８０３８７用高速数値演算ライブラリ

Mathma387

ユーザーズ・マニュアル

Ver. 1.0

株式会社　アークブレイン

御注意

●このソフトウェア（Mathma387）およびこのマニュアルは株式会社アークブレインが著作権を保有します。
●このソフトウェアは１台のコンピュータにつき１セットが原則になっています。
●このマニュアルの一部又は全部を無断で複写、使用することはできません。
●このソフトウェアの中のライブラリをアプリケーションに組み込んで、そのアプリケーションを販売した場合、株式会社アークブレインに対してローヤリティを支払う必要はありません。
●このマニュアルおよびソフトウェアは将来予告なしに変更することがあります。
●このソフトウェアを運用した結果の影響に付いては責任を負いかねますので御了承下さい。
●このマニュアルの内容については万全を期して作成いたしましたが、万一御不審な点や、誤り、記載漏れなど、お気付きの点がありましたら御連絡下さい。
●付属のユーザー登録カードは、できるだけ早く株式会社アークブレインまで御返送下さい。

目　次

１．１　Mathma387 の概要

Mathma387 は、Intel Corporation の 80387、80486 やサイリックス社の FasMath（CX－83D87、CX－83S87）用に対応した C 言語用数値演算ライブラリです。

MS-DOS のほとんどの C コンパイラは、32 ビット CPU である 80386 やそのコプロセッサである 80387 が実行可能な 32 ビットのオブジェック・コードを出力することができません。MS-DOS 上で動作する 32 ビットに対応した C コンパイラも若干出現し始めていますが、DOS エクステンダを使用したプロテクトモードでないと動作不可能であり、何かと不便です。

Mathma387 は 80387 で使用可能な特殊な命令をサポートすることにより、sin()や cos()などの関数を飛躍的に向上します。

Mathma387 にはパソコンにインテル社の 80387 相当の FPU が搭載されているかどうかを調べる関数が用意されており、アプリケーション実行時に FPU の種類を判別してマシンにあった最良のスピードで、画像処理や構造解析、レイトレーシングによる CG の様に非常に計算時間がかかるアプリケーションの実行スピードを飛躍的に向上させることが可能となります。

Mathma387 でサポートされている関数の種類は、コンパイラのライブラリに含まれるほぼ同じです。しかし、acosh()、asinh()、atanh()等コンパイラのライブラリにはないものもあります。なんといっても大きな違いは、引き数をバリュー渡しで行っている C 言語の関数と互換性のある関数と、互換性は犠牲にしても高速化が可能なポインタ渡しの関数が用意されていることです。特に float 実数をポインタ渡しで行う関数は高速です。

kindcpu()と kindfpu()は弊社のグラフィックス・ライブラリ「Ultra C Graph」（UCG）に含まれている kind_cpu()と kind_fpu()と同じ機能を持つものです。同じ名前だと、Mathma387 と UCG のライブラリを同時に使用したときにエラーとなってしまいます。そのために Mathma387 では同じ機能でも名前を変えています。Mathma387 では関数の名前がぶつからないように、'_'を省略してあります。

i387 で始まる関数は、インテルの数値演算コプロセッサ 80387 を搭載した 80386 ないしは 80486（80486 には 80387 相当の数値演算コプロセッサは内蔵されています）でないと使用できません。

80287XL の様に 80387 の命令をサポートしている数値演算コプロセッサを搭載した 80286 マシンでも、使用可能です

i87 で始まる関数は、8086 用の数値演算コプロセッサである 8087 や、80286 用の数値演算コプロセッサである 80287 でも使用可能な関数です。数値演算プロセッサが内部的に使用する定数機能をプログラムでも使用できるようにした関数です。

１．２　Mathma387 の動作可能な機種

Mathma387 を使用したプログラムが動作するためには、インテル社 80387 以上の数値演算コプロセッサを搭載した MS-DOS パソコンが必要です。

外部バスが 32bit の 80386DX や、外部バスが 16bit の 80386SX のどちらでも 80387 相当の数値演算コプロセッサが搭載されていれば動作可能です。

ノート型パソコンの様に 80386SX を搭載したパソコンでも、数値演算コプロセッサが取り付けできないタイプのものでは使用できません。

インテル社の 80486 は 80387 相当の数値演算コプロセッサが内蔵されているため、Mathma387 を使用したプログラムを実行することができます。

インテル社の 80287XL は、80387 の拡張命令をサポートしているため動作可能です。

インテル社 8087、80287 等では、'i387_'で始まる 80387 専用の関数は動作しませんが、他の関数は動作可能です。

インテル社以外の数値演算プロセッサでも 80387 と互換性のあるコプロセッサであれば、動作可能です。特に、Cyrix 社の FasMath は FSIN、FCOS などの命令は少ないクロック数で実行可能なため、Mathma387 の威力が発揮できます。

DEMO3D.EXE の様なグラフィックス・ソフトは NEC の PC9801 でのみ動作します。

１．３　ファイルの内容

ディレクトリ ¥

AUTOEXEC.BAT	Mathma387 用の場合の一例

ディレクトリ INCLUDE　インクルード・ファイル

MATHM387.H	TYNY モデル以外（MSC Ver.6 対応）
MATH387S.H	TYNY、SMALL モデル用（MSC Ver.6 対応）
MATHM387.INC	MS　FORTRAN 用
MATHM387.BI	QuickBASIC 用
CGRAPH.H	Ultra C Graphg 用
MATHM387.MS6	MSC Ver.6 用　（MATHM387.H と同じ）
MATHM387.MS5	MSC Ver.5.1 用
MATH387S.MS6	MSC Ver.6 用　（MATH387S.H と同じ）
MATH387S.MS5	MSC Ver.5.1 用
CGRAPH.MS6	MSC Ver.6 用　（CGRAPH.H と同じ）
CGRAPH.MS5	MSC Ver.5.1 用

ディレクトリ LIB　ライブラリ

MATHM387.LIB	LARGE モデル（TYNY を除く全モデルで使用可）
MATH387S.LIB	SMALL モデル（TYNY、SMALL モデル用）
UCGMSCSD.LIB	UCG の SMALL モデル（g_init()を外したデモ版）

ディレクトリ BIN　バッチ、実行ファイル

CLSML.BAT	MSC SMALL モデル用
CLMML.BAT	MSC MIDUUM モデル用
CLCML.BAT	MSC COMPACT モデル用
CLLML.BAT	MSC LARGE モデル用
CLHML.BAT	MSC HUGE モデル用
CLTMSL.BAT	MSC TYNE モデル用（MATH387S.LIB をリンク）
CLSMSL.BAT	MSC SMALL モデル用（MATH387S.LIB をリンク）
QCLSML.BAT	QC　SMALL モデル用
QCLMML.BAT	QC　MIDIUM モデル用
QCLCML.BAT	QC　COMPACT モデル用
QCLLML.BAT	QC　LARGE モデル用
QCLHML.BAT	QC　HUGE モデル用
QCLSMSL.BAT	QC　TYNY モデル用（MATH387S.LIB をリンク）
QCLTMSL.BAT	QC　SMALL モデル用（MATH387S.LIB をリンク）
GINIT.EXE	UCG の初期化コマンド
GCLS.EXE	グラフィックス画面消去コマンド

ディレクトリ DEMO

MK.BAT	デモ作成バッチファイル
MAKEFILE	メイクファイル
TSFPUS.C	FPU テストプログラム（TYNY、SMALL 用）
TSFPUS.EXE	同　実行プログラム
TSFPUL.C	FPU テストプログラム
TSFPUL.EXE	同　実行プログラム
TSACCURA.C	精度チェックプログラム
TSACCURA.EXE	同　実行プログラム
TSATAN.C	atan()用テストプログラム
TSATAN.EXE	同　実行プログラム
TSATAN2.C	atan2()用テストプログラム
TSATAN2.EXE	同　実行プログラム
TSCONST.C	定数表示プログラム
TSCONST.EXE	同　実行プログラム
DYNALOOP.C	ダイナミックループ表示プログラム
DYNALOOP.EXE	同　実行プログラム
TSEXP.C	exp()テストプログラム
TSEXP.EXE	同　実行プログラム
TSACOSH.C	acosh()テストプログラム
TSACOSH.EXE	同　実行プログラム
FORDEMO.FOR	MS FORTRAN デモプログラム
FORDEMO.EXE	同　実行プログラム
MAKEFOR	同　メイクファイル
BASDEMO.BAS	MS QuickBASIC デモプログラム
BASDEMO.EXE	同　実行プログラム
MAKEBAS	同　メイクファイル

ディレクトリ DEMO3D

DEMO3D.C	三次元ワイヤフレーム表示プログラム
MK.BAT	デモ作成バッチファイル
MAKEFILE	メイクファイル
CUBE.BIN	CUBE 三次元データ
CAMERA.BIN	CAMERA 三次元データ
CUBE.POS	CUBE 視点データ
CAMERA.POS	CAMERA 視点データ
DEMO3D.EXE	三次元ワイヤフレーム表示プログラム
DEMO3D.MAP	マップファイル

１．４　Mathma387 の最も簡単な使用方法

Mathma387 は基本的に、C 言語用のライブラリとして作られています。C 言語用の例題は豊富に含まれています。

Microsoft FORTRAN や Microsoft QuickBASIC とのインタフェースをとるためのインクルード・ファイルが用意されており、これらの言語からも使用することができます。

先ずは、Mathma387 のインクルードファイルとライブラリをコンパイラが使用できるディレクトリにコピーして下さい。

```
copy c:¥include¥*.h a:¥include¥*.*
copy c:¥lib¥*.lib a:¥lib¥*.*
```

（フロッピーが C ドライブ、インストールするディスクのディレクトリを A と仮定）

コピーしたディレクトリを環境変数で指定して下さい。

```
set INCLUDE=a:¥c6¥include;a:¥include
set LIB=a:¥c6¥lib;a:¥lib
```

指定されていないと、ヘッダーファイルをインクルードできなかったり、ライブラリをリンクで着なかったりすることがあります。

プログラムを Mathma387 用に直す方法は非常に簡単です。プログラム冒頭の

```
#include <math.h>
```

の後に次の様に 2 行追加してコンパイルとリンクをしなおせば、sin()、cos()、tan()等の関数を 80387 対応の Mathma387 のライブラリと入れ替えることができます。

```
#include <math.h>
#define MATHMA387MACRO 1
#include <mathm387.h>
```

これにより、cos()の様な関数は Mathma387 の i387_dcos()という関数に置換されます。この両関数は引き数や返り値の返し方が同一であるため、プログラムの直しは非常に簡単です。これだけの作業で、数値演算ライブラリ関数は約 2 倍高速化されます。

なお、コンパイルは

```
cl /FPi87 /W3 test.c /link mathm387.lib
```

の様に、'/W3'または'/W4'（MSC Ver.6.0 より使用可能）というオプションを指定して文法チェックを厳しくすることをお勧めします。これにより、単純なミスによるバグをコンパイラが発見することができます。

コンパイル・オプション'/FPi87'は、数値演算コプロセッサを搭載したコンピュータ用のコードを生成します。このオプションを指定すると、プログラムが高速かつコンパクトになります。しかし、数値演算コプロセッサが無い場合には、プログラムの冒頭で自動的にエラーメッセージを表示して MS-DOS に戻ってしまうため、kindfpu()によって数値演算コプロセッサが搭載されているかどうかのチェックはできなくなります。

サブディレクトリ<BIN>には、各モデルのコンパイル用のバッチフィルがあります。

CLSML.BAT	MSC SMALL モデル用	(MATHM387.LIB をリンク)
CLMML.BAT	MSC MIDIUM モデル用	(MATHM387.LIB をリンク)
CLCML.BAT	MSC COMPACT モデル用	(MATHM387.LIB をリンク)
CLLML.BAT	MSC LARGE モデル用	(MATHM387.LIB をリンク)
CLHML.BAT	MSC HUGE モデル用	(MATHM387.LIB をリンク)
CLTMSL.BAT	MSC TYNY モデル用	(MATH387S.LIB をリンク)
CLSMSL.BAT	MSC SMALL モデル用	(MATH387S.LIB をリンク)
QCLSML.BAT	QC SMALL モデル用	(MATHM387.LIB をリンク)
QCLMML.BAT	QC MIDIUM モデル用	(MATHM387.LIB をリンク)
QCLCML.BAT	QC COMPACT モデル用	(MATHM387.LIB をリンク)
QCLLML.BAT	QC LARGE モデル用	(MATHM387.LIB をリンク)
QCLHML.BAT	QC HUGE モデル用	(MATHM387.LIB をリンク)
QCLSMSL.BAT	QC SMALL モデル用	(MATH387S.LIB をリンク)
FLML.BAT	MS FORTRAN 用	
QBML.BAT	QuickBASIC 用	

TYNY モデルは、MSC Ver.6.0 以降でないと使用できません。

１．５　MATHM387.LIB のリンクの方法

Mathma387 のライブラリをリンクするには、リンク時に MATHM387.LIB というライブラリを指定して下さい。

```
cl /FPi87 test.c /link mathm387.lib
```

または、

```
link test.obj,test.exe,test.map,mathm387.lib;
```

メモリモデルをラージにする場合は、

```
cl /AL /FPi87 test.c /link mathm387.lib
```

の様に指定して下さい。

Mathma387 はラージモデルのライブラリですが、スモール、ミディアム、コンパクト、ラージ、ヒュージのどのモデルのオブジェクトともリンク可能です。

マイクロソフトのＣコンパイラでは、環境変数を次のように設定することにより、自動的に Mathma387 をリンクすることが可能です。

```
SET LIB=c:¥lib                    (Mathma387.lib が置かれているディレクトリ)
SET CL=/W4 /FPi87 /link mathm387.lib
```

この様に、環境変数が設定されていれば、

```
cl test.c
```

とキー入力するだけで、自動的に mathm387.lib がリンクされます。

１．６　スモール、タイニー・モデルによる高速なプログラム

少しでも計算時間を速くする必要があり、なおかつスモール・モデル（プログラムとデータが各 64Kバイトに納まるモデル）やタイニー・モデル（プログラムとデータを合わせて 64Kバイトに納まるモデル）で構わない場合のために、タイニー、スモール・モデル専用のヘッダー・ファイルとライブラリが用意されています。

ヘッダー・ファイルは

```
#include <math387s.h>
```

ライブラリは

```
MATH387S.LIB
```

です。

当然のことですが、これらのヘッダー・ファイルやライブラリを使用した場合にはタイニー・モデルかスモール・モデルでしかコンパイルできません。

スモール・モデル専用にするとラージ・モデルの場合に比べて約 10 数%高速になります。

タイニー・モデルでは、実行ファイルを COM 形式にすることができます。EXE 形式の場合に比べて、COM 形式の方がプログラムサイズは一般的に小さくなります。

タイニー・モデルにする場合には、<math387s.h>と MATH387S.LIB を必ず使用しなければいけません。スモール・モデルの場合は、<mathm387.h>と MATHM387.LIB の組み合わせでも使えますが、ミディアム、コンパクト、ラージ、ヒュージ・モデルでは<mathm387.h>と MATHM387.LIB を必ず使わなければいけません。

１．７　32 ビット CPU 用高速アクセスデータ

16 ビット CPU の場合でも、16 ビット（ワード）のデータをワード境界（アドレスが 2 の倍数）に置かないと、データのアクセスが通常の 2 倍のクロック数を必要とします。

これと同様に、32 ビットＣＰＵの場合は、32 ビット（ダブルワード）のデータをダブルワード（アドレスが 4 の倍数）に置かないと、通常の 2 倍のクロック数が必要となってしまいます。

リアルモード用のコンパイラでは、ＣＰＵがダブルワードのデータをアクセスするときはワードごとに分けてアクセスするため問題ありませんが、ＦＰＵはダブルワードはおろか、8 バイトや 10 バイトのデータにアクセスすることが頻繁に起こります。このときに、データのアドレスがダブルワード境界にないと、FPU 用の FLD や FST などの命令が 2 倍の時間を必要としてしまいます。

そのため、Mathma387 ではダブルワード境界に置かれる 4 バイト、8 バイトそして 10 バイトデータを 12 個ずつ用意しています。これらのデータは以下の通りです。

```
float        fdata32a,  fdata32b,  fdata32c,  fdata32d;
float        fdata32e,  fdata32f,  fdata32g,  fdata32h;
float        fdata32i,  fdata32j,  fdata32k,  fdata32l;
double       ddata32a,  ddata32b,  ddata32c,  ddata32d;
double       ddata32e,  ddata32f,  ddata32g,  ddata32h;
double       ddata32i,  ddata32j,  ddata32k,  ddata32l;
long double lddata32a, lddata32b, lddata32c, lddata32d;
long double lddata32e, lddata32f, lddata32g, lddata32h;
long double lddata32i, lddata32j, lddata32k, lddata32l;
```

これらの変数は、<mathm387.h>の中で外部変数として定義されています。変数名を変更することはできませんが、次の様にマクロを使用して、他の名前で使用することができます。

例：

```
#define ldx  lddata32a
#define  dx   ddata32a
```

注意：

MSC Ver.6.0 は、プログラム中で long double の定数を記述しても、double と同じ精度しか有効でないという問題があります。

このため、πを

```
PI = +3.141592653589793239
```

というように定義しても、15 ないし 16 桁しか有効でありません。

この問題を避けるためにも、long　double の精度を必要とするような定数は、Mathma387 の定数を御利用下さい。

１．８　Mathma387 によるアドバンスト・プログラミング

i387_pdcos()や i387_pfcos () は共に引き数と返り値をポインタ渡しにすることにより倍精度の場合で約 1 割程度、単精度では約 2 割程度高速になります。

例：

```
#include <mathm387.h>

        long double ldx, ldy;
        double dx, dy;
        float fx, fy;

        ldx = i87_ldPI() / 4.0;
        fx = dx = i87_dPI() / 4.0;
        i387_pldcos(&ldx, &ldy);
        i387_pdcos(&dx, &dy);
        i387_pfcos(&fx, &fy);
        printf("ldcos(%f) = %25.22LE¥n", ldx, ldy);
        printf(" dcos(%f) = %25.22lE¥n",  dx,  dy);
        printf(" fcos(%f) = %25.22E¥n",   fx,  fy);
```

MSC Ver.6.0 よりサポートされている long double の関数は、値を数値演算コプロセッサのスタックに積んで渡すため、__fac という内部変数にコピーして__fac のオフセット値を返す double の場合と比較して約 10%高速になっています。精度をあげる意味でも、実数の計算は long double で行うことをお勧めします。

ポインタ渡しの long double の関数は扱うバイト数が多いため、バリュー渡しの関数よりは高速ですが、ポインタ渡しの double の関数よりは遅くなります。

また、sin()と cos()を同時に求める様な場合には、i387_ldsincos(), i387_dsincos()、i387_pldsincos(), i387_pdsincos()、i387_pfsincos()の様な関数も用意されています。

１．９　MASM バージョン 5.1 を使ったオブジェクト作成

Microsoft C で/G2（80287 用）オプションを指定しても、8087 という CPU と同期をとるために必要な WAIT 命令がオブジェクトの中の数値演算プロセッサ命令の前に挿入されてしまいます。

/Fa /FPi87 /G2 オプションで 80286 用のアセンブラ・ソースを作成し、MASM（マイクロソフトのマクロアセンブラ）のバージョン 5.1 でアセンブルすることにより、WAIT を取り去ることができ、実行速度は 1 割程度高速になります。ただし、バージョン 4 以前の MASM ではこの様なことができません。

しかし、プログラムによってはＣコンパイラが出力するアセンブラは期待通りのコードを出力しない場合があります。この様な場合は、アセンブラのプログラムを修正する必要があり、アセンブラに関するかなりの知識を必要とします。

main()を含むモジュールはマクロアセンブラでアセンブルすることはできません。通常の様に、必ずＣコンパイラでオブジェクトを作成して下さい。

１．１０　他の言語での使用

Mathma387Ver. 1.0 は、Microsoft C　Ver.6.0 用のライブラリとして作られておりますが、以下の言語からも呼び出すことが可能です。

・Microsoft QuickC Ver.2.0
・Microsoft FORTRAN Version 4.1
・Microsoft QuickBASIC Version 4.5
・Microsoft Macro Assembler Version 5.1

FORTRAN の場合は INTERFACE 文を使うことにより Mathma387 を呼ぶことができます。以下の様に、プログラムの冒頭でヘッダーファイル<mathm387.inc>を$INCLUDE メタコマンドによりインクルードすることにより Mathma387 の関数が使用可能になります。

```
$INCLUDE:'mathm387.inc'
```

QuickBASIC の場合はＤＥＣＬＡＲＥ文を使うことにより Mathma387 を呼ぶことができます。以下の様に、プログラムの冒頭で<mathm387.bi>をインクルードすることにより、Mathma387 の関数を使用することが可能となります。

```
'$INCDLUDE: 'mathm387.bi'
```

注意：

Mathma387 は、32bit CPU 用の特別な命令を含んでいるため、MS　FORTRAN に付属している LINK.EXE は使用できません。MSC Ver.5.1 か MSC　Ver.6.0 等に含まれている Segmented-Executable Linker を使用して下さい。

QuickBASIC では、言語仕様のため、i387_acos という様に'_'を含む関数名は i387acos という用に'_'を含まない関数名に変更されています。

QuickBASIC に付属している Overlay Linker Ver.3.69 では、Mathma387 をリンクすることができます。

Macro Assembler から使用するときは C 言語の仕様に沿って呼んで下さい。

詳細は、各言語のマニュアルの「複数の言語によるプログラミング」の項目を御参照下さい。

１．１１　IEEE 浮動小数点規格

Intel Corporation の 8087 等で使用されている浮動小数点表現は、IEEE（the Institute　of Electrical and Electronics Engineers, Inc：アメリカ電気電子学会）の規格委員会の下にあるマイクロコンピュータ規格委員会のワーキンググループの一つである、タスク P754 によって制定されています。

インテルの数値演算コプロセッサだけでなく、NEC のμPD72191、モトローラの 68881 や 68882、TI、アナログ・ディバイスの ADSP3210 や 3220、NS の 32081、ウェイテックの WTL1164、1165 や 3167 などの数々の数値演算コプロセッサで採用されています。

◆浮動小数点形式

ＩＥＥＥの浮動小数点には 3 種類の表現形式があります。

・単一精度	single-precision	short real	float
・倍精度	double-precision	long real	double
・拡張精度（長倍精度）	extended-precision	temporary real	long double

◆内部表現

これらの 3 種類の浮動小数点は、コンピュータ内部では次の様に表現されています。

	バイト数 Number of bytes	指数 Expornent Length	仮数 Mantisal Length
float	4 bytes (32 bits)	8 bits	23 bits
double	8 bytes (64 bits)	11 bits	52 bits
long double	10 bytes (80 bits)	15 bits	64 bits

この他に、サインビットが 1bit 使用されます。

インテル社の数値演算コプロセッサ 80x87 の内部レジスタでは、80bits で処理を行っていますが、サイリックス社の数値演算コプロセッサ FasMath の内部レジスタは 91bits で処理を行っています。

◆有効桁数

有効桁数は以下の通りです。

	Significant digit	
float	6 - 7 桁	（約 7.2 桁）
double	15 - 16 桁	（約 15.9 桁）
long double	19 桁	（約 19.2 桁）

◆浮動小数点の表現範囲

	最小値	最大値
float	-1.175494351E-038 080800000r	-3.402823466E+038 0ff7fffffr
	+1.175494351E-038 000800000r	+3.402823466E+038 07f7fffffr
double	-2.225073858507201E-308 08010000000000000r	-1.797693134862316E+308 0ffefffffffffffr
	+2.225073858507201E-308 00010000000000000r	+1.797693134862316E+308 07fefffffffffffr
long double	-3.362103143112093506E-4932 080018000000000000000r	-1.189731495357231765E+4932 0fffefffffffffffffffr
	+3.362103143112093506E-4932 000018000000000000000r	+1.189731495357231765E+4932 07ffefffffffffffffffr

◆不定値（INDefined）

flaot	-0.#IND00000E-000 1 11111111 10000000000000B
double	-0.#IND00000000000E-000 1 11111111111 1000....000B
long double	-0.#IND00000000000000E-0000 1 111111111111111 1000....000B

◆ゼロ　　正と負のゼロの表現が可能です。

float	-0.000000000E-000 080000000r +0.000000000E-000 000000000r
double	-0.0000000000000000E-000 08000000000000000r +0.0000000000000000E-000 00000000000000000r
long double	-0.00000000000000000000E-0000 080000000000000000000r +0.00000000000000000000E+0000 000000000000000000000r

◆無限大　　正と負の無限大（INFinity）の表現が可能です。

float	-0.#INF00000E-000 0ff800000r +0.#INF00000E-000 07f800000r
double	-0.#INF00000000000E-000 0fff0000000000000r +0.#INF00000000000E-000 07ff0000000000000r
long double	-0.#INF00000000000000E-0000 0ffff8000000000000000r +0.#INF00000000000000E+0000 07fff8000000000000000r

◆NAN（Not A Number）　NAN（非数）にはSNANとQNANがあります。

float	+0.#QNAN0000E-000 1 11111111 1xxxxxxxxxxxxB +0.#SNAN0000E-000 1 11111111 0xxxxxxxxxxxxB
double	+0.#QNAN0000000000E-000 1 11111111111 1xxx....xxxB +0.#SNAN0000000000E-000 1 11111111111 0xxx....xxxB
long double	+0.#QNAN00000000000000E-0000 1 111111111111111 1xxx....xxxB +0.#SNAN00000000000000E+0000 1 111111111111111 0xxx....xxxB

◆デノーマル

表現できる最小値より小さい数で（指数部が表現範囲より小さい）、仮数部先頭が 1 とならず、正規化実数で表現できなくなってしまった実数のことをいいます。

仮数部の先頭にゼロが入るため、有効ビットが減少して精度が低下します。

float	-0.xxxxxxxxE-038 x 00000000 xxxxxxxxxxxx1B
double	-0.xxxxxxxxxxxxxxxE-308 x 00000000000 xxxx....xx1B
long double	-0.xxxxxxxxxxxxxxxxxxE-4932 x 000000000000000 0xxx....xx1B

１．１２　Mathma387 の定数

１．１２．１　Mathma387 で定義済みのデータ

以下の変数は<mathm387.h>中で外部変数として定義されています。しかも、データ境界はダブルワード（4 の倍数）となっており、高速にアクセスすることができます。

```
double             D_PI = +3.141592653589793
long double       LD_PI = +3.1415926535897932 39
double             D_EXP = +2.718281828459045
long double      LD_EXP = +2.7182818284590452 35
float           FLOAT_M1 = -1.000000000
double        DOUBLE_M1 = -1.000000000000000
long double LDOUBLE_M1 = -1.0000000000000000 00
float           FLOAT_1  = +1.000000000
double        DOUBLE_1  = +1.000000000000000
long double LDOUBLE_1  = +1.0000000000000000 00
float         F_M2_63   = -2.063 = -9.223372E+018
double        D_M2_63   = -2.063 = -9.223372036854775E+018
long double LD_M2_63   = -2.063 = -9.2233720368547758 08E+018
float         F_2_63    = +2.063 = +9.223372E+018
double        D_2_63    = +2.063 = +9.223372036854775E+018
long double LD_2_63    = +2.063 = +9.2233720368547758 08E+018
float         F_MHUGE   = -3.402823466E+038
double        D_MHUGE   = -1.797693134862316E+308
long double LD_MHUGE   = -1.1897314953572317 65E+4932
float         F_HUGE    = +3.402823466E+038
double        D_HUGE    = +1.797693134862316E+308
long double LD_HUGE    = +1.1897314953572317 65E+4932
float         F_MTYNY   = -1.175494351E-038
double        D_MTYNY   = -2.225073858507201E-308
long double LD_MTYNY   = -3.3621031431120935 06E-4932
float         F_TYNY    = +1.175494351E-038
double        D_TYNY    = +2.225073858507201E-308
long double LD_TYNY    = +3.3621031431120935 06E-4932
```

```
float       F_EXPMX   = -8.733654785E+001
double      D_EXPMX   = -7.083964185322641E+002
long double LD_EXPMX  = -1.135513711193302406E+004
float       F_EXPX    = +8.872283936E+001
double      D_EXPX    = +7.097827128933840E+002
long double LD_EXPX   = +1.135652340629414395E+004
float       F_IND     = -1.#IND00000E+000
double      D_IND     = -1.#IND00000000000E+000
long double LD_IND    = -1.#IND00000000000000E+000
float       F_QNAN    = +1.#QNAN0000E+000
double      D_QNAN    = +1.#QNAN0000000000E+000
long double LD_QNAN   = +1.#QNAN0000000000000E+000
float       F_SNAN    = +1.#SNAN0000E+000
double      D_SNAN    = +1.#SNAN0000000000E+000
long double LD_SNAN   = +1.#SNAN0000000000000E+000
float       F_MINF    = -1.#INF00000E+000
double      D_MINF    = -1.#INF00000000000E+000
long double LD_MINF   = -1.#INF00000000000000E+000
float       F_INF     = +1.#INF00000E+000
double      D_INF     = +1.#INF00000000000E+000
long double LD_INF    = +1.#INF00000000000000E+000
```

１．１２．２　Mathma387 の関数による（FPU が返す）定数

以下の定数は数値演算プロセッサが返す値の関数として、マクロ定義されています。

```
long double FPU_LD_PI  = i87_ldPI()   = +3.141592653589793239
double      FPU_D_PI   = i87_dPI()    = +3.141592653589793
long double FPU_LD_L2E = i87_ldL2E()  = +1.442695040888963407E+000
double      FPU_D_L2E  = i87_dL2E()   = +1.442695040888963E+000
long double FPU_LD_L2T = i87_ldL2T()  = +3.321928094887362348E+000
double      FPU_D_L2T  = i87_dL2T()   = +3.321928094887362E+000
long double FPU_LD_LG2 = i87_ldLG2()  = +3.010299956639811952E-001
double      FPU_D_LG2  = i87_dLG2()   = +3.010299956639812E-001
long doubel FPU_LD_LN2 = i87_ldLN2()  = +6.931471805599453094E-001
double      FPU_D_LN2  = i87_dLN2()   = +6.931471805599453E-001
```

１．１３　参考文献

誠文堂新光社，8087 マシン語プログラム集（上巻、下巻）
CQ 出版社，インターフェース　1985 年 3 月号　特集　浮動小数点演算と 8087
CQ 出版社，別冊インターフェース　数値演算プロセッサ
翔泳社　高速科学計算コプロセッサ　80387
Microsoft Macros Assembler Programmer's Guide
Microsoft C Advanced Programming Techniques
A Wiley Press Book, THE 8087 PRIMER
Microsoft Press, The 80386 Book
Sybex, Programming The 80386
秀和システムトレーディング株式会社、80386 ブック
ASCII 出版、Super ASCII　386 マシン徹底活用講座　386 de MS-DOS
技術評論社、　マクロアセンブラプログラミング入門

第二章　リファレンス

C　　i３８７_ｌｄａｃｏｓ()
　　i３８７_ｄａｃｏｓ()
　　i３８７_ｐｌｄａｃｏｓ()
　　i３８７_ｐｄａｃｏｓ()
　　i３８７_ｐｆａｃｏｓ()
・・

概　要　：　アークコサイン（逆余弦）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldacos(long double ldx);
  double _far _cdecl i387_dacos(double ldx);
/* ポインタ版 */
  void _far _cdecl i387_pldacos(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdacos(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfaCos(float _far *fx, float _far *fy);
```

戻り値　：　0.0 からπまでの範囲の値を返します。

解　説　：

　アークコサイン（逆余弦）を計算します。

　結果は、0.0 からπの値を返します。

　引き数 x の値は、-1.0 以上 1.0 以下の範囲でなければなりません。この範囲外の数が渡された場合には、関数の精度に合わせて以下の様に HUGE（表現できる最大の数）が返されます。

　単精度関数　　F_HUGE　(3.402823E+038)
　倍精度関数　　D_HUGE　(1.797693134862316E+308)
　長倍精度関数　LD_HUGE (1.189731495357231765E+4932)

　アークコサインは次の公式を利用して求めています。

$$aCos(x) = -atan(x / sqrt(1.0 - x * x)) + \pi / 2.0 \qquad (|x| <= 1.0)$$

参　照　：

　　i387_ldashin(), i387_dasin(),
　　i387_pldashin(),i387_pdasin(),　i387_pfasin(),
　　i387_ldatan(),　i387_datan(),
　　i387_pldatan(), i387_pdatan(),　i387_pfatan()

C　　　ｉ３８７＿ｌｄａｃｏｓｈ()
ｉ３８７＿ｄａｃｏｓｈ()
ｉ３８７＿ｐｌｄａｃｏｓｈ()
ｉ３８７＿ｐｄａｃｏｓｈ()
ｉ３８７＿ｐｆａｃｏｓｈ()

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　ハイパボリック・アークコサイン（逆双曲線余弦）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldacosh(long double ldx);
  double _far _cdecl i387_daCosh(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldacosh(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdacosh(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfacosh(float _far *fx, float _far *fy);
```

戻り値　：　0.0 以上の値を返します。

解　説　：

ハイパボリックアークコサイン（逆双曲線余弦）を計算します。

aCosh()の解は正と負の二つの解がありますが、正のみの値を返します。

引き数ｘの値は、1.0 以上でなければなりません。この範囲外の数が渡された場合には、関数の精度に合わせて以下の様にMHUGE（表現できる負の最大の数）が返されます。

単精度関数	F_MHUGE (-3.402823E+038)
倍精度関数	D_MHUGE (-1.797693134862316E+308)
長倍精度関数	LD_MHUGE (-1.189731495357231765E+4932)

アークコサインは次の公式を利用して求めています。

```
acosh(x) = +log(x + sqrt(x * x - 1.0))  (x >= 1.0)
acosh(x) = -log(x + sqrt(x * x - 1.0))  (x >= 1.0)
```

参　照　：

i387_ldashinh(),i387_dasinh(),
i387_pldashin(),i387_pdasinh(), i387_pfasinh(),
i387_ldatanh(), i387_datanh(),
i387_pldatanh(),i387_pdatanh(), i387_pfatanh()

C　　i３８７_ｌｄａｓｉｎ()
　　i３８７_ｄａｓｉｎ()
　　i３８７_ｐｌｄａｓｉｎ()
　　i３８７_ｐｄａｓｉｎ()
　　i３８７_ｐｆａｓｉｎ()
・・

概　要　：　アークサイン（逆正弦）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldsin(long double ldx);
  double _far _cdecl i387_dsin(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldsin(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdsin(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfsin(float _far *fx, float _far *fy);
```

戻り値　：　-π／2.0 からπ／2.0 までの範囲の値を返します。

解　説　：

　アークサイン（逆正弦）を計算します。

　結果は、-π／2.0 からπ／2.0 の値を返します。

　引き数 x の値は、-1.0 以上 1.0 以下の範囲でなければなりません。この範囲外の数が渡された場合には、関数の精度に合わせて以下の様に HUGE（表現できる最大の数）が返されます。

　単精度関数　　F_HUGE　(3.402823E+038)
　倍精度関数　　D_HUGE　(1.797693134862316E+308)
　長倍精度関数　LD_HUGE (1.189731495357231765E+4932)

　アークコサインは次の公式を利用して求めています。

```
        asin(x) = atan(x / sqrt(1.0 - x * x))    ( | x | <= 1.0)
```

参　照　：

　　i387_ldacos(),　i387_dacos(),
　　i387_pldacos(), i387_pdacos(),　i387_pfacos(),
　　i387_ldatan(),　i387_datan(),
　　i387_pldatan(), i387_pdatan(),　i387_pfatan()

C　　i３８７_ｌｄａｓｉｎｈ()
　　i３８７_ｄａｓｉｎｈ()
　　i３８７_ｐｌｄａｓｉｎｈ()
　　i３８７_ｐｄａｓｉｎｈ()
　　i３８７_ｐｆａｓｉｎｈ()

・・

概　要　：　ハイパボリック・アークコサイン（逆双曲線正弦）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C言語と互換な関数 */
  long double _far _cdecl i387_ldsinh(long double ldx);
  double _far _cdecl i387_dsinh(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldsinh(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdsinh(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfsinh(float _far *fx, float _far *fy);
```

戻り値　：　引き数ｘのハイパボリック・アークコサインの値を返します。

解　説　：

アークサイン（逆双曲線正弦）を計算します。

ハイパボリック・アークコサインは次の公式を利用して求めています。

```
asinh(x) = log(x + sqrt(x * x + 1.0))
```

参　照　：

i387_ldacosh(), i387_dacosh(),
i387_pldacosh(),i387_pdacosh(), i387_pfacosh(),
i387_ldatanh(), i387_datanh(),
i387_pldatanh(),i387_pdatanh(), i387_pfatanh()

C　　i３８７_ｌｄａｔａｎ()
　　i３８７_ｄａｔａｎ()
　　i３８７_ｐｌｄａｔａｎ()
　　i３８７_ｐｄａｔａｎ()
　　i３８７_ｐｆａｔａｎ()

・・

概　要　：　アークタンジェント（逆正接）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldatan(long double ldx);
  double _far _cdecl i387_datan(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldatan(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdatan(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfatan(float _far *fx, float _far *fy);
```

戻り値　：　-π／2.0 からπ／2.0 までの範囲の値を返します。

解　説　：

アークタンジェント（逆正接）を計算します。

結果は、-π／2.0 からπ／2.0 の値を返します。

数値演算コプロセッサ 8087 では

```
0.0 < ST(1) / ST(0) < 1.0
```

という制限がありましたが、80387 ではこの制限が無くなりました。

アークコサインは次の公式を利用して求めることもできます。

```
atan(x) = log((1.0 + x) / (1.0 - x)) / 2.0
```

参　照　：

i387_ldaCos(), i387_daCos(),
i387_pldaCos(), i387_pdaCos(), i387_pfaCos(),
i387_ldashin(), i387_dasin(),
i387_pldasin(), i387_pdasin(), i387_pfasin(),
i387_ldatan2(), i387_datan2(),
i387_pldatan2(),i387_pdatan2(), i387_pfatan2()

C　　ｉ３８７_ｌｄａｔａｎ２ ()
　　ｉ３８７_ｄａｔａｎ２ ()
　　ｉ３８７_ｐｌｄａｔａｎ２ ()
　　ｉ３８７_ｐｄａｔａｎ２ ()
　　ｉ３８７_ｐｆａｔａｎ２ ()

・・

概　要　：　ｙ／ｘのアークコサイン（逆正接）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldatan2(long double ldy, long double ldx);
  double _far _cdecl i387_datan2(double dy, double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldatan2(long double _far *ldy, long double _far *ldx,
        long double _far *ldz);
  void _far _cdecl i387_pdatan2(double _far *dy, double _far *dx, double _far *dz);
  void _far _cdecl i387_pfatan2(float _far *fy, float _far *fx, float _far *fz);
```

戻り値　：　－πからπまでの範囲の値を返します。

解　説　：

ｙ／ｘのアークタンジェントを計算します。
結果は、-πからπの値を返します。
atan()では-π／2.0 からπ／2.0 までの結果しかえることができません。
atan2(0.0/0.0)は 0.0 を返します。

参　照　：

i387_ldaCos(),　i387_daCos(),
i387_pldaCos(), i387_pdaCos(),　i387_pfaCos(),
i387_ldasin(),　i387_dasin(),
i387_pldasin(), i387_pdasin(),　i387_pfasin(),
i387_ldatan(),　i387_datan(),
i387_pldatan(), i387_pdatan(),　i387_pfatan()

C　　i387_ldatanh()
　　i387_datanh()
　　i387_pldatanh()
　　i387_pdatanh()
　　i387_pfatanh()

・・

概　要　：　ハイパボリック・アークタンジェント（逆双曲線正接）を計算します。

書　式　：

```
#include <mathm387.h>
/* C 言語と互換な関数 */
long double _far _cdecl i387_ldatanh(long double ldx);
double _far _cdecl i387_datanh(double dx);
/* ポインタ版 */
void _far _cdecl i387_pldatanh(long double _far *ldx, long double _far *ldy);
void _far _cdecl i387_pdatanh(double _far *dx, double _far *dy);
void _far _cdecl i387_pfatanh(float _far *fx, float _far *fy);
```

戻り値　：　引き数ｘのハイパボリック・アークタンジェントを返します。

解　説　：

ハイパボリック・アークタンジェント（逆双曲線正接）を計算します。

引き数 x の値は、-1.0 を越え、かつ 1.0 を越えないの範囲でなければなりません。この範囲外の数が渡された場合には、関数の精度に合わせて以下の様に HUGE（表現できる最大の数）が返されます。

単精度関数	F_HUGE (3.402823E+038)
倍精度関数	D_HUGE (1.797693134862316E+308)
長倍精度関数	LD_HUGE (1.189731495357231765E+4932)

ハイパボリック・アークタンジェントは次の公式を利用して求めています。

$$atanh(x) = \log((1.0 + x) + (1.0 - x)) / 2.0 \qquad (|x| < 1.0)$$

参　照　：

i387_ldacosh(), i387_dacosh(),
i387_pldacosh(), i387_pdacosh(), i387_pfacosh(),
i387_ldasinh(), i387_dasinh(),
i387_pldasinh(), i387_pdasinh(), i387_pfasinh()

C　　i３８７_ｌｄｃｏｓ()
　　i３８７_ｄｃｏｓ()
　　i３８７_ｐｌｄｃｏｓ()
　　i３８７_ｐｄｃｏｓ()
　　i３８７_ｐｆｃｏｓ()

・・

概　要　：　コサイン（余弦）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far i387_ldCos(long double ldx);
  double _far i387_dCos(double dx);
/* ポインタ版 */
  void _far i387_pldCos(long double _far *ldx, long double _far *ldy);
  void _far i387_pdCos(double _far *dx, double _far *dy);
  void _far i387_pfCos(float _far *fx, float _far *fy);
```

戻り値　：　-1.0 以上 1.0 以下の範囲の値を返します。

解　説　：

　コサイン（余弦）を計算します。

　結果は、-1.0 以上 1.0 以下の値を返します。

　引き数 x の値は、-2.063 以上 2.063 以下の範囲でなければなりません。-2.063 と 2.063 の値は Mathma387 では次の外部変数で定義してあるのでこれを利用すると便利です。

```
float       F_M2_63  = -2.063 = -9.223372E+018
double      D_M2_63  = -2.063 = -9.223372036854775E+018
long double LD_M2_63 = -2.063 = -9.22337203685477580 8E+018
float       F_2_63   = +2.063 = +9.223372E+018
double      D_2_63   = +2.063 = +9.223372036854775E+018
long double LD_2_63  = +2.063 = +9.22337203685477580 8E+018
```

参　照　：

　　i387_ldsin(),　i387_dsin(),
　　i387_pldsin(),　i387_pdsin(),　i387_pfsin(),
　　i387_ldtan(),　i387_dtan(),
　　i387_pldtan(),　i387_pdtan(),　i387_pftan()

C　　　　ｉ３８７＿ｌｄｃｏｓｈ()
　　　　　ｉ３８７＿ｄｃｏｓｈ()
　　　　　ｉ３８７＿ｐｌｄｃｏｓｈ()
　　　　　ｉ３８７＿ｐｄｃｏｓｈ()
　　　　　ｉ３８７＿ｐｆｃｏｓｈ()

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　ハイパボリック・コサイン（双曲線余弦）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldcosh(long double ldx);
  double _far _cdecl i387_dcosh(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldcosh(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdcosh(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfcosh(float _far *fx, float _far *fy);
```

戻り値　：　1.0 以上の値を返します。

解　説　：

ハイパボリック・コサインを計算します。
結果は、1.0 以上の値を返します。
引き数 x の値は、以下の範囲でなければなりません。

単精度関数	最小値	F_EXPMX = -8.733654785E+001
	最大値	F_EXPX = +8.872283936E+001
倍精度関数	最小値	D_EXPMX = -7.083964185322641E+002
	最大値	D_EXPX = +7.097827128933840E+002
長倍精度関数	最小値	LD_EXPMX = -1.135513711193302406E+004
	最大値	LD_EXPX = +1.135652340629414395E+004

アークコサインは次の公式を利用して求めています。

```
cosh(x) = (exp(x) + 1.0 / exp(x)) / 2.0
```

参　照　：

i387_ldsinh(), i387_dsinh(),
i387_pldsinh(), i387_pdsinh(), i387_pfsinh(),

i387_ldtanh(), i387_dtanh(),
i387_pldtanh(), i387_pdtanh(), i387_pftanh()

C　　ｉ３８７_ｌｄｅｘｐ()
　　ｉ３８７_ｄｅｘｐ()
　　ｉ３８７_ｐｌｄｅｘｐ()
　　ｉ３８７_ｐｄｅｘｐ()
　　ｉ３８７_ｐｆｅｘｐ()

・・

概　要　：　引き数 x の指数関数 ex を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldexp(long double ldx);
  double _far _cdecl i387_dexp(double dx);
/* ポインタ版 */
  void _far _cdecl i387_lpdexp(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdexp(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfexp(float _far *fx, float _far *fy);
```

戻り値　：　引き数 x の指数関数ｅｘを返します。

解　説　：

引き数 x の指数関数 ex を計算します。

引き数 x の値は、以下の範囲でなければなりません。

単精度関数	最小値	F_EXPMX	= -8.733654785E+001
	最大値	F_EXPX	= +8.872283936E+001
倍精度関数	最小値	D_EXPMX	= -7.083964185322641E+002
	最大値	D_EXPX	= +7.097827128933840E+002
長倍精度関数	最小値	LD_EXPMX	= -1.135513711193302406E+004
	最大値	LD_EXPX	= +1.135652340629414395E+004

ex は次の式を利用して求めています。

```
exp(x) = pow(2.0, log10(e * x))
```

参　照　：

i387_ldlog(),　i387_dlog(),
i387_pldlog(),　i387_pdlog(),　i387_pflog()

Ｃ　　　　ｉ３８７＿ｌｄｌｏｇ（）
ｉ３８７＿ｄｌｏｇ（）
ｉ３８７＿ｐｌｄｌｏｇ（）
ｉ３８７＿ｐｄｌｏｇ（）
ｉ３８７＿ｐｆｌｏｇ（）

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　引き数ｘの自然対数を計算します。

書　式　：

```
  #include <mathm387.h>
/* C言語と互換な関数 */
  long double _far _cdecl i387_ldlog(long double ldx);
  double _far _cdecl i387_dlog(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldlog(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdlog(double _far *dx, double _far *dy);
  void _far _cdecl i387_pflog(float _far *fx, float _far *fy);
```

戻り値　：　引き数ｘの自然対数の値を返します。

解　説　：

引き数ｘの自然対数を計算します。

引き数ｘの値は、0.0を越える値でなければなりません。この範囲外の数が渡された場合には、関数の精度に合わせて以下の様にMHUGE（表現できる負の最大の数）が返されます。

単精度関数	F_MHUGE	-3.402823E+038)
倍精度関数	D_MHUGE	-1.797693134862316E+308)
長倍精度関数	LD_MHUGE	(-1.189731495357231765E+4932)

参　照　：

i387_ldexp(),　i387_dexp(),
i387_pldexp(),　i387_pdexp(),　i387_pfexp(),
i387_ldlog10(),　i387_dlog10(),
i387_pldlog10(), i387_pdlog10(),　i387_pflog10()

C　　　ｉ３８７＿ｌｄｌｏｇ１０（）
ｉ３８７＿ｄｌｏｇ１０（）
ｉ３８７＿ｐｌｄｌｏｇ１０（）
ｉ３８７＿ｐｄｌｏｇ１０（）
ｉ３８７＿ｐｆｌｏｇ１０（）

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　引き数ｘの常用対数を計算します。

書　式　：

```
  #include <mathm387.h>
/* C言語と互換な関数 */
  long double _far _cdecl i387_ldlog10(long double ldx);
  double _far _cdecl i387_dlog10(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldlog(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdlog(double _far *fx, double _far *dy);
  void _far _cdecl i387_pflog(float _far *fx, float _far *fy);
```

戻り値　：　引き数ｘの常用対数の値を返します。

解　説　：

引き数ｘの常用対数を計算します。

引き数ｘの値は、0.0を越える値でなければなりません。この範囲外の数が渡された場合には、関数の精度に合わせて以下の様にMHUGE（表現できる負の最大の数）が返されます。

単精度関数	F_MHUGE	(-3.402823E+038)
倍精度関数	D_MHUGE	(-1.797693134862316E+308)
長倍精度関数	LD_MHUGE	(-1.189731495357231765E+4932)

参　照　：

i387_ldexp(),　i387_dexp(),
i387_pldexp(),　i387_pdexp(),　i387_pfexp(),
i387_ldlog(),　i387_dlog(),
i387_pldlog(),　i387_pdlog(),　i387_pflog()

Ｃ　　　ｉ３８７＿ｌｄｓｉｎ()
　　　　ｉ３８７＿ｄｓｉｎ()
　　　　ｉ３８７＿ｐｌｄｓｉｎ()
　　　　ｉ３８７＿ｐｄｓｉｎ()
　　　　ｉ３８７＿ｐｆｓｉｎ()

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　サイン（正弦）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldsin(long double ldx);
  double _far _cdecl i387_dsin(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldsin(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdsin(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfsin(float _far *fx, float _far *fy);
```

戻り値　：　-1.0 以上 1.0 以下の範囲の値を返します。

解　説　：

サイン（正弦）を計算します。

結果は、-1.0 以上 1.0 以下の値を返します。

引き数 x の値は、-2.063 以上 2.063 以下の範囲でなければなりません。-2.063 と 2.063 の値は Mathma387 では次の外部変数で定義してあるのでこれを利用すると便利です。

```
float       F_M2_63  = -2.063 = -9.223372E+018
double      D_M2_63  = -2.063 = -9.223372036854775E+018
long double LD_M2_63 = -2.063 = -9.223372036854775808E+018
float       F_2_63   = +2.063 = +9.223372E+018
double      D_2_63   = +2.063 = +9.223372036854775E+018
long double LD_2_63  = +2.063 = +9.223372036854775808E+018
```

参　照　：

i387_ldsin(),　i387_dsin(),
i387_pldsin(), i387_pdsin(),　i387_pfsin(),
i387_ldtan(),　i387_dtan(),
i387_pldtan(), i387_pdtan(),　i387_pftan()

Ｃ　　　　ｉ３８７＿ｌｄｓｉｎCｏｓ（）
　　　　　ｉ３８７＿ｄｓｉｎCｏｓ（）
　　　　　ｉ３８７＿ｐｌｄｓｉｎCｏｓ（）
　　　　　ｉ３８７＿ｐｄｓｉｎCｏｓ（）
　　　　　ｉ３８７＿ｐｆｓｉｎCｏｓ（）

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　サイン（正弦）とコサイン（余弦）を同時に計算します。

書　式　：

```
  #include <mathm387.h>
/* C言語と互換な関数 */
  long double _far _cdecl i387_ldsincos(long double ldx,
        long double _far *dsin, long double _far *dcos);
  double _far _cdecl i387_dsincos(double dx,
        double _far *dsin, double _far *dcos);
/* ポインタ版 */
  void _far _cdecl i387_pldsincos(long double _far *ldx,
        long double _far *ldsin, long double _far *ldcos);
  void _far _cdecl i387_pdsincos(double _far *dx,
        double _far *dsin, double _far *dcos);
  void _far _cdecl i387_pfsincos(float _far *fx,
        float _far *fcos, float _far *fcos);
```

戻り値　：　sin()とcos()を、ポインタ引き数渡しで同時に計算し値を戻します。

解　説　：

80387が持つ、FSINCOS命令を使用してサインとコサインを同時に計算します。戻り値は二つあるため、ポインタ渡しの引き数を通して計算結果を戻します。

結果は、-1.0以上1.0以下の値となります。

引き数xの値は、-2.063以上2.063以下の範囲でなければなりません。-2.063と2.063の値はMathma387では次の外部変数で定義してあるのでこれを利用すると便利です。

```
float       F_M2_63  = -2.063 = -9.223372E+018
double      D_M2_63  = -2.063 = -9.223372036854775E+018
long double LD_M2_63 = -2.063 = -9.223372036854775808E+018
float       F_2_63   = +2.063 = +9.223372E+018
double      D_2_63   = +2.063 = +9.223372036854775E+018
long double LD_2_63  = +2.063 = +9.223372036854775808E+018
```

参 照 ：

i387_ldcos(), i387_dcos(),
i387_pldcos(), i387_pdcos(), i387_pfcos(),
i387_ldsin(), i387_dsin(),
i387_pldsin(), i387_pdsin(), i387_pfsin()

Ｃ　　　ｉ３８７＿ｌｄｓｉｎｈ()
　　　　ｉ３８７＿ｄｓｉｎｈ()
　　　　ｉ３８７＿ｐｌｄｓｉｎｈ()
　　　　ｉ３８７＿ｐｄｓｉｎｈ()
　　　　ｉ３８７＿ｐｆｓｉｎｈ()

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　ハイパボリックサイン（双曲線正弦）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C言語と互換な関数 */
  long double _far _cdecl i387_ldsinh(long double ldx);
  double _far _cdecl i387_dsinh(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldsinh(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdsinh(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfsinh(float _far *fx, float _far *fy);
```

戻り値　：　引き数ｘのハイパボリック・サインの値を返します。

解　説　：

ハイパボリック・サイン（双曲線正弦）を計算します。

引き数ｘの値は、以下の範囲でなければなりません。

単精度関数	最小値	F_EXPMX	= -8.733654785E+001
	最大値	F_EXPX	= +8.872283936E+001
倍精度関数	最小値	D_EXPMX	= -7.083964185322641E+002
	最大値	D_EXPX	= +7.097827128933840E+002
長倍精度関数	最小値	LD_EXPMX	= -1.135513711193302406E+004
	最大値	LD_EXPX	= +1.135652340629414395E+004

ハイパボリック・サインは次の公式を利用して求めています。

```
sinh(x) = ( exp(x) - 1.0 / exp(x) ) / 2.0
```

参　照　：

i387_ldcosh(), i387_dcosh(),
i387_pdcosh(), i387_pdcosh(), i387_pfcosh(),
i387_ldtanh(), i387_dtanh(),
i387_pldtanh(), i387_pdtanh(), i387_pftanh()

C　　　ｉ３８７＿ｌｄｓｑｒｔ（）
　　　　ｉ３８７＿ｄｓｑｒｔ（）
　　　　ｉ３８７＿ｐｌｄｓｑｒｔ（）
　　　　ｉ３８７＿ｐｄｓｑｒｔ（）
　　　　ｉ３８７＿ｐｆｓｑｒｔ（）

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　引き数ｘの平方根を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldsqrt(long double ldx);
  double _far _cdecl i387_dsqrt(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldsqrt(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdsqrt(double _far *dx, double _far *dy);
  void _far _cdecl i387_pfsqrt(float _far *fx, float _far *fy);
```

戻り値　：　0.0 以上範囲の値を返します。

解　説　：

　引き数ｘの平方根を計算します。

　結果は、0.0 以上の値を返します。

　引き数ｘの値は、0.0 以上の範囲でなければなりません。この範囲外の数が渡された場合には、関数の精度に合わせて以下の様にMHUGE（表現できる負の最大の数）が返されます。

　単精度関数　　F_MHUGE (-3.402823E+038)
　倍精度関数　　D_MHUGE (-1.797693134862316E+308)
　長倍精度関数　LD_MHUGE　　(-1.189731495357231765E+4932)

Ｃ　　　　ｉ３８７＿ｌｄｔａｎ()
　　　　　ｉ３８７＿ｄｔａｎ()
　　　　　ｉ３８７＿ｐｌｄｔａｎ()
　　　　　ｉ３８７＿ｐｄｔａｎ()
　　　　　ｉ３８７＿ｐｆｔａｎ()

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

概　要　：　タンジェント（正接）を計算します。

書　式　：

```
  #include &lt;mathm387.h&gt;
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldtan(long double ldx);
  double _far _cdecl i387_dtan(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldtan(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdtan(double _far *dx, double _far *dy);
  void _far _cdecl i387_pftan(float _far *fx, float _far *fy);
```

戻り値　：　引き数 x のタンジェントの値を返します。

解　説　：

　タンジェント（正接）を計算します。

　引き数 x の値は、-2.063 以上 2.063 以下の範囲でなければなりません。-2.063 と 2.063 の値は Mathma387 では次の外部変数で定義してあるのでこれを利用すると便利です。

```
float       F_M2_63  = -2.063 = -9.223372E+018
double      D_M2_63  = -2.063 = -9.223372036854775E+018
long double LD_M2_63 = -2.063 = -9.223372036854775808E+018
float       F_2_63   = +2.063 = +9.223372E+018
double      D_2_63   = +2.063 = +9.223372036854775E+018
long double LD_2_63  = +2.063 = +9.223372036854775808E+018
```

参　照　：

　　　　i387_ldcos(),　i387_dcos(),
　　　　i387_pldcos(),　i387_pdcos(),　i387_pfcos(),
　　　　i387_ldsin(),　i387_dsin(),
　　　　i387_pldsin(),　i387_pdsin(),　i387_pfcos()

C　　i３８７_ｌｄｔａｎｈ()
　　i３８７_ｄｔａｎｈ()
　　i３８７_ｐｌｄｔａｎｈ()
　　i３８７_ｐｄｔａｎｈ()
　　i３８７_ｐｆｔａｎｈ()

・・

概　要　：　ハイパボリック・タンジェント（双曲線正接）を計算します。

書　式　：

```
  #include <mathm387.h>
/* C 言語と互換な関数 */
  long double _far _cdecl i387_ldtanh(long double ldx);
  double _far _cdecl i387_dtanh(double dx);
/* ポインタ版 */
  void _far _cdecl i387_pldtanh(long double _far *ldx, long double _far *ldy);
  void _far _cdecl i387_pdtanh(double _far *dx, double _far *dy);
  void _far _cdecl i387_pftanh(float _far *fx, float _far *fy);
```

戻り値　：　-1.0 を越え 1.0 を越えないの範囲の値を返します。

解　説　：

ハイパボリック・タンジェント（双曲線正接）を計算します。
結果は、-1.0 を越え 1.0 を越えない範囲の値を返します。
引き数 x の値は、以下の範囲でなければなりません。

単精度関数　　最小値　F_EXPMX = -8.733654785E+001
　　最大値　F_EXPX　= +8.872283936E+001

倍精度関数　　最小値　D_EXPMX = -7.083964185322641E+002
　　最大値　D_EXPX　= +7.097827128933840E+002

長倍精度関数　最小値　LD_EXPMX = -1.135513711193302406E+004
　　最大値　LD_EXPX　= +1.135652340629414395E+004

ハイパボリック・サインは次の公式を利用して求めています。

$$\tanh(x) = (\exp(x) - 1.0 / \exp(x)) / (\exp(x) + 1.0 / \exp(x))$$
$$= 1.0 - 2.0 / (\exp(x) * \exp(x) + 1.0)$$

参　照　：

i387_pdCosh(), i387_dCosh(),

i387_pldcosh(), i387_pdcosh(), i387_pfcosh(),
i387_ldsinh(), i387_dsinh(),
i387_pldsinh(), i387_pdsinh(), i387_pfsinh()

Ｃ　　ｉ８７＿ｌｄＰＩ（）
　　ｉ８７＿ｄＰＩ（）

・・

概　要　：　円周率（π）の値を返します。

書　式　：

```
#include <mathm387.h>
long double _far _cdecl i87_ldPI(void);
double _far _cdecl i87_dPI(void);
```

戻り値　：　FPU が返すπ（円周率）の値、3.141592653589793851 を返します。

解　説　：

i87_ldPI()および i87_dPI()は円周率（π）の値を返します。
返す値は、長倍精度の 3.14159265358979323851 または倍精度の 3.141592653589793 です。
この関数は、FPU が 80387 でなくとも、8087 や 80287 でも使うことが可能です。

参　照　：

i87_ldL2E(),　i87_dL2E(),
i87_ldL2T(),　i87_dL2T(),
i87_ldLG2(),　i87_dLG2(),
i87_ldLN2(),　i87_dLN2()

Ｃ　　ｉ８７＿ｌｄＬ２Ｅ()
　　ｉ８７＿ｄＬ２Ｅ()

・・

概　要　：　LOG_2eの値を返します。

書　式　：

```
#include <mathm387.h>
long double _far _cdecl i87_ldL2E(void);
double _far _cdecl i87_dL2E(void);
```

戻り値　：　FPU が返す LOG2e の値、1.4426950408896340739 を返します。

解　説　：

i87_ldL2E()および i87_dL2E()は log2ｅの値を返します。
返す値は、長倍精度の 1.44269501408896340739 または倍精度の 1.442695014088963 です。
この関数は、FPU が 80387 でなくとも、8087 や 80287 でも使うことが可能です。

参　照　：

i87_ldPI(),　i87_dPI(),
i87_ldL2T(),　i87_dL2T(),
i87_ldLG2(),　i87_dLG2(),
i87_ldLN2(),　i87_dLN2()

Ｃ　　ｉ８７＿ｌｄＬ２Ｔ()
　　ｉ８７＿ｄＬ２Ｔ()

・・

概　要　：　$LOG_2$１０の値を返します。

書　式　：

```
#include <mathm387.h>
long double _far _cdecl i87_ldL2T(void);
double _far _cdecl i87_dL2T(void);
```

戻り値　：　ＦＰＵが返すＬＯＧ２１０の値、3.32192809488736234781 を返します。

解　説　：

i87_ldL2T()および i87_dL2T()は log210.0 の値を返します。
返す値は、長倍精度の 3.32192809488736234781 または倍精度の 3.321928094887362 です。
この関数は、ＦＰＵが 80387 でなくとも、8087 や 80287 でも使うことが可能です。

参　照　：

i87_ldPI(),　i87_dPI(),
i87_ldL2E(),　i87_dL2E(),
i87_ldLG2(),　i87_dLG2(),
i87_ldLN2(),　i87_dLN2()

Ｃ　　ｉ８７＿ｌｄＬＧ２（）
　　ｉ８７＿ｄＬＧ２（）

・・

概　要　：　$LOG_{10}2$の値を返します。

書　式　：

```
#include <mathm387.h>
long double _far _cdecl i87_ldLG2(void);
double _far _cdecl i87_dLG2(void);
```

戻り値　：　ＦＰＵが返す$LOG_{10}2$の値、0.30102999566398119522 6 を返します。

解　説　：

i87_ldLG2()および i87_dLG2()は log10(2.0)の値を返します。
返す値は、長倍精度の 0.30102999566398119522 6 または倍精度の 0.3010299956639812 です。
この関数は、ＦＰＵが 80387 でなくとも、8087 や 80287 でも使うことが可能です。

参　照　：

i87_ldPI(),　i87_dPI(),
i87_ldL2E(),　i87_dL2E(),
i87_ldL2T(),　i87_dL2T(),
i87_ldLN2(),　i87_dLN2()

Ｃ　　ｉ８７＿ｌｄＬＮ２（）
　　ｉ８７＿ｄＬＮ２（）

・・

概　要　：　ＬＯＧｅ２の値を返します。

書　式　：

```
#include <mathm387.h>
long double _far _cdecl i87_ldLN2(void);
double _far _cdecl i87_dLN2(void);
```

戻り値　：　ＦＰＵが返すＬＯＧｅ２の値、0.69314718055994530942 を返します。

解　説　：

i87_ldLN2()および i87_d()は log(2.0)の値を返します。
返す値は、長倍精度の 0.69314718055994530942 または倍精度の 0.6931471805599453 です。
この関数は、ＦＰＵが 80387 でなくとも、8087 や 80287 でも使うことが可能です。

参　照　：

i87_ldL2E(),　i87_dL2E(),
i87_ldL2T(),　i87_dL2T(),
i87_ldLG2(),　i87_dLG2(),
i87_ldLN2(),　i87_dLN2()

C　　ｋｉｎｄｃｐｕ()
OS　　ＫＩＮＤＣＰＵ
・・

機　能　：　CPU の種類を取得します。

書　式　：

```
#include <mathm387.h>
int _far _cdecl kindcpu(void);
int _far _cdecl kindcpu(void);
```

戻り値　：　（int 型）　1　CPU は 8086 または V30
　　2　CPU は 80186
　　3　CPU は 80286
　　4　CPU は 80386

文　例　：　C　: int cpu = kindcpu();
　　<cpu>に現在使用している CPU の種類を代入します。

解　説　：

パーソナル・コンピュータが使用している CPU（Central Processing Unit）を調べて、CPU の種類を取得します。

CPU の違いによって次の値を返します。

CPU の種類	戻り値
8086 または V30	1
80186	2
80286	3
80386	4

OS のコマンドを実行した場合も、C 言語のときと同様な値を返します。

参　照　：　kindfpu()

C　　kindfpu()　Floating Point Unit
OS　　KINDFPU

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

機　能　：　FPU（浮動小数点演算プロセッサ）の種類を取得します。

書　式　：

```
#include <mathm387.h>
int _far _cdecl kindfpu(void);
int _far _cdecl kindfpu(void);
```

戻り値　：　（int 型）　0　FPU はありません
　　1　FPU は 8087 です
　　2　FPU は 80287 です
　　3　FPU は 80387 です

文　例　：　C　: int fpu = kindfpu();
　　<fpu>に現在使用している FPU の種類を代入します。

解　説　：

パーソナル・コンピュータが使用している FPU（Floating Point Unit）を調べて、FPU の種類を取得します。
FPU の違いによって次の値を返します。

FPU の種類	戻り値
8087	1
80287	2
80387	3

OS のコマンドを実行した場合も、C 言語のときと同様な値を返します。

コンパイル時にオプション'/FPi87'または'/FPC87'を指定すると、数値演算コプロセッサを搭載したコンピュータ用のコードを生成します。このオプションを指定すると、プログラムが高速かつコンパクトになります。しかし、数値演算コプロセッサが無い場合には、プログラムの冒頭で自動的にエラーメッセージを表示して MS-DOS に戻ってしまうため、kindfpu()によって数値演算コプロセッサが搭載されているかどうかのチェックはできなくなります。

参　照　：
　　kindcpu()

発行者　　土橋英三
発行所　　株式会社アークブレイン
http://arcbrain.jp/
〒151-0072 東京都渋谷区笹塚 1 丁目 62 番 3 号 アルス笹塚
電話　03-375-8968
FAX　03-375-8767

1990 年 8 月第一版発行
1990 年 11 月第二版発行
2003 年 11 月 27 日第三版発行
2009 年 10 月 6 日 復刻版