性オプション

同様に，互換性を設定するオプションを設け，通常はJIS非互換で動作するが，オプション設定をJISの側に変更すればJISの規定どおりに動作する文がいくつか存在する。

そのひとつは，書式指定文字列の解釈で，本BASICの通常の動作ではTrue BASICのように#のみで書かれた書式で負数の出力が可能なようにしている。

もうひとつは，DEF 文の引数の扱いで，Full BASIC では関数定義の引数は値渡しであるが， DEF 文の通常の動作では可能な場合は参照渡しとして実行時間の短縮を図っている。

4.3. 動作上の非互換

以下，JIS 合致プログラムの動作上の非互換について述べる。

4.3.1. 計算結果の正確さ

なお，計算結果の正確さについては，現時点で，Intel 製のプロセッサで実行する範囲で規格に合わない事例は見つかっていないが，各社製CPUのFPU命令の正確さに関する情報が入手できていないので，保証もできない。計算結果はCPUのFPU命令の正確さに依存する。実際，FPU 命令をエミュレートするMacのVirtual PC上では計算誤差が大きい。[9]

4.3.2. 配列の上下限

本 BASIC では，配列の上下限は，-2147483647～214748364 の範囲になければならないものとしている。これは，配列下限の管理を 32 ビット整数で行っているのが原因であるので，それを 64 ビット化するかBASIC の数値に置き換えればこの問題は解決できるが，その一方で実行速度の低下も招きかねない。実用上，極端に上下限の範囲が片寄った配列を宣言することはないと思われるので，現状では非互換のままとしている。

4.3.3. プログラム連鎖（CHAIN 文）

本 BASIC では，CHAIN 文の引数の型とそれを受け取る PROGRAM 文とで型の突合せを行う手段を用意していない。そのため，不当な引数の対応関係にある CHAIN 文と PROGRAM 文が例外状態にならずに実行できてしまうことがある。

CHAIN 文自体が現在の構造化プログラミングの基本的な考え方にそぐわないものであるので，CHAIN 文で BASIC のプログラムに連鎖するような使い方は推奨できるものではない。CHAIN 文，PROGRAM 文の存在意義は他言語で書かれたプログラムとの連携にあると考えているので，そちらを優先し，少なくとも，実用上困ることのない，正しくないプログラムの実行を行った場合の例外生成についてのみ関係する JIS 非互換の存在は，気にしないことにしている。

4.3.4. MAT PRINT 文

MAT PRINT 文では，一つの印字配列名並び中の後続する配列との間に行末が生成されることになっている。しかし，MAT PRINT 文の最後に行末を生成することにはなっていない。

そのため，規格通りの動作では，

```
10 MAT PRINT A,B
```

では，配列 A と配列 B の間に 1 行分の余白が入るのに，

```
10 MAT PRINT A
20 MAT PRINT B
```

では，配列 A と配列 B が続けて出力される。

この動作は不合理であるので，本 BASIC では規格を無視し，MAT PRINT 文は最後に余分な行末を生成することにしている。

4.3.5. 例外処理区における例外状態

本 BASIC では，例外状態の扱いは，JIS Full BASIC の 12.1.4(1)の規定の通りになっている。これは，当該の文が例外処理区にあったとしても例外ではない。すなわち，例外が発生して例外処理区に移り，その処理の過

程で続行可能例外を起こす文で例外が発生した場合には，その例外処理区を（従ってその保護区を）含むwhen本体が存在しなければ，省略時想定の回復手続きを実行する。たとえば，次のプログラムの50行で数値以外のものを入力すると，INPUT文に関する省略時想定の回復手続きを適用して再入力を求める。

```
10 LET a=0
20 WHEN EXCEPTION IN
30     PRINT 1/a
40 USE
50     INPUT a
60     RETRY
70 END WHEN
80 END
```

しかし，JIS Full BASICは，12.1.4(8)において，「例外処理区の内部において重ねて起きた例外状態は，続行不能なものとして扱われる。このことは，（中略），その他の続行可能または続行不能なあらゆる例外を含む」とし，例外処理区において発生した続行可能例外は続行不能扱いすべきことを求めている。

この文には，参考として，「ANSI X3.113の文章はやや異なるが，TIBによって訂正した。」との補足があり，ANS規格では，「例外処理区の構文的に内部にある文で再び例外状態が起こると，この新しい例外状態は省略時想定の例外状態手続きで処理される」（訳文は，共立出版刊 bit 別冊アメリカ規格 Full BASIC－全訳と解説－による）となっていて，JISの規定とは正反対である。

つまり，本BASICの動作は，その続行可能例外を起こす文を含む例外処理区を含む保護区を囲むwhen本体が存在しない場合にはJIS非互換でANS合致，反対に，その続行可能例外を起こす文を含む例外処理区を含む保護区を囲むwhen本体が存在する場合にはJIS合致でANS非互換である。（JIS Full BASICには，TIBはANSに対する変更ではないとANSIが述べていると書いてある。）

4.3.6. TRACE文

JISではTRACE文は代入文の実行によって代入される値を報告することになっているが，本BASICでは，値の変化した変数の値のみを報告することにしている。

4.4. 図形（JIS Full BASIC 13章）

4.4.1. PLOT TEXT

現時点で，本BASICと規格との最大の相違は文字出力文GRAPH TEXTとPLOT TEXTの動作である。

Full BASICの図形機能の特徴は，図形が問題座標で定められることである。Full BASICでは，文字も問題座標で定められるものとしている。Full BASICには，図形を変形して描く機能があり，それを正しく動作させるためには，文字も問題座標で定義されなければならないのであるが，しかし，これは，計算結果の図示という最も多く利用されるであろう応用に対しては大きな不都合である。つまり，問題座標系では，たとえば，横軸は時刻，縦軸は長さを表すというような使われ方をする。横軸の単位は秒であることもあるし，分であることもある。また，縦軸はmm単位であるか，m単位であるかも知れない。そのような情報を図示するのに，文字の大きさを問題座標系で定め，なおかつ，字形の縦横比を問題座標上で正しく表示したのでは，正常に読める状態で文字を描くのは容易ではない。

本BASICの従来のバージョンでは，文字の字形は物理座標で定義されるものとし，PLOT TEXT文では，AT句で指定された座標のみを変換するものとしてきた。しかし，最新版では，文字の字形は物理座標で定義されるという性質を変えずに，PLOT TEXT文では，文字の大きさと向きを図形変形の対象とすることにした。

将来のバージョンでこの部分を完全にJISに対応させるべきか否かが大きな課題である。JISのPLOT TEXT文の動作は，文字を図形

の一部として変換して描くためには必須の動作であるから，無視はしたくない。しかし，そのために，多くの応用上の利便性を失うのは耐え難い。

4.5. 拡張構文の存在

Full BASIC 規格は，規格合致でないプログラムを受け入れてよいことになっている。しかし，そのためには，拡張構文が明確に文書化されていることが必要とされる。

本 BASIC では，Full BASIC の構文に忠実に翻訳し，処理系の簡略化にとって都合のよい拡張を行わないことを原則としているが，常識的な解釈で不都合がでない範囲では手抜きのための拡張も存在する。しかし，現時点でそれらの文書化は完成していない。完全な文書化を進めることは今後の課題である。

4.6. 規格自体の不具合

4.6.1. USING$関数

USING$関数では，「書式付き出力の例外状態についての規定も適用する」となっているが，本 BASIC では，例外状態となった場合，「その次の行に書式なし出力の表現で値を報告し，･･･」の部分については無視している。これは，かなり無理のある要求であり，規格のミスのように思われる。

4.6.2. 多義性のある構文規則

次のプログラムの 40 行は数値 mat 文とも変形指示 mat 文とも解釈できる。

```
10 DEF ROTATE(x)=2*x
20 DIM a(4,4)
30 MAT a=IDN
40 MAT a=ROTATE(PI/6)*a
```

実際上，このような文が書かれる可能性はほとんどないと思われるが，一応，規格そのものが抱える問題として指摘しておく。

なお，本 BASIC の現バージョンでは，数値 mat 文としての解釈が優先する。

4.6.3. def 文において発生した例外

JIS Full BASIC 12.1.6 注意(9)に，「例えば，def 文，on-goto 文，on-gosub 文，if 文などの中で例外状態が起こったときに CONTINUE 文を実行すると，制御はその行の構文的に次の行に移る。」となっているが，def 文で発生した例外状態は無条件に呼び出し元（つまり定義関数呼び出しを書いた式を含む行）に伝達されるから，def 文で例外が発生したとき，CONTINUE 文を実行したとき実行される文は def 文の次の行ではなくて，その関数を呼び出した行の次の行である。この記述は「注意」であり，動作を規定するものではないから，単なる誤りであろう。

4.6.4. 例外処理区における EXIT 文の実行

do 区や for 区が when-in 区を含むとき，その when-in 区の例外処理区に exit-do 文や exit-for 文を書くことは規格上禁止されていない。しかし，規格上，それらの文の効果は明確ではない。すなわち，それらの文を実行すれば当然例外処理区を出ることになるはずであるが，12.1.4(4)の記述では，例外処理区から出る場合に含まれていない。本 BASIC は，それらの exit-do 文，exit-for 文を実行した場合には，例外処理区から出るものとして扱う。

4.6.5. 誤記・誤植

Full BASIC 規格 9.2.5(2,5)のプログラムに

```
100 DECALARE INTERNAL SUB S
```

という行が含まれるが，これは Full BASIC の構文規則からは導かれない。

また，13.4.2(7)の range 句の後に ? がないが，range 区を省略した場合の動作の記述が存在し，また，ANS には ? が存在することから，これは誤植と思われる。

4.7. 付属書 I（モジュール・単文字入力）

本 BASIC は，付属書 I に規定されるモジュールおよび単文字入力にも対応している。

なお，Full BASIC のモジュール機能に関する規定は，他の機能に比較し，未熟さが目立つ。完全に記述どおりの実装は不可能である。

4.7.1. 単文字入力文における行末

本BASICでは，CHARACTER INPUT文をファイルをバイト単位でよむために利用することを想定している。そのため，すべての文字を非プログラム文字として扱い，行末を未定義としている。

4.7.2. モジュールの構文規則

Full BASICの構文規則は，JIS本体および付属書Iによれば

修飾識別名コモジュール名 小数点 数値単純変数名 (18.2)

数値単純変数名＝数値識別名 (5.2)

数値識別名コモジュール名 小数点 英字識別名文字* (18.2)

(*は反復を表す)

となるので，

モジュール名 小数点 モジュール名 小数点 英字 識別名文字*

という修飾識別名が存在することになる。これは合理的でないので，本BASICではJIS付属書の構文規則に制限を加えて，意味不明な識別名が生成されないようにしている。

4.7.3. OPTION文を書く順序

附属書Iの構文規則では，PUBLIC文，SHARE文はモジュール本体区よりも先に書かなければならないが，本BASICではコンパイラの都合で，次のような変更（非互換）が存在する。

OPTION ARITHMETIC，あるいはMODULE OPTION ARITHMETICを書く場合は，PUBLIC NUMERIC，およびSHARE NUMERICより先に書かなければならない。

OPTION BASE，あるいはMODULE OPTION BASEを書く場合は，PUBLIC文，SHARE文による配列宣言より先に書かなければならない。

4.7.4. 手続きのSHARE宣言

現バージョンでは，モジュール内の手続きはすべて広域的なものとみなして処理している。そのため，SHARE宣言を書いても，その手続きが外部から参照されていたときにエラーにできない。変数に比べて手続きは数が少なく，よほど巨大なプログラムを作らないかぎり手続きのSHARE宣言の機能が欲しくなることはないので，将来の課題として，後回しになっている。

5. 終わりに

最も重要な課題は教育関係者の意識改革である。情報教育が，既存の情報技術の使い方を教えることよりも，アルゴリズムの作り方を教え，アルゴリズムの記述により問題解決が可能となることを教えることを目標とした教育に変わることを期待したい。

【参考文献】

[1] 日本工業規格 電子計算機プログラム言語 Full BASIC JIS X 3003-1998

[2] American National Standard for Information Systems － Programming Languages －Full BASIC, ANSI X3.113-1987

[3] International Standard Information technology －Programming Languages － Full BASIC, ISO/IEC 10279(1991)

[4] 白石和夫, Windows環境におけるJIS Full BASICの実装, 情報処理学会論文誌：プログラミング Vol.41 No.SIG9(PRO 8), pp.52-61 (2000)

[5] 白石和夫, (仮称) 十進BASICのLinuxへの移植, 文教大学教育学部紀要第38号pp.109-115 (2004)

[6] 白石和夫, JIS準拠BASIC言語処理系の開発とその目的, 日本数学教育学会会誌 52-3, pp.8-15, 日本数学教育学会(1998)

[7] Kazuo Shiraishi, Development of a BASIC Language Processing System in accordance with ISO and its Aim, Proceedings of the Third Asian Technology Conference in Mathematics, pp.225-232, Springer(1998)

[8] 白石和夫, 学校数学におけるアルゴリズム教育の位置と役割, 日本数学教育学会 Yearbook 第5号(2004) pp.169-181

[9] Ken Kiukchi, FPU Emulation Bug of VirtualPC, May 31, 1998 http://www.geocities.jp/ken_kikuchi3/virtualpc/VirtualPC_Bug.html