お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

アプリケーション・ノート

μPD77016 ファミリ

ディジタル・シグナル・プロセッサ

ライブラリ編

対象デバイス

μPD77016
μPD77018A
μPD77019
μPD77110
μPD77111
μPD77112
μPD77113A
μPD77114
μPD77115
μPD77210
μPD77213

資料番号 U12021JJ2V0AN00（第 2 版）
発行年月 August 2001 NS CP（K）

〔メ　モ〕

目次要約

本製品のうち，外国為替および外国貿易管理法の規定により規制貨物等（または役務）に該当するものについては，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

- **本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。**
- **文書による当社の承諾なしに本資料の転載複製を禁じます。**
- **本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。**
- **本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。**
- **当社は品質，信頼性の向上に努めていますが，半導体製品はある確率で故障が発生します。当社半導体製品の故障により結果として，人身事故，火災事故，社会的な損害等を生じさせない冗長設計，延焼対策設計，誤動作防止設計等安全設計に十分ご注意願います。**
- **当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定して頂く「特定水準」に分類しております。また，各品質水準は以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認の上ご使用願います。**

  標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

  特別水準：輸送機器（自動車，列車，船舶等），交通用信号機器，防災／防犯装置，各種安全装置，生命維持を直接の目的としない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート／データ・ブック等の資料で，特に品質水準の表示がない場合は標準水準製品であることを表します。当社製品を上記の「標準水準」の用途以外でご使用をお考えのお客様は，必ず事前に当社販売窓口までご相談頂きますようお願い致します。

M7　98.8

本版で改訂された主な箇所

箇　所	内　　容
全般	対象デバイスにμ PD77111 ファミリ（μ PD77110, 77111, 77112, 77113A, 77114, 77115），μ PD77210 ファミリ（μ PD77210, 77213）を追加
p.12	1.2　ライブラリのインストールに説明を追加

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

対象者　このアプリケーション・ノートは，μPD77016 ファミリの機能を理解し，それを用いたアプリケーション・プログラムを設計するユーザを対象としています。

μPD77016 ファミリは，μPD7701x ファミリ（μPD77015, 77016, 77017, 77018A, 77019），μPD77111 ファミリ（μPD77110, 77111, 77112, 77113A, 77114, 77115），およびμPD77210 ファミリ（μPD77210, 77213）の総称です。

目　的　このアプリケーション・ノートは，μPD77016 ファミリの基礎的な機能について，応用プログラムを用いてユーザに理解していただくことを目的としています。なお，掲載したプログラムは例示的に示したものであり，量産設計を対象にしたものではありません。

構　成　このアプリケーション・ノートは，大きく分けて次の内容で構成されています。

第 1 章　概　要
第 2 章　供給ファイルの内容
第 3 章　固定小数点演算ライブラリ

読み方　このマニュアルの読者は，電気，論理回路やマイクロコンピュータ，C 言語に関する一般的知識が必要となります。

μPD7701x ファミリのハードウエア機能を知りたいとき
→**μPD7701x ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください。

μPD77111 ファミリのハードウエア機能を知りたいとき
→**μPD77111 ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください。

μPD77016 ファミリの命令機能を知りたいとき
→**μPD77016 ファミリ　ユーザーズ・マニュアル　命令編**を参照してください。

凡　例

注	**：本文中につけた注の説明**
注意	**：気をつけて読んでいただきたい内容**
備考	**：本文中の補足説明**
数の表記	**：2 進数 …** ××××**または** 0b××××
	10 進数 … ××××
	16 進数 … 0x××××

表現形式と対応レジスタ

表現形式	対応レジスタ
ro，ro’，ro”	R0-R7
rl，rl’	R0L-R7L
rh，rh’	R0H-R7H
re	R0E-R7E
reh	R0EH-R7EH
dp	DP0-DP7
dn	DN0-DN7
dm	DMX，DMY
dpx	DP0-DP3
dpy	DP4-DP7
dpx_mod	DPn，DPn++，DPn--，DPn##，DPn%%，!DPn##(**n**=0-3)
dpy_mod	DPn，DPn++，DPn--，DPn##，DPn%%，!DPn##(**n**=4-7)
dp_imm	DPn##imm(n=0-7)
＊××××	**×××をアドレスとするメモリの内容** **例** DP0 **レジスタの内容が** 1000 **のとき，***DP0 **はメモリの** 1000 **番地の内容を表します。**

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

μPD77016 ファミリに関する資料

資料名 / 品名	パンフレット	データ・シート	ユーザーズ・マニュアル		アプリケーション・ノート	
			アーキテクチャ編	命令編	基本ソフトウエア編	ライブラリ編
μPD77016	−	U10891J	U10503J	U13116J	U11958J	このマニュアル
μPD77018A		U11849J				
μPD77019						
μPD77110	U12395J	U12801J	U14623J			
μPD77111						
μPD77112						
μPD77113A		U14373J				
μPD77114						
μPD77115		U14867J				
μPD77210		U15203J	作成予定			
μPD77213		U15398J				

開発ツールに関する資料

資 料 名		資料番号
HSM77016　ユーザーズ・マニュアル		U11602J
WB77016　ユーザーズ・マニュアル	言語編	U10078J
	操作編	U11506J
ID77016　ユーザーズ・マニュアル		U10118J
CC77016　ユーザーズ・マニュアル		U15037J
RX77016　ユーザーズ・マニュアル	機能編	U14397J
	コンフィギュレーション・ツール編	U14404J
RX77016　アプリケーション・ノート	HOST API 編	U14371J

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

目　次

第1章　概　要

1.1　アプリケーション・ノートの構成

このアプリケーション・ノートで解説しているμ PD77016 ファミリ用固定小数点ライブラリは，C 言語で使用でき，標準 C ライブラリに見られる多くの浮動小数点関数に相当する固定小数点関数を用意しています。また，これらの関数は規定のレジスタに引数として値を設定することでアセンブラの使用も可能になっています。

μ PD77016 ファミリ固有の機能を使用しているこれらのルーチンは，ほかの類似したルーチンに比べ，より速い実行時間を実現しています。

このアプリケーション・ノートでは，次の 2 つのグループに分けて説明します。

・一次関数群

int	exponent(accum x)		
fixed	fxabs(fixed x);	long fixed	lfxabs(long fixed x);
fixed	_fxabs(fixed x);	long fixed	_lfxabs(long fixed x);
fixed	fxacos(fixed x);	long fixed	lfxacos(long fixed x);
fixed	fxasin(fixed x);	long fixed	lfxasin(long fixed x);
fixed	fxatan(accum x);	long fixed	lfxatan(accum x);
fixed	fxatan2(fixed x, fixed y);	long fixed	lfxatan2(long fixed x, long fixed y);
fixed	fxcos(fixed x);	long fixed	lfxcos(long fixed x);
fixed	fxdiv(fixed x, fixed y);	long fixed	lfxdiv(long fixed x, long fixed y);
accum	fxlog(accum x);		
accum	fxlog10(accum x);		
fixed	fxlog2(fixed x);	long fixed	lfxlog2(long fixed x);
fixed	fxmod(fixed x, fixed y);	long fixed	lfxmod(long fixed x, long fixed y);
fixed	fxsin(fixed x);	long fixed	lfxsin(long fixed x);
fixed	fxsqrt(fixed x);	long fixed	lfxsqrt(long fixed x);
accum	fxtan(fixed x);	accum	lfxtan(long fixed x);
accum	normalize(accum x, int y);		

・**二次関数群**:

accum	fxcosh(fixed x)	accum	lfxcosh(accum x)
accum	fxexp(fixed x)	accum	lfxexp(accum x)
accum	fxldexp(fixed x, int exp)	accum	lfxldexp(long fixed x, int exp)
accum	fxpow(fixed x, fixed y)	accum	lfxpow(long fixed x, long fixed y)
fixed	fxrand(void)	long fixed	lfxrand(void)
fixed	fxrexp(accum x, int *eptr)	long fixed	lfxrexp(accum x, int *eptr)
accum	fxsinh(fixed x)	accum	lfxsinh(accum x)
void	fxsrand(fixed seed)	void	lfxsrand(long fixed seed)
fixed	fxtanh(fixed x)	long fixed	lfxtanh(accum x)

このアプリケーション・ノートでは，関数の使用に際して必要な情報を含めて，関数ごとに解説しています。各ページでは引数や返却値といった書式，演算内容，引数の制限について解説しています。

各関数では項目プログラマーズ・ノートを記載し，その中で使用アルゴリズムの詳細，処理速度の結果，演算値の精度，ソースの変更による処理速度と演算精度の相関について記述しています。

1.2 ライブラリのインストール

ライブラリは，自己展開する実行ファイル SPXLIB5.EXE **で提供しています。**

インストール時に，ライブラリを展開するディレクトリに移動し，このファイルを実行します。実行ファイルを現在のディレクトリに置く必要はありません。

インストール例として，インストール元を A **ドライブ（**SPXLIB5.EXE **のあるディスク，ディレクトリ），インストール先をディレクトリ** C:¥DSPTOOLS¥SPXLIB **とします。**

```
?:¥?>C:                                ← C ドライブに移動
C:¥>CD ¥DSPTOOLS¥SPXLIB                ← ディレクトリ C:¥DSPTOOLS¥SPXLIB に移動
C:¥DSPTOOLS¥SPXLIB>A:SPXLIB5.EXE ← ファイル SPXLIB5.EXE の実行
```

この結果，自己解凍を行い，INC**，**LIB**，**SRC**，**TEST **などのサブディレクトリにライブラリを出力します。**

インストール先は，ディレクトリ C:¥DSPTOOLS¥SPXLIB **をお勧めします。**

これは，ライブラリの再構築時に使用する，ワークベンチ（WB77016**）用プロジェクト・ファイル（**LIB¥FXLIB.PRJ**）にファイルの絶対パスが格納されているためです。これにより，プロジェクト・ファイルを変更することなくライブラリの再構築を行うことができます。**

★ SP77016 **に付属しているライブラリのインストールについては，**SP77016 **に添付されているドキュメントを参照してください。**

1.3 ライブラリの構成

- **μPD77016 ファミリ用ライブラリの構成**

.¥DOC¥SPXLIB5.DOC	：このアプリケーション・ノート
.¥DOC¥README.TXT	：アスキー形式のドキュメント
.¥INC¥FXLIB.H	：C 言語用ヘッダ・ファイル
.¥INC¥FXLIB.INC	：アセンブリ言語用ヘッダ・ファイル
.¥INC¥FXMACRO.INC	：ライブラリ関数が使用するアセンブラ言語マクロ
.¥LIB¥FXLIB.LIB	：すべての関数で使用するリロケータブル・モジュールを含むライブラリ
.¥SRC¥*.ASM	：ライブラリ関数のソース・ファイル
.¥SRC¥FXLIB.PRJ	：すべてのライブラリ・ファイルの WB77016 用プロジェクト・ファイル
.¥TEST¥TEST1.C	：関数 fxsqrt で使用するテスト・サンプルのソース・ファイル
.¥TEST¥TEST1.ASM	：C コンパイラで生成したアセンブラ・ファイル
.¥TEST¥TEST1.PRJ	：テスト・プログラムのための WB77016 用プロジェクト・ファイル
.¥TEST¥TEST1.LNK	：リンク・ファイル
.¥TEST¥TEST1.TMG	：ハイスピード・シミュレータ（HSM77016）用タイミング・ファイル
.¥TEST¥TEST1OUT.DAT	：HSM77016 のタイミング・ファイルにより生成された出力データ・ファイル

- 各関数の検査では，すべての引数を与えることはせず，－1.0，0，＋1.0 に近似している領域，関数の機能の変化点などの特別な条件に留意し，引数の値を決めました。

各関数の演算精度は，返却される値の型に対し，±0.5LSB 以内を条件としました。これは fixed 型で$\pm 1.53\times 10^{-5}$，long fixed 型および accum 型では$\pm 2.33\times 10^{-10}$となります。関数の作成では，返却値が fixed 型では処理速度に，long fixed 型および accum 型では演算精度に重点を置いて作成しました。関数の処理速度と演算精度の相関については，各関数の項目**プログラマーズ・ノート**で説明しています。

1.4 演算速度と精度

次に示すデータは，解析のために出力した外部ファイルとともに，HSM77016 のシミュレータ機能で実行し，得られたデータを記載しています。

演算速度は，関数の呼び出しを開始点として，関数からの戻りまでを計測しているため，サブルーチン呼び出し，および復帰のための 4 サイクルが含まれています。また，関数に設定されるべきオプションを設定しているため，エラー・チェックや符号拡張のためのサイクル数も含まれています。

次に示した最大誤差は絶対値の最大値で，平均誤差は絶対値の平均で，中間誤差は誤差値の中間値を表しています。

演算速度，精度の詳細については**第 3 章　固定小数点演算ライブラリ**を参照してください。

関　数	演　算　速　度 (クロック数)	精　度		
		最大誤差(絶対値)	平均誤差(絶対値)	中間誤差
exponent(x)	7(－1.0≦x＜1.0) 11(**上記範囲外**)	0	0	0
fxabs(x)	8	3.052 E-5(x＝－1.0 **時**) **そのほかは** 0	0＋	0＋
_fxabs(x)	4	3.052 E-5(x＝－1.0 **時**) **そのほかは** 0	0＋	0＋
fxacos(x)	118(0≦|x|＜0.707) 161(0.707≦|x|＜1.0)	3.135 E-5	9.864 E-6	-7.129 E-6
fxasin(x)	109(0≦|x|＜0.707) 152(0.707≦|x|＜1.0)	3.721 E-5	9.291 E-6	4.082 E-7
fxatan(x)	33(0≦|x|＜1) 59(|x|≧1)	3.668 E-5	5.972 E-6	-7.411 E-11
fxatan2(x, y)	73(|x|＜|y|) 70(|x|≧|y|) 12(x＝y＝0)	2.617 E-5 (x=0, y＜0 : 3.052 E-5)	8.037 E-6	1.186 E-7
fxcos(x)	17	4.496 E-5	1.170 E-5	-1.742 E-6
fxcosh(x)	13	1.862 E-5	4.398 E-6	7.316 E-7
fxdiv(x, y, *remain)	48	0	0	0
fxexp(x)	23	2.027 E-5	4.642 E-6	1.202 E-6
fxldexp(x, exp)	9(x＝0) 20(x≠0, exp＜0) 19(x≠0, exp≧0) 21(**オーバフローあり**)	2.328 E-10	3.706 E-11	0
fxlog(x)	265(0.5≦x＜1) 265 - 335 (x＜0.5) 285 - 299 (x＞1)	8.583 E-10	2.019 E-10	8.953 E-11
fxlog10(x)	271(0.5≦x＜1) 271 - 341 (x＜0.5) 291 - 305 (x＞1)	5.626 E-10	1.659 E-10	5.757 E-11
fxlog2(x)	30	2.081 E-5	7.765 E-6	1.053 E-6
fxmod(x, y)	52(max.) 42(typ.)	0	0	0

関 数	演 算 速 度 (クロック数)	精 度		
		最大誤差(絶対値)	平均誤差(絶対値)	中間誤差
fxpow(x, y)	12(x＜0) 14(x = 0) 72-79(**オーバフローなし**) 81 (**オーバフローあり**)	4.855 E-4	1.522 E-5	9.691 E-6
fxrand()	15	－	－	－
fxrexp(x, eptr)	14	3.052 E-5	7.599 E-6	-1.239 E-7
fxsin(x)	20	5.246 E-5	1.147 E-5	9.725 E-6
fxsinh(x)	14	1.348 E-5	3.748 E-6	5.400 E-6
fxsqrt(x)	39	1.614 E-5	7.598 E-6	-1.023 E-7
fxsrand	10-12	－	－	－
fxtan(x)	74(0≦|x| <0.5) 102(0.5≦|x|＜0.9975) 69(0.9975≦|x|＜1)	1.029(x＞0.5) 7.399 E-5(x＜0.5)	0.004(x＞0.5) 1.337 E-5(x ＜ 0.5)	0
fxtanh	18	2.812 E-5	7.993 E-6	4.070 E-7
lfxabs(x)	8	4.657 E-10(x＝－1.0 **時**)	0＋	0＋
_lfxabs(x)	4	4.657 E-10(x＝－1.0 **時**)	0＋	0＋
lfxacos(x)	126(min.) 259 - 334 (typ.) 407(max.)	1.344 E-8**(すべての誤差のうち** 99%**は約** 6E-10 **を下回っています)**	1.653 E-10	1.337 E-12
lfxasin(x)	117(min.) 250 - 325 (typ.) 398(max.)	1.344 E-8**(すべての誤差のうち** 99%**は約** 6E-10 **を下回っています)**	1.653 E-10	1.337 E-12
lfxatan(x)	103(0≦|x|＜0.5) 177(0.5≦|x|≦1) 175(1＜|x|≦2) 178(2＜|x|≦4) 181(|x|＞4)	7.541 E-10	1.454 E-10	0
lfxatan2(x, y)	198(0≦|x|/|y|＜0.5) 201(0.5≦|x|/|y|≦1) 197(1＜|x|/|y|≦2) 194 (2≦|x|/|y|) 12(x＝y＝0)	5.211 E-10	1.811 E-10	-3.708 E-12
lfxcos(x)	93	7.072 E-10	2.048 E-10	-5.436 E-11
lfxcosh(x)	13(x＝－256) 109(6.238≦|x|＜256) 261(1.693≦|x|＜6.238) 228(0.693≦|x|＜1.693) 254(0≦|x|＜0.693)	－ － 4.264 E-8 － 5.800 E-10	－ － 2.812 E-9 － 1.383 E-10	－ － 3 E-10 － 2.772 E-11
lfxdiv(x, y)	77	＜2.33 E-10	－	－
lfxexp(x)	11(x＜-22) 167(-22≦x＜0) 141(0≦x＜1) 174(1≦x＜5.545) 17(x＞5.545)	－ － 4.968 E-10 3.613 E-8	－ － 1.309 E-10 3.314 E-9	－ － -7.151 E-12 －

関 数	演 算 速 度 (クロック数)	精 度		
		最大誤差(絶対値)	平均誤差(絶対値)	中間誤差
lfxldexp(x, exp)	9(x＝0) 20(x≠0, exp＜0) 19(x≠0, exp≧0) 21(**オーバフローあり**)	2.328 E-10	5.349 E-11	0
lfxlog2(x)	235	3.982 E-10	1.280 E-10	5.879 E-11
lfxmod(x)	68(max.) 55(typ.)	0	0	0
lfxpow(x, y)	12(x＜0 **エラー時**) 14(x＝0) 444-547 (**オーバフローあり**) 346-390 (**オーバフローなし**)	4.522 E-8	2.954 E-10	-2.353 E-13
lfxrand()	23	－	－	－
lfxrexp(x, eptr)	13(－1≦x＜1) 17(x＜－1, 1≦x)	2.328 E-10	3.424 E-12	0
lfxsin(x)	109	4.685 E-10	1.359 E-10	2.516 E-11
lfxsinh(x)	15(x＝-256) 37(6.238≦|x|＜256) 265(1.693≦|x|＜6.238) 232(0.693≦|x|＜1.693) 258(0≦|x|＜0.693)	8 E-8(all x) 8.088 E-10(0≦|x|＜1)	2 E-9(all x) 2.120 E-10(0≦|x|＜1)	9 E-11 (all x) － 4.801 E-12(0≦|x|＜1)
lfxsqrt(x)	60	2.327 E-10	1.149 E-10	7.447 E-11
lfxsrand(seed)	13	－	－	－
lfxtan(x)	285(0≦|x|＜0.5) 307(0.5≦|x|＜0.9975) 238(0.9975≦|x|＜1)	9.870 E-10 (|x|＜0.5) 2.843 E-5 (|x|＞0.5)	2.098 E-10(|x|＜0.5) 8.827 E-8(|x|＞0.5)	0
lfxtanh(x)	42(12.47＜|x|)	－	－	－
	311-362(0＜|x|＜12.47)	6.806 E-10	1.616 E-10	1.911 E-11
normalize(x, y)	8(y≧0) 11(y＜0)	0	0	0

第２章　供給ファイルの内容

2.1　ファイルの構成

2.1.1　ライブラリ・ファイル“ FXLIB.LIB ”

ディレクトリ.¥LIB の，ファイル FXLIB.LIB には，リロケータブルなオブジェクトとして各関数が入っています。このライブラリ・ファイルは，WB77016 上で各関数をアセンブルしたあと，WB77016 とともに提供されている LB77016 を使用し，作成されています。

2.1.2　プロジェクト・ファイル“ FXLIB.PRJ ”

ディレクトリ.¥SRC の，ファイル FXLIB.PRJ には，オブジェクト・ファイルの更新に必要となる情報が入っています。このプロジェクト・ファイルは，ライブラリを構成するソース・ファイルやインクルード・ファイルを変更したあと，WB77016 の MAKE コマンド実行するときに使用します。

ライブラリの作成では，MAKE コマンド実行後，LB77016 を使用し，ライブラリ FXLIB.LIB に追加する必要があります。

2.1.3　ソース・ファイル“ ＊.ASM ”

各関数は，関数名をファイルに持つソース・ファイルに分割し，ディレクトリ.¥SRC に格納しています。関数の変更がある場合，これらソース・ファイルを変更し，WB77016 のアセンブル機能によりオブジェクト化します。

2.1.4　インクルード・ファイル“ FXLIB.H ”

ディレクトリ.¥INC の，ファイル FXLIB.H は，C 言語用のインクルード・ファイルです。

このファイルには，関数のプロトタイプ，グローバル・ライブラリ変数のための外部宣言，エラー状態を示すシンボル定義値が記述されています。

ファイル FXLIB.INC 内に定義されている FX_ERRCK, FX_SIGNEXT, FXABS_CLIP, LFXABS_CLIP は同じく本ファイル内でも定義されています。

2.1.5　インクルード・ファイル“ FXLIB.INC ”

ディレクトリ.¥INC の，ファイル FXLIB.INC は，アセンブリ言語用のインクルード・ファイルです。このファイルには，条件付きアセンブルや，エラー状態を示すシンボル定義値が定義されています。

2.1.6　インクルード・ファイル“ FXMACRO.INC ”

ディレクトリ.¥INC の，ファイル FXMACRO.INC は，アセンブリ言語用のインクルード・ファイルです。このファイルには，ライブラリ関数で使用するマクロ定義が含まれています。

2.1.7　領域確保“ FXERRNO.ASM ”

ディレクトリ.¥SRC の,ファイル FXERRNO.ASM では,グローバル変数_fx_errno と_fx_signext の領域確保を行っています。変数_fx_errno は，エラー・チェックを有効にした状態でライブラリを構築したときの，ライブラリの状態を保存するのに使われます。変数_fx_signext は，符号拡張を有効化した状態で，テンポラリとして使用されます。

2.2　アセンブル条件

ディレクトリ.¥INC の，ファイル FXLIB.H および，ファイル FXLIB.INC には，ソース・ファイルのアセンブル条件を決めるいくつかのシンボルが記述されています。

2.2.1　エラー・チェック“ FX_ERRCK ”

FX_ERRCK は，関数のエラー（引数エラーおよび演算エラー）のチェックの有無を定めます。

FX_ERRCK が次のように定義されている場合，引数や演算により変数_fx_errno のセット／クリアが行われます。エラーがある場合，関数の返却値は値 0 になります。

```
#define   FX_ERRCK
```

FX_ERRCK が次のようにコメント・アウトされている場合，引数のエラー・チェック，変数_fx_errno のセットおよびリセットは行われません。これにより，関数の処理は速くなります。引数が正しいことが事前にわかっている場合に，このように設定します。

```
/* #define   FX_ERRCK */
```

たとえば，関数 fxdiv は，引数として除数と被除数を与え，商と剰余を求める関数です。戻り値が 0 の場合，除算でのエラーを表します。また，被除数が除数と同等あるいは大きい値でも，返却値（商）が 1.0 以上の（16 ビットを越える）ため，エラーとなります。

FX_ERRCK が有効時，変数_fx_errno に設定されるシンボル名は，ファイル FXLIB.INI，FXLIB.H で，次のように設定されています。

```
#define   FX_ERR_OK       0       /* no error */
#define   FX_ERR_DIV0     1       /* divide by zero */
#define   FX_ERR_ARGS     2       /* argument error */
#define   FX_ERR_RANGE    3       /* range error */
```

現行ファイル FXLIB.H および FXLIB.INI の FX_ERRCK は，エラー・チェックを行うようになっています。

2.2.2　符号拡張“ FX_SIGNEXT ”

FX_SIGNEXT は，関数に渡された引数の符号拡張の有無を定めます。

FX_SIGNEXT が次のように定義されている場合，引数の符号拡張を行う命令を関数に追加します。この命令はレジスタの L パート（R0L-R7L）に値 0 を，E パート（R0E-R7E）に H パート（R0H-R7H）の符号拡張した値を設定します。これにより，コード・サイズ，実行時間ともに増えることになります。

```
#define FX_SIGNEXT
```

次のようにコメント・アウトしている場合，符号拡張の命令は追加されません。

```
/* #define FX_SIGNEXT */
```

関数への引数として渡される 16 ビット固定小数値は，関数 fxdiv の引数（被除数）がレジスタ R0H を使用していることでもわかるように，レジスタ R0, R1, R2, R3 のビット 16 - ビット 31 を使用しています。そのほかのレジスタ R0L-R7L（値 0 を設定）や R0E-R7E（R0H-R7H の符号を設定）は，ロード時のレジスタ R0H-R7H に依存します

（詳しくはμPD77016 ファミリ　ユーザーズ・マニュアル　命令編を参照）。

次の記述では，R0H に指定メモリの値をロードし，R0L に値 0 が，R0E には R0H の符号（00 あるいは 0xFF）が設定されます。

```
r0=*dp0;
   :
call _fxdiv;
```

しかし，次のように R0H に設定した場合，メモリの値を R0H にロードし，R0L および R0E は変化しません。

```
r0h=*dp0;
   :
call _fxdiv;
```

このように R0L に値 0 が，R0E に符号が設定されない場合，関数は，引数を 32 ビット固定小数値(long fixed 型)とみなし，符号の情報は欠落したしたまま演算を行ってしまうため，不正な結果になってしまいます。

注意　関数が引数として 40 ビット固定小数値を要求しない場合，命令の追加は不要になりますが，C コンパイラを使用する場合は，必ず定義してください。

現行ファイル FXLIB.H および FXLIB.INI の FX_SIGNEXT は，引数の符号拡張を行うよう定義されています。

2.2.3　クリッピング“ FXABS_CLIP ”

FXABS_CLIP は関数 fxabs,マクロ_fxabs のみに使用され，クリッピング処理の有無を定めます。

関数 fxabs は，引数の絶対値を得る ABS 命令を使用して作成しています。この関数は，引数－1.0 に対し fixed 型であるため返却値 1.0 を表現できません。

そのため，FXABS_CLIP を定義することにより，0x8000 を 0x7FFF に変換する ROUND 命令を関数に加え，fixed 型 1.0 に近似させます。

ファイル FXLIB.H 中に#define 文で FXABS_CLIP が定義されており，これにより，マクロ_fxabs に命令が追加されます。

2.2.4　クリッピング“ LFXABS_CLIP ”

LFXABS_CLIP は関数 lfxabs,マクロ_lfxabs のみに使用され，クリッピング処理の有無を定めます。

LFXABS_CLIP の定義により，0x80000000 を 0x7FFFFFFF に変換する CLIP 命令を関数に加え，long fixed 型 1.0 に近似させます。ファイル FXLIB.H 中に#define 文で LFXABS_CLIP が定義されており，これにより，マクロ_lfxabs に命令が追加されます。

通常，満数に対し命令生成の制御オプションはそのソース・ファイル内で行われます。しかし，マクロ_fxabs や _ifxabs は，ファイル FXLIB.H で定義されているため，FXABS_CLIP や LFXABS_CLIP もまた。ヘッダ・ファイルで定義する必要があります。

第３章　固定小数点演算ライブラリ

3.1　記述フォーマット

各関数を次のフォーマットに従い説明します。

[書　式]

関数の使用にあたり必要となるインクルード・ファイルと，その関数の型を記述します。

[機　能]

関数の機能について説明します。

[制　限]

関数の使用，引数，返却値についての制限を説明します。

[プログラマーズ・ノート]

関数の演算にあたり，使用するレジスタの一覧や，使用するアルゴリズムについて詳細に説明します。

・アセンブル条件

関数を使用する際の付加条件と，それによる動作を説明します。

・実行サイクル数

関数の実行サイクル数を説明します。この実行サイクルには分岐命令（CALL 命令，RET 命令）の 4 サイクルが含まれます。

・誤差

演算結果の誤差特性を記述しています。誤差の表記では最大，平均，中間の 3 つを記述します。

ここでの最大は誤差を絶対値で表した最大値，平均は誤差の平均値，中間は誤差の中間値で記述し，特に記述がないかぎり，引数で定義されている型の範囲になります。

また，0.25LSB ごとに分割した誤差の度数分布を表す誤差分布の概要も記載している場合もあります。

この場合，返却値に fixed 型を返す関数では，1LSB が 2^{-15}，0.25LSB では 2^{-17}，つまり 7.63 E-6 になり，返却値に long fixed 型または accum 型を返す関数では，1LSB が 2^{-13}，0.25LSB では 2^{-33}，つまり 1.16 E-10 となります。

・備考

処理速度，性能，機能について改善点がある場合，それらプログラムの変更の内容を説明します。

3.2　exponent

[書　式]

#include <fxlib.h>

int　exponent(accum x);

[機　能]

引数 x **を正規化するのに必要なシフト量を求める関数です。**

関数内部では，引数が正のとき 0.5 - 0.999...,**負のとき**－1.0～－0.5 **の範囲になるまでビット・シフトを行います。**

引数が－1≦x＜1 **の場合，返却値は** EXP**（指数）命令と同じ結果を返します。引数が** accum **型を越えると** EXP**（指数）命令は不定となり，その結果返却値に負の値を返します。**

引数と返却値の関係を次に示します。

例

引数 x=0x0020000000(0.25)	**返却値**=1
引数 x=0x0000002000	**返却値**=11h（17）
引数 x=0x0400000000(8.0)	**返却値**=－4
引数 x=0xFC00000000(－8.0)**注1**	**返却値**=－3 **注2**
引数 x=0	**返却値**=0

注 1．この場合，－1 **になった時点でシフト・ダウンを終了します。**

2．引数が正と負では，返却値が違いますので注意してください。

この関数の返却値（指数）の符号は，関数 fxrexp **や関数** lfxrexp **の結果と反対になりますので注意してください。**

[制　限]

引数は accum **型を設定します。**

[プログラマーズ・ノート]

引数　　**：**R0

返却値　　**：**R0H

使用レジスタ　**：**R0

呼び出し関数　**：なし**

関数内部では，引数が－1≦x＜1 **ではそのままの値を，**x＜－1, 1≦x **では** 8 **ビット・シフトして適合させた値を，**EXP**（指数）命令に渡す処理を行います。**

・アセンブル条件

FX_SIGNEXT **および，**FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
7	−1≦x<1
11	上記範囲外

・誤差

誤差はありません。

・備考

マクロ定義を行うことで，CALL 命令，RET 命令の 4 サイクルを省くことができます。マクロはユーザが定義してください。

3.3　fxabs/_fxabs

[書　式]

```
#include <fxlib.h>
fixed    fxabs(fixed x);
fixed    _fxabs(fixed x);
```

[機　能]

引数 x の絶対値を求める関数です。

引数が 0≦x はそのままの値を，x<0 は負を積算した値を返します。

絶対値の演算では関数 fxabs(fixed x)と，マクロ_fxabs(fixed x)の 2 つを用意しています。

[制　限]

引数−1.0 に対し，返却値 1.0 を表現できません。そのため，RND（丸め）命令によるクリッピングを行う処理が含まれています。このクリッピングは FXABS_CLIP の定義により実行されます。

[プログラマーズ・ノート]

引数　　：R0H

返却値　　：R0H

使用レジスタ　：R0

呼び出し関数　：なし

この関数は，サブルーチンとマクロの 2 つを用意しています。これらは同等の動作を行います。

関数内部では，ABS（絶対値）命令を使用しています。

・アセンブル条件

FXABS_CLIP を定義すると，返却値のクリッピング処理を行う命令が追加されます。

FX_SIGNEXT を定義すると，引数の符号拡張を行う 2 命令が追加されます。

これにより，R0E には符号拡張された値が，R0L には値 0 が設定されるようになります。

FX_SIGNEXT を無効にした場合は，呼び出し時に R0E に R0H の符号拡張を行った値を設定してください。たとえば，レジスタ R0E に 0xFF，レジスタ R0H に 0x0001 を設定することで，0xFFFF が返されます。

FX_ERRCK の定義は必要ありません。

・実行サイクル数

ルーチン	サイクル数	引　数
関数 fxabs	8	FXABS_CLIP 定義（ディフォールト）
	7	FXABS_CLIP 未定義
マクロ_fxabs	4	FXABS_CLIP 定義（ディフォールト）
	3	FXABS_CLIP 未定義

・誤差

	誤差		備考
	クリッピングあり	クリッピングなし	
最大	3.052E-5	2	引数－1.0 の場合
平均	0 注	0 注	－
中間	0	0	－

注　引数－1.0 のときの誤差を除くと，最大，平均，中間誤差はほぼすべて 0 になります。

3.4　fxacos

[書　式]

```
#include <fxlib.h>
fixed    fxacos(fixed x);
```

[機　能]

引数 x の逆余弦を求める関数です。

通常，逆余弦は 0-πラジアンになりますが，これは fixed 型の範囲を外れるため，この関数では 0≦返却値＜1.0 の範囲にして返します。このため弧度は，返却値とπの積算で求めます。

[制　限]

引数は，fixed 型（0x8000(－1.0)- 0x7FFF(0.9999...)の範囲）のため，値 1.0 は与えられません。

また，返却値も同様に値 1.0 を返せませんので注意してください。

たとえば，値－1.0 の逆余弦は値πになりますが，この関数では値 1.0 を返すところを 0x7FFF(0.999...)としています。このため引数や返却値のチェックを行う必要があります。

[プログラマーズ・ノート]

引数　　：R0H

返却値　　：R0H

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：関数 fxasin　（関数 fxasin の内部では，関数 fxatan を使用しています）

関数内部では，関数 fxasin を呼び出し，次の演算を行います。

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x)$$

・アセンブル条件

FX_SIGNEXT **の定義は，関数** fxasin **で行うため，この関数での定義は必要ありません。**

関数 fxasin **で定義してください。**

FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
16	0
18	－1
118	0＜|x|＜0.707
161	0.707≦|x|＜1.0

・誤差

	誤差
最大	3.667E-5
平均	1.006E-5
中間	－7.129E-6

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	44.51	74.71	94.81	99.86	100	－

3.5　fxasin

[書　式]

#include <fxlib.h>

fixed　fxasin(fixed x);

[機　能]

引数 x **の逆正弦を求める関数です。**

通常，逆正弦は－π/2～π/2 **ラジアンになりますが，これは** fixed **型の範囲を外れるため，この関数では**－1.0≦**返却値＜**1.0 **の範囲にして返します。このため弧度は，返却値と**π/2(1.5707...)**の積算で求めます。**

[制　限]

引数は fixed **型（**0x8000(－1.0) - 0x7FFF(0.9999...)**の範囲）のため，値** 1.0 **は与えられません。**

また，返却値も同様に値 1.0 **を返せませんので注意してください。**

たとえば，値 1.0 **の逆正弦は値**π/2 **となりますが，この関数では値** 1.0 **を返すところを** 0x7FFF(0.999...)**としています。このため引数や返却値のチェックを行う必要があります。**

[プログラマーズ・ノート]

引数　　：R0H

返却値　　：R0H

使用レジスタ　：R0,R1, R2, R3, R4, DP1

呼び出し関数　：fxatan

関数内部では，関数 fxatan **や関数** fxsqrt **の一部ルーチンを呼び出し，次の演算を行います。**

$$\arcsin(x) = \arctan\left(\frac{x}{sqrt(1-x^2)}\right)$$

ただし，引数が 1.0 **に近似すると誤差が生じるため，**0.707≦x＜1 **の範囲では，次の式**

$$\arcsin(x) = \frac{\pi}{4} + \arcsin\left(\frac{x - sqrt(1-x^2)}{sqrt(2)}\right)$$

を置換した次の演算を行います。

$$\arcsin(x) = \frac{\pi}{4} + \arctan\left(\frac{x - sqrt(1-x^2)}{sqrt(1+2\times x\times sqrt(1-x^2))}\right)$$

・アセンブル条件

FX_SIGNEXT **を定義すると，引数の符号拡張を行う** 2 **命令が追加されます。**

FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
7	0
8	－1
109	0≦|x|＜0.707
152	0.707≦|x|＜1.0

・誤差

	誤差
最大	4.283E-5
平均	9.681E-6
中間	4.082E-7

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	45.55	79.02	95.23	99.47	99.98	100

3.6　fxatan

[書　式]

#include <fxlib.h>

fixed　fxatan(accum x);

[機　能]

引数 x **の逆正接を求める関数です。**

通常，逆正接は－π/2～π/2 **ラジアンになりますが，これは** fixed **型の範囲を外れるため，この関数では，**

－1.0≦**返却値**＜1.0 **の範囲にして返します。このため弧度は，返却値と**π/2(1.5707...)**の積算で求めます。**

[制　限]

引数は accum **型を設定します。**

返却値は，±π/2 **に相当する**±1.0 **を返せませんので注意してください。**

この場合，返却値は 1.567**（約**±89.7747 **度）に相当する約**±0.9975 **を返します。**

[プログラマーズ・ノート]

引数　　**：**R0

返却値　　**：**R0H

使用レジスタ　**：**R0, R1,R2, R3, R4, DP1

呼び出し関数　**：なし**

関数内部では，引数が 0≦|x|＜1 **の場合は** arctan(x)**の演算を，**|x|≧1 **の場合は次の演算を行います。**

$$\arctan(x) = \frac{\pi}{2} - \arctan\left(\frac{1}{x}\right)$$

・アセンブル条件

FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
33	0<|x|<1
59	1≦|x|

・誤差

	誤差
最大	3.668E-5
平均	5.972E-6
中間	－7.411E-11

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	67.67	88.72	98.09	99.8	100	100

3.7　fxatan2

[書　式]

#include <fxlib.h>

fixed　fxatan2(fixed x, fixed y);

[機　能]

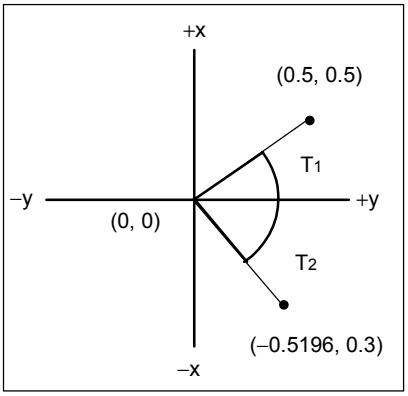
引数 x/y **の逆正接を求める関数です。**

この関数は，引数 x, y **をそれぞれ平面の垂直軸，水平軸の座標（**x, y**）とし，＋側の** y **軸線と，原点（**0, 0**），座標（**x,y**）を結ぶ線の成す角をラジアンとする** T **を返却値として返します。**

通常，逆正接は－π**～＋**π**の値をとりますが，これは** fixed **型の範囲をはずれるため，この関数では－**1.0**≦返却値＜**1.0 **の範囲にして返します。このため弧度は，返却値と**π**の積算で求めます。**

たとえば，引数 x=0.5, y=0.5 **を設定すると，返却値に** 0.25 **を返します。これに**π**を積算して得られた値**π/4 **が角** T1 **となります。**

引数 x=－0.5196, y=0.3 **を設定すると，返却値に－**0.333...**を返します。これに**π**を積算して得られた値－**π/3 **となり，この結果角** T_2 **にあたり，＋側の** y **軸線と原点（**0, 0**），座標（－**0.5196, 0.3**）を結ぶ線の作る角と同一になります。**



この関数は,引数 x=0, y≧0 **の場合,値** 0 **を返し,引数** x=0, y＜0 **の場合,値** 1.0(π)**ではなく** 0x7FFF(0.999....) **を返します。**

また，引数 x≧0, y=0 **の場合値，**0.5(π/2)**を，引数** x＜0, y=0 **の場合，値－**0.5(－π/2)**を返します。**

[制　限]

引数 x **および** y **には値** 0 **を設定しないでください。**

引数 x=0, y=0 **を設定した場合，**FX_ERRCK **が定義されていないと，返却値は不定になります。**

[プログラマーズ・ノート]

引数　　：R0H（**垂直軸**:x）
　　R1H（**水平軸**:y）

返却値　　：R0（1/π×arctan(x/y)）

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：なし

この関数では引数は絶対値で演算し，その座標を固定象限上に置きます。内部の演算では引数が x＜y **の場合は** arctan(x/y)**を，**x≧y **の場合は次の演算を行います。**

$$\arctan(x) = \frac{\pi}{2} - \arctan\left(\frac{1}{x}\right)$$

・アセンブル条件

FX_SIGNEXT **を定義すると，引数の符号拡張を行う** 2 **命令が追加されます。**

FX_ERRCK **を定義すると，関数のエラー・チェックをする命令が追加されます。**

たとえば引数 x=0, y=0 **の場合では，**_fx_errno **に** FX_ERR_DIV0 **を設定し，返却値に値** 0 **を返すようになります。この場合実行時間に** 6 **サイクルが加わります。**

・実行サイクル数

サイクル数	引数
12	x=y=0
73	|x| < |y|
70	|y|≦|x|

・誤差

	誤差
最大	3.052E-5 **注**
平均	8.037E-6
中間	1.186E-7

注　引数 x=0,y <0 **が設定された場合は** 2.617 E-5 **となります。**

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	50.13	90.66	99.87	99.96	100	ー

3.8　fxcos

[書　式]

#include <fxlib.h>

fixed　fxcos(fixed x);

[機　能]

引数 x **の余弦を求める関数です。**

返却値は cos(π/2×x)**を返します。引数**－π/2～＋π/2 **の範囲は，**fixed **型として表現できないため，引数に** 2/π(0.6366...)**を積算してください。これにより得られる返却値は積算する前の引数の余弦が得られます。**

[制　限]

値 0 **の余弦は**＋1.0 **ですが，**fixed **型で表現できないため，この場合，返却値には** 0x7FFF (0.9999...)**を返します。このとき最下位ビットに誤差（**2^{-15}**）が発生します。**

[プログラマーズ・ノート]

引数　　**：**R0H

返却値　　**：**R0H

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：なし

関数内部では，引数に関係なく次の演算を行います。

$$\cos(x) = 1 + c_1 x^2 + c_2 x^4 + c_3 x^6$$

- **アセンブル条件**

  FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

- **実行サイクル数**

サイクル数	引数
17	すべての引数

- **誤差**

	誤差
最大	4.496 E-5
平均	1.171 E-5
中間	－1.742 E-6

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	38.80	69.12	88.26	97.32	99.77	100

3.9　fxcosh

[書　式]

#include <fxlib.h>

accum　fxcosh(fixed x);

[機　能]

引数 x **の双曲余弦を求める関数です。**

引数 x=－1.0 **で返却値は最大となり，約** 1.543 **を返します。引数が** fixed **型を越える場合は，3.31　lfxcos を参照してください。**

[制　限]

引数は fixed **型を設定します。**

[プログラマーズ・ノート]

引数　　**：**R0H

返却値　　**：**R0

使用レジスタ　：R0, R1, R2, R3, DP1

呼び出し関数　：なし

関数内部では，引数に関係なく次の演算を行います。

$$\cosh(x) = 1 + c_1 x^2 + c_2 x^4 + c_3 x^6$$

- **アセンブル条件**

  FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

- **実行サイクル数**

サイクル数	引数
13	すべての引数

- **誤差**

	誤差
最大	1.862 E-5
平均	4.398 E-6
中間	7.316 E-7

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	85.93	99.71	100	100	100	100

3.10　fxdiv

[書　式]

#include <fxlib.h>
fixed fxdiv(fixed x, fixed y, fixed ＊remain);

[機　能]

引数 x を被除数，y を除数とする商および余りを求める関数です。
商は返却値として，余りは remain で示したアドレスに格納します。

[制　限]

引数 y には 0 を設定しないでください。
引数 x, y には|x|＜|y|の条件を満たす値を設定してください。

[プログラマーズ・ノート]

引数　　：R0H（引数 x）
　　R1H（引数 y）
　　R2H（引数*remain）
返却値　　：R0
　　ポインタ remain
使用レジスタ　：R0, R1, R2, R3, R4, DP5
呼び出し関数　：なし

関数内部では，DIV 命令（演算ごとに 1 ビットずつ求める，除算命令）を用い，16 ビット固定小数値の除算を行います。この DIV 命令を 17 回行うことで，17 ビットの精度を実現しています。
除算の余りは，被除数－商×除数で求めます。精度は fixed 型の±1/2LSB に調整されるため，32 あるいは 40 ビット固定小数値の下位になります。格納時，余りは 2^{16} まで引き上げています。

- **アセンブル条件**

  FX_SIGNEXT を定義すると，引数の符号拡張を行う 4 命令が追加されます。
  内部ではレジスタ R0, R1 の全ビットを使用するため，符号拡張は有効にしておいてください。
  FX_SIGNEXT を無効にした場合は，R0, R1 に符号拡張を行った値と同等の値を設定してください。
  FX_ERRCK を定義すると，関数のエラー・チェックを行う 14 命令が追加されます。
  これにより，エラーなしの場合にも 11 クロック多い実行時間がかかります。
  引数にエラーがある場合，返却値および remain の示すアドレスには値 0 が格納され，_fx_errno に次に示す値が設定されます。

エラー条件	値
y=0	FX_ERR_DIV0
y≦x	FX_ERR_ARGS
エラーなし	0

・実行サイクル数

サイクル数	引数
15	y=0（エラー時）
23	|y|＜|x|（エラー時）
48	その他

・誤差

固定小数値最下位±0.5LSB の誤差（$\pm 2^{-16}$,または±1.526 E-5）となります。

これは商，余りともに誤差値は同じです。

・備考

関数内部の変更では次に示す 3 点があります。かっこ内行数は，ファイル FXDIV.ASM の変更する行数を表します。

（1）余りを必要としない場合，余りに関する処理を削除し，処理速度を実行時に 7 サイクルを，エラー時に 3 サイクルを縮めることができます。この関数で，余りの処理に関する命令は次のとおりです。

- レジスタ R2H の値をレジスタ DP5 に退避する 2 命令（32, 33 行）
- 引数エラー時に，remain で示すアドレスに値 0 を格納する 1 命令（48 行）
- 余りの算出とストアに関わる 4 命令（68 行，79 行，81, 82 行）

商を求める演算では積算を使用しないため，正しい符号を得る左シフト後の右シフトといった作業が必要なくなります。これによりシフト命令で行う 2 命令を R0=R0 SLL 15 の 1 命令に置き換えることができます（74, 75 行）。

（2）商の精度がそれほど必要がない場合，次のように商の 17 ビット目の演算や結果の丸め処理を行う命令を省略します（71 - 76 行）。

これにより 2 命令を省くことができ，商の精度は±1LSB に保つことができます。また，これによる余りの精度は変化しません。

```
REP 17;
R0 /= R1;
R0 = R0 SLL 23;
R0 = R0 SRA 8;
ROUND(R0);
```

→

```
REP 16;
R0 /= R1;
R0 = R0 SLL 24;
R0 = R0 SRA 8;
```

（3）さらに，エラー・チェックに続く命令（68 - 84 行）を次のように変更します。これにより商の誤差は±1LSB になります。

```
REP 16:
R0 /= R1;
R0 = R0 SLL 16
RET;
```

さらに処理速度を上げるため，商のビット数を減らしたり，除算のループ回数を減らしたり，R0 レジスタのビット・シフト数を増すなどの処置が考えられます。

3.11　fxexp

[書　式]

#include <fxlib.h>

accum　fxexp(fixed x);

[機　能]

引数 x **を正規化するのに必要なシフト量を求める関数です。**

返却値は e^{-1}**（約** 0.368**）から** $e^{0.999...}$**（約** e=2.718...**）の範囲となります。この返却値は** 1.0 **以上の値になるため** accum **型になります。**

[制　限]

引数は fixed **型を設定します。**

[プログラマーズ・ノート]

引数　　**：**R0H

返却値　　**：**R0

使用レジスタ　：R0, R1, R2, R3, DP1

呼び出し関数　：なし

関数内部では，次の演算を行います。

$$e^x = 1 + c_1x^1 + c_2x^2 + c_3x^3 + c_4x^4 + c_5x^5$$

それぞれに設定している係数は引数 x **の符号によって違ってきます。**

- **アセンブル条件**

  FX_SIGNEXT **が定義されている場合，引数の符号拡張のため** 2 **命令が追加されます。**

  FX_ERRCK **の定義は必要ありません。**

- **実行サイクル数**

サイクル数	引数
23	すべての引数

- **誤差**

	誤差
最大	2.027 E-5
平均	4.642 E-6
中間	1.202 E-6

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	81.03	99.69	100	100	100	100

3.12　fxldexp

[書　式]

#include <fxlib.h>

accum　fxldexp(fixed x, int exp);

[機　能]

引数 x, exp **から，演算（**$x \times 2^{exp}$**）の結果を得る関数です。**

引数 exp **の値に応じた引数** x **のビット・シフトを行います。この場合，正のときは左シフト，負のときは右シフトを行います。**

返却値は 1.0 **以上の値になるため** accum **型になります。**

関数 lfxldexp **では，引数それぞれが** long fixed **型と** int **型，返却値が** accum **型となっています。詳しくは，3.35 lfxldexp を参照してください。**

[制　限]

$-63 \leqq exp \leqq 63$

$-256 \leqq x \times 2^{exp} < 256$

引数 x **は** fixed **型を設定します。**

引数 exp **は** int **型として定義されていますが，下位** 6 **ビットはシフト・カウント，最上位** 1 **ビットは符号ビットになります。**|exp|＞40 **は，返却値の全ビットが** 0 **または** 1 **になりますので注意してください。**

FX_ERRCK **を定義し，|返却値|＞**256**（正では** 0x7FFFFFFFFF(255.999...)**，負では** 0x8000000000(－256)**）になる引数を与えると，**_fx_errno **に** FX_ERR_RANGE **が設定されます。**

[プログラマーズ・ノート]

引数	**：**R0H**（引数** x**）**
	R1H**（引数** exp**）**
返却値	**：**R0
使用レジスタ	**：**R0,R1,R2,R4
呼び出し関数	**：なし**

引数 exp **の符号（正：左シフト，負：右シフト）により，引数** x **のシフトする方向が決まります。**

・アセンブル条件

FX_SIGNEXT **を定義すると，引数** x, exp **の符号拡張のため** 3 **命令が追加されます。**

FX_ERRCK **を定義すると，**_fx_errno **にエラーの状態を表す値が設定されます。**

FXLDEXP_ROUND **を定義すると，処理に** 2 **命令が追加され，右ビット・ローテートされた結果が得られます。**

FXLDEXP_ROUND **を未定義にすると，定義時に比べ，返却値の精度が劣ります。**

このラベルはソース・ファイル中に定義されています。

・実行サイクル数

サイクル数	引数
9	x = 0
20/18	x != 0, exp < 0（round/no round）
19	x != 0, 0≦exp, オーバーフローなし
21	x != 0, 0≦exp, オーバーフローあり

・誤差

	誤差	
	round	no round
最大	2.328 E-10	4.657 E-10
平均	3.706 E-11	6.318 E-11
中間	0	0

		0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	round	82.80	97.90	100	100	100	100
	no round	77.90	85.70	94.60	100	100	100

3.13　fxlog

[書　式]

```
#include <fxlib.h>
accum   fxlog(accum x);
```

[機　能]

引数 x の自然対数（対数の底を *e* とした，つまり ln）を求める関数です。

[制　限]

引数 x＞0 の範囲を設定してください。

引数 x≦0 を設定した場合，FX_ERRCK が定義されていないと，返却値は不定になります。

[プログラマーズ・ノート]

引数　　：R0

返却値　　：R0

使用レジスタ　：R0,R1,R2,R3,R4,DP1

呼び出し関数　：lfxlog2

関数内部では，関数 lfxlog2 を使用し，次の演算を行います。

$$\ln(x) = \log_2(x) \times \ln(2)$$

関数 lfxlog2 の引数には，ビット・シフトにより 0.5≦x＜1.0 の範囲にした値を与えます。関数 lfxlog2 の返却値では，引数で行ったビット・シフト回数に対応して ln(2)の加減算を行うことで値の調整をしています。

・アセンブル条件

FX_SIGNEXT の定義は必要ありません。

FX_ERRCK を定義すると，引数 x≦0 の場合，_fx_errno に FX_ERR_ARGS が設定され，返却値は 0 となります。

・実行サイクル数

サイクル数	引数
10	x≦0（エラー）
265	0.5≦x＜1
265 - 335	x＜0.5
285 - 299	x≧1

・誤差

	誤差
最大	9.251 E-10
平均	1.863 E-10
中間	5.266 E-11

	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00
最下位ビットの誤差率（%）	38.28	68.47	86.83	95.34	98.36	99.66	99.92	100

これら誤差は，実際に使用する精度と異なる場合があります。

3.14　fxlog10

[書　式]

```
#include <fxlib.h>
accum    fxlog10(accum x);
```

[機　能]

底を 10 とする引数 x の対数を求める関数です。

[制　限]

引数 x＞0 の範囲を設定してください。

引数 x≦0 を設定した場合，FX_ERRCK が定義されていないと，返却値は不定になります。

[プログラマーズ・ノート]

引数　　：R0

返却値　　：R0

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：lfxlog2

関数内部では，関数 lfxlog2 を使用し，次の演算を行います。

$$\log_{10}(x) = \log_2(x) \times \log_{10}(2)$$

関数 lfxlog2 の引数には，ビット・シフトにより 0.5≦x＜1.0 の範囲にした値を与えます。関数 lfxlog2 の返却値では，引数で行ったビット・シフト回数に対応して ln(2)の加減算を行うことで値の調整をしています。

・アセンブル条件

FX_SIGNEXT の定義は必要ありません。

FX_ERRCK を定義すると，引数 x≦0 の場合，_fx_errno に FX_ERR_ARGS が設定され，返却値は 0 となります。

・実行サイクル数

サイクル数	引数
12	x≦0（エラー）
271	0.5≦x＜1
271 - 341	x＜0.5
291 - 305	x≧1

・誤差

	誤差
最大	5.626 E-10
平均	1.449 E-10
中間	－2.122 E-11

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	46.55	80.04	95.22	99.64	100	100

備考　これら誤差は，実際に使用する精度と異なる場合があります。

3.15　fxlog2

[書　式]

```
#include <fxlib.h>
fixed   fxlog10(fixed x);
```

[機　能]

底を 2 とする引数 x の対数を求める関数です。

[制　限]

0.5≦引数 x＜1 の範囲を設定してください。

引数 x＜0.5 または，1≦引数 x を設定した場合，FX_ERRCK が定義されていないと，返却値は不定になります。

[プログラマーズ・ノート]

引数　　：R0H

返却値　　：R0

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：なし

関数内部ではデータ・テーブルの参照，およびその補間により結果を求めます。

・アセンブル条件

FX_SIGNEXT の定義は必要ありません。

FX_ERRCK を定義すると，引数 x＜0.5 または，1≦引数 x の場合，_fx_errno に FX_ERR_ARGS が設定され，返却値は 0 となります。

・実行サイクル数

サイクル数	引数
12	x<0.5（エラー）
30	0.5≦x＜1

・誤差

	誤差
最大	2.081 E-5
平均	7.765 E-6
中間	1.053 E-6

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	49.85	94.56	100	100	100	100

3.16　fxmod

[書　式]

```
#include <fxlib.h>
fixed    fxmod(fixed x, fixed y);
```

[機　能]

x/y の余りを求める関数です。

x/y の余りは，q（商の整数部）を用い，x－q×y で求められます。この関数の返却値は，関数 fxdiv と同一値ではありませんので，注意してください。

例　x=0.25, y=0.1 の場合

x/y は 2.5 になり，この商の整数部は 2 です。

余りは，0.25－2×0.1=0.05 です。この値がこの関数の返却値になります。

例　x=0.3, y=0.1 の場合

返却値は 0 です。

例　x=0.2, y=0.3 の場合

返却値は 0.2 です。

[制　限]

引数 y に 0 を設定しないでください。

[プログラマーズ・ノート]

引数　　：R0H(x)
　　：R1H(y)
返却値　　：R0H(x mod y)
使用レジスタ　：R0, R1, R2, R3, R4
呼び出し関数　：なし

内部アルゴリズムは x と y の大きさを調べます。|x| = |y|の場合，0 が返却値です。
|x|＜|y|の場合，x が返却値です。|x|＞|y|の場合，余りが返却値です。

・アセンブル条件

FX_SIGNEXT を定義すると，引数の符号拡張を行うための 4 命令が追加されます。

FX_ERRCK を定義すると，関数 fxmod は 0 を返し，引数 y=0 で，_fx_errno に FX_ERR_DIV0 が設定されます。FX_ERRCK が未定義の場合，引数 y=0 の返却値は不定になります。

・実行サイクル数

サイクル数	引数
12	y = 0 (エラー)
18	|x|＜|y|
19	|x| = |y|
35~66	|x|＞|y|

・誤差

誤差はありません。

3.17　fxpow

[書　式]

```
#include <fxlib.h>
fixed    fxpow(fixed x, fixed y);
```

[機　能]

x^y（x の y 乗）を求める関数です。

－1.0≦y＜1.0 の範囲なので，引数 x の分数べきが求められます。たとえば，y=0.5 の場合，返却値は $x^{0.5}$（x の平方根）です。y が正の場合，0 - 1 の値になります。y が負の場合，1 より大きい値になります。そのため，返却値は accum 型です。

x, y の設定によって，最大値（255.999...）を越える場合があります。これは，x の値が小さく，y の値が負に大きい場合に発生します。たとえば，x=0.001，y=－0.9 の場合，xy は約 501.2 となり，accum 型として表現することができません。そのような場合，FX_ERRCK の定義に関係なく，最大値（0x7FFFFF FFFF または 255.999...）が返却値になります。

[制　限]

$0 \leqq x < 1.0$

$x^y < 256.0$　　**つまり** $\log_2(x) \times y < 8$　　$8 = \log_2(256)$

引数 x **は，必ず正の数値にしてください。負の数値を引数にすると，結果は複素数になります。引数** y **は有効な** fixed **型を設定してください。**

ここで $\log_2(x) \times y$ **は** 8 **より小さい値にしてください。**$\log_2(x) \times y$ **を** 8 **以上の値に設定すると，**xy **は** 256 **より大きい値になります。**$\log_2(x) \times y \geqq 8$ **の場合，最大可能数値（**0x7FFFFFFFFF **または** 256.999...**）が返却値になります。**

FX_ERRCK **が定義されている場合，**_fx_errno **に** FX_ERR_RANGE **が設定されます。**

[プログラマーズ・ノート]

引数	**：**R0H(x) R1H(y)
返却値	**：**R0H(x^y)
使用レジスタ	**：**R0, R1, R2, R3, R4, DP1
呼び出し関数	**：**fxlog2

関数内部では，次の演算を行います。

$$xy = (2\log_2(x))y = 2(\log_2(x) \times y)$$
$$2\,fract(\log_2(x) \times y) \times 2\,\mathrm{int}(\log_2(x) \times y) = 2(\log_2(x) \times y) = xy$$

はじめに，$0.5 \leqq x < 1$ **の範囲に正規化します。そして，**2 **を底とする対数を得るために関数** fxlog_2 **を使用します。**x **を正規化した際に，シフトした分だけ関数** fxlog_2 **の返却値をシフトします。これに** y **を掛け，**2 **の指数とし，**x^y **と同値にしています。**

この指数は，整数部と小数部に分けられます。まず，$2^{\mathrm{fract}(\log2(x)\times y)}$**を演算します。この結果を，指数の整数部のビット数だけシフトします。つまり，**$2^{\mathrm{int}(\log2(x)\times y)}$**を掛けたものに相当します。**

・アセンブル条件

FX_SIGNEXT **を定義すると，引数** x**，**y **は符号拡張されます。**

FX_ERRCK **を定義すると，引数** x **が負の場合，**_fx_errno **に** FX_ERR_ARGS **が設定されます。**

また，$\log_2(x) \times y$ **の値が** 8 **以上の場合（オーバフローが原因となる），**_fx_error **に** FX_ERR_RANGE **が設定されます。**FX_ERRCK **の設定に関係なく，正の最大値が返却されます。**

・実行サイクル数

サイクル数	引数
12	x＜0（エラー）
14	x = 0
79	0＜x, 0＜y
72	0＜x, y = 0
72	0＜x, 0≦$\log_2(x)\times y$＜1
76	0＜x, 1≦$\log_2(x)\times y$＜8
81	x != 0, 8≦$\log_2(x)\times y$ (エラー）

・誤差

	誤差
最大	4.855 E-4
平均	1.522 E-5
中間	9.691 E-6

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	41.50	67.50	87.30	93.20	95.40	96.60

演算方法（$x^y=2^{\log2(x)\times y}$）のため，返却値が大きい値になると，内部アルゴリズムは誤差を生じます。
返却値を分数とした相対誤差（誤差／返却値）は次のようになります。

	誤差／返却値
最大	3.106 E-5
平均	1.100 E-5
中間	1.053 E-5

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差/返却値率（%）	36.80	71.50	96.00	99.80	100	100

3.18　fxrand

[書　式]

```
#include <fxlib.h>
fixed    fxrand(void);
```

[機　能]

疑似乱数を得る関数です。
返却値は 0x0001(3.05 E-5) - 0x7FEC(0.99939)の fixed 型で，つまり，整数値の 1 - 32748(2^{15}−20)に相当します。
返却値である乱数の数列は，関数 fxsrand の引数（seed 値）設定に依存します。たとえば，関数 fxsrand の引数（seed 値）に 0x0001 を設定すると，関数 fxrand からの初めの返却値は，0x00DB になります。
乱数数列は 32748 を 1 周期とします。この間，同じ値が 2 度発生することはありません。

[制　限]

引数はありません。

[プログラマーズ・ノート]

引数　　：なし

返却値　　：R0H**（乱数）**

使用レジスタ　：R0,R1,R2

呼び出し関数　：なし

関数内部では，次に示す演算を行います。

$$randi + 1 = (randi \times a) \bmod m$$

この場合，乗数 a **は** 219 **注，係数** m **は** 32749 **注です。**

係数 m **は，**$2^{15}-19$ **で，**2^{15} **より小さい最大素数となります。**

注 Pierre L Ecuyer : Efficient and Portable Combined Random Number Generators , *Communications of the ACM*, June 1988, Volume 31, Number6. (1988)

・アセンブル条件

FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

FXRAND_CKRAND **は，現行では定義されていません。定義されている場合，乱数が生成されるたびに** seed **値をチェックする命令が追加されます。**seed **値が** 0＜seed＜32749 **の範囲でなければならないため，**seed **値が設定されるとき，または乱数が生成されるたびにチェックが行われます。**seed **値の設定は，3.23 fxsrand を参照してください。**

・実行サイクル数

サイクル数	引数	備考
15	seed **値のチェックなし（**FXRAND_CKRAND **未定義）**	**ディフォールト**
16	seed **値のチェックあり（**FXRAND_CKRAND **定義）**	－

・誤差

誤差はありません。

3.19　fxrexp

[書　式]

#include <fxlib.h>

fixed　fxrexp(accum x, int *eptr);

[機　能]

引数 x **を** 0.5≦|x|≦1.0 **の範囲に正規化した値を求める関数です。**

eptr **によって示される** YRAM **に** x **を戻すために必要な，底を** 2 **とする指数を格納します。返却値は，**fixed **型の値に丸め処理されます。関数** lfxrexp **の返却値は** long fixed **型です。詳しくは，3.40　lfxrexp を参照してください。**

[制　限]

引数 eptr **には有効な** Y **メモリのアドレスを設定してください。**

[プログラマーズ・ノート]

引数　　**：**R0（x）
　　R1H（eptr）

返却値　　**：**R0H（**正規化された** x）
　　*eptr（**指数**）

使用レジスタ　：R0, R1, R2, DP5

呼び出し関数　：なし

関数内部では，x **の値により** 2 **つの処理が実行されます。**－1≦x＜1 **の場合，**EXP **命令が使用され，左に数ビット・シフトされます。**x＜－1，1≦x **の場合，**x **を右に** 8 **ビット・シフトし，**EXP **命令を使用します。それから，**8 **をひくことによってその値を調整します。返却値は，**x **を適切なビット数で右シフトし，**fixed **型に調整されます。**

- **アセンブル条件**

  FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

- **実行サイクル数**

サイクル数	引数
14	すべての引数

- **誤差**

	誤差
最大	3.052 E-5
平均	7.599 E-6
中間	－1.239 E-7

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	50.26	99.97	99.99	100	100	100

3.20　fxsin

[書　式]

#include <fxlib.h>

fixed　fxsin(fixed x)

[機　能]

引数 x **の正弦を求める関数です。**

引数－π/2～π/2 **の範囲は** fixed **型として表現できないため，引数に** 2/π(0.6366..)**を積算してください。この場合，関数** fxsin **は積算前の引数** x **の正弦を返します。**

[制　限]

引数および返却値で 1.0 **を表現できないため，プログラムでチェックを行う必要があります。**

[プログラマーズ・ノート]

引数　　：R0（x）

返却値　　：R0（sin(π/2×x)）

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：なし

関数内部では，次の演算を行います。

$$\sin(x) = c_1 x + c_2 x^3 + c_3 x^5 + c_4 x^7$$

・アセンブル条件

FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
20	すべての引数

・誤差

	誤差
最大	5.246 E-5
平均	1.147 E-5
中間	－5.173 E-7

	0.25	0.50	0.75	1.00	1.25	1.50	1.75
最下位ビットの誤差率（%）	39.92	72.77	88.70	96.13	99.09	99.87	100

・備考

（1）内部では，使用しているレジスタをシンボル定義しています。シンボル定義を変更することによって，使用するレジスタを変更できます。

(2) 内部で使用する係数は，DP1 を使用して XRAM に格納します。YRAM に係数を格納したい場合，DP5 に変更してください。

(3) 演算の過程で ROUND 命令を 4 箇所使用しています。そのため，これらの命令を省くことで，処理速度を 16 サイクルから 12 サイクルにすることができます。その精度を次に示します。これは，ROUND 命令を使用したアルゴリズムの誤差の 2 倍にあたります。

	誤差
最大	1.084 E-4
平均	2.209 E-5
中間	1.886 E-5

ROUND 命令を使用しない場合，引数－1.0 の入力時には正しい値を返しませんので注意してください。このため，引数をチェックする処理を追加しなければなりません。ROUND 命令を抜いた fxsin ルーチンは，引数に－1.0 を与えない場合のみ有効です。

(4) プログラム内で，演算精度よりも処理時間に重点をおいて，現在使用している 4 つの係数を 3 つにします。これにより，このルーチンの実行時間は 12＋4 で，合計 16 サイクルになります（call，return 命令を含みます）。

```
COEF:
        DW      0x4A10      ; C2
        DW      0x6482      ; C0
        DW      0xD6DF      ; C1
                  :
                  :
                  :
        sum = sum sll 4;
        xpow = xpowH * xsqH;
        xpow = round( xpow );                                  } sum = sum sra 4;
        sum = sum - xpowH*coefH     coef = *dp1++;
        sum = sum sra 8;
```

この変更による精度を次に示します。

	誤差
最大	1.305 E-4
平均	3.779 E-5
中間	3.689 E-5

3.21　fxsinh

[書　式]

```
#include <fxlib.h>
accum   fxsinh(fixed x);
```

[機　能]

引数 x **の双曲正弦を求める関数です。**

引数 x=－1.0，x=0.999...**で返却値は最大となり，約**±1.175 **を返します。**

[制　限]

引数 x **は** fixed **型を設定します。**

[プログラマーズ・ノート]

引数　　：R0(x)

返却値　　：R0H(sinh)

使用レジスタ　：R0, R1, R2, R3, DP1

呼び出し関数　：**なし**

関数内部では，次の演算を行います。

$$\sinh(x) = c_1 x + c_2 x^3 + c_3 x^5$$

- **アセンブル条件**

  FX_SIGNEXT **が定義されると，引数の符号拡張を行う** 2 **命令が追加されます。**

  FX_ERRCK **の定義は必要ありません。**

- **実行サイクル数**

サイクル数	引数
14	すべての引数

- **誤差**

	誤差
最大	1.348 E-5
平均	3.748 E-6
中間	5.400 E-6

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	89.61	100	100	100	100	100

3.22　fxsqrt

[書　式]

#include <fxlib.h>

fixed　　fxsqrt(fixed x);

[機　能]

引数 x **の正の平方根を求める関数です。**

[制　限]

引数 x≧0 **を設定してください。**

x＜0 **の場合，**FX_ERRCK **が定義されていないと返却値は不定になります。**

[プログラマーズ・ノート]

引数　　**：**R0H(x)

返却値　　**：**R0H(sqrt)

使用レジスタ　　**：**R0, R1, R2, R3, R4, DP1

呼び出し関数　　**：なし**

内部アルゴリズム[注]は，引数を 0.25-1 **の範囲に正規化します。そして，シフト量を保存します。次に示す演算で，最初の評価を行います。**

$$Beta = ax^2 + bx + c$$
$$sqrt_0 = dx + e$$

平方根の評価は，ベータ値を使用した次の演算を 2 **回行います。**

$$sqrt_{i+1} = sqrt_i + Beta \times (x - sqrt_i^{\,2})$$

この結果，（シフト値/2**）の数だけ右シフトし，正規化され，丸め処理が行われます。**

注 Mikami, Kobayashi & Yokoyama : A New DSP-Oriented Algorithm for Calculation of the Square Root Using a Nonlinear Digital Filter , *IEEE Transactions on Signal Processing*, Vol.40, No.7, July 1992.(1992)

・アセンブル条件

FX_ERRCK **を定義すると，引数がチェックされます。引数が** 0 **未満の場合，**_fx_errno **に** FX_ERR_AGES **が設定され，返却値が** 0 **になります。また，引数が** 0 **以上の場合，**_fx_errno **に** 0 **が設定されます。**FX_SIGNEXT **を定義すると，引数を符号拡張するための** 2 **命令が追加されます。**

この関数に定義されている ROUND_RSLT **は，結果に対し，正規化や丸め処理といった処理方法を定義するシンボルです。このシンボルは** FXSQRT.ASM **ファイル内に定義されています。その値と処理内容は次のとおりです。**

値	速度	精度	命令数	備考
0	速い	欠ける	2	－
1	普通	普通	3	－
2	遅い	正確	7	ディフォールト

・実行サイクル数

サイクル数	引数
10	x＜0（エラー）
13	x = 0
39	0＜x

・誤差

	誤差
最大	1.614 E-5
平均	7.598 E-6
中間	－1.023 E-7

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	50.21	99.99	100	100	100	100

3.23　fxsrand

[書　式]

#include < fxlib.h>

void　fxsrand(fixed seed);

[機　能]

疑似乱数の初期値を設定する関数です。

引数（seed 値）を 0x0001 に設定してください。引数（seed 値）をほかの値に設定すると，その値が疑似乱数の初期値となります。

[制　限]

0x0000＜seed＜0x7FED の範囲にしてください。関数 fxrand の内部アルゴリズムは，0x7FED （2^{15}－19 つまり 32749） を係数として使用しています。

[プログラマーズ・ノート]

引数　　：R0H(seed)

返却値　　：なし

使用レジスタ　：R0, R1

呼び出し関数　：なし

fxsrand ルーチンは，FXRAND.ASM ファイル内の関数 fxrand とともに記述されています。

・アセンブル条件

FX_SIGNEXT および FX_ERRCK の定義は必要ありません。

FXRAND_CKSEED を定義すると（ディフォールト設定時），引数 seed 値について範囲のチェックが行われます。範囲を越えた場合，0x0001 に設定します。

この関数の引数が，必ず範囲内であると保証している場合，または乱数生成時にチェックする

FXRAND_CKRAND が定義されている場合，FXRAND_CKSEED の定義は必要ありません。

・実行サイクル数

サイクル数	引数
10	0＜seed＜0x7FED
11	seed≦0
12	0x7FED≦seed
5	FXRAND_CKSEED が定義されていない場合

・誤差

返却値はありません。

3.24　fxtan

[書　式]

#include < fxlib.h>

accum　fxtan(fixed x);

[機　能]

引数 x の正接を求める関数です。

引数－π/2～＋π/2 の範囲は fixed 型として表現できないため，引数に 2/π(0.6366..)を積算してください。この場合，関数 fxtan は積算前の引数 x の正接を返します。

[制　限]

－0.9975 (0x8052)≦x≦0.9975 (0x7FAE)

返却値は±π/2 に相当する±1.0 を返せませんので注意してください。この場合，約±1.567 ラジアン（約±89.7748 度）に相当する約±0.9975 を返します。

[プログラマーズ・ノート]

引数	：R0H(x)
返却値	：R0H(tan)
使用レジスタ	：R0, R1, R2, R3, R4, DP1
呼び出し関数	：なし

関数内部では，次の演算を行います。

$$\tan(x) = \frac{\sin(x)}{\cos(x)}$$

・アセンブル条件

FX_SIGNEXT および FX_ERRCK の定義は必要ありません。

・実行サイクル数

サイクル数	引数
29	x = －1.0
74	|x|＜0.5
102	0.5≦|x|≦0.9975
69	0.9975＜|x|＜1

・誤差

関数内部の演算 sin(x)/cos(x)では，cos(x)が値 0 に近似すると誤差が大きくなるなどの理由により，誤差率は次の条件で表現しています。

	誤差	
	|tan|≧1	|tan|＜1
最大	1.029	7.399 E-5
平均	0.002	6.6654 E-6
中間	0	0
最大／tan	0.0066	0.273
平均／tan	4.418 E-5	8.985 E-5

	1	2	3	4	5	6	7
最下位ビットの誤差率（%）	59.53	72.5	77.99	81.74	83.80	85.36	86.76

3.25　fxtanh

[書　式]

```
#include < fxlib.h>
fixed    fxtanh( fixed x );
```

[機　能]

引数 x の双曲正接を求める関数です。

accum 型，または long fixed 型で返却値を得る場合，プログラマーズ・ノートの備考を参照してください。

[制　限]

引数 x は fixed 型を設定します。

[プログラマーズ・ノート]

引数　　：R0(x)

返却値　　：R0H(tanh)

使用レジスタ　：R0,R1,R2,R3,DP1

呼び出し関数　：なし

関数内部では，次の演算を行います。

$$\tanh(x) = c_1 x + c_2 x^3 + c_3 x^5 + c_4 x^7 + c_5 x^9$$

・アセンブル条件

FX_SIGNEXT および FX_ERRCK の定義は必要ありません。

・実行サイクル数

サイクル数	引数
18	すべての引数

・誤差

	誤差
最大	2.812 E-5
平均	7.993 E-6
中間	4.070 E-7

・備考

関数 fxtanh は演算結果を丸め処理により fixed 型にし，R0 に値を返しています。

丸め処理が行われない場合，返却値は accum 型，または long fixed 型で扱うことができます。これにより，丸め処理を行うよりも正確な値が得られ，短い時間で結果を得ることができます。処理速度を向上したいと望むユーザは，accum 型，または long fixed 型の返却値を使用することができます。

そのために，返却される前に丸め処理を除いたり，FXLIB.H 内の関数 fxtanh の宣言を accum 型，または long fixed 型に変更する必要があります。

```
/*   FXLIB.H   */
      :
fixed fxtanh( fixed );   → long fixed fxtanh( fixed );
      :                    または，  accum fxtanh( fixed );
```

accum 型，または long fixed 型で求めた演算結果をさらに fixed 型に変換すると精度が落ちます。その場合，次に示す処理を行います。

```
/*  ユーザ・プログラム  */
      :                          :
fixed t, x;                fixed t, x;
      :          →               :
t = tanh( x );             t = (fixed ) tanh(x);
      :                          :
```

しかし，これは丸め処理ではなく，accum 型，または long fixed 型の返却値を切りつめて得ているため，精度が落ちます。

	誤差
最大	1.339 E-5
平均	3.253 E-6
中間	1.028 E-6

結果を accum 型，または long fixed 型として扱う場合，命令が増えます。結果を fixed 型として扱う場合，誤差が増えます。accum 型，または long fixed 型に変更したこの関数を，キャストにより返却値を fixed 型で得る場合にも，誤差が増えます。

```
long fixed fxtanh(fixed);    /* in FXLIB.H */
fixed t, x;
t = (fixed) fxtanh( x );
```

	誤差
最大	4.277 E-5
平均	1.695 E-5
中間	1.713 E-5

3.26　lfxabs/_lfxabs

[書　式]

```
#include <fxlib.h>
long fixed    lfxabs(long fixed x);
long fixed    _lfxabs(long fixed x);
```

[機　能]

引数 x の絶対値を求める関数です。
引数が 0≦x はそのままの値を，x＜0 は負を積算した値を返します。
絶対値の演算では関数 lfxabs(long fixed x)と，マクロ_lfxabs(long fixed x)の 2 つを用意しています。

[制　限]

引数が－1.0 に対し，返却値 1.0 を表現できません。そのため，CLIP（クリップ）命令によるクリッピングを行う処理が含まれています。このクリッピングは LFXABS_CLIP の定義により実行されます。

[プログラマーズ・ノート]

引数　　：R0H/L
返却値　　：R0H/L
使用レジスタ　：R0
呼び出し関数　：なし

この関数は，サブルーチンとマクロの 2 つを用意しています。これらは同等の動作を行います。関数内部では，ABS（絶対値）命令を使用しています。

・アセンブル条件

FXABS_CLIP を定義すると，返却値のクリッピング処理を行う命令が追加されます。

FX_SIGNEXT を定義すると，引数の符号拡張を行う 2 命令が追加されます。

これにより，R0E には符号拡張された値が設定されるようになります。

FX_SIGNEXT を無効にした場合は，呼び出し時に R0E に R0H/L の符号拡張を行った値を設定してください。たとえば，レジスタ R0E に 0xFF，レジスタ R0H/L に 0x00000001 を設定することで，0xFFFFFFFF が返されます。

FX_ERRCK の定義は必要ありません。

・実行サイクル数

ルーチン	サイクル数	引数
関数 lfxabs	8	FXABS_CLIP 定義（ディフォールト）
	7	FXABS_CLIP 未定義
マクロ_lfxabs	4	FXABS_CLIP 定義（ディフォールト）
	3	FXABS_CLIP 未定義

・誤差

	誤差		備考
	クリッピングあり	クリッピングなし	
最大	4.657 E-10	2	引数－1.0 の場合
平均	0 注	0 注	－
中間	0	0	－

注　引数－1.0 のときの誤差を除くと，最大，平均，中間誤差はほぼすべて 0 になります。

3.27　lfxacos

[書　式]

```
#include <fxlib.h>
long fixed    lfxacos(long fixed x);
```

[機　能]

引数 x の逆余弦を求める関数です。

通常，逆余弦は 0 - πラジアンになりますが，これは long fixed 型の範囲を外れるため，この関数では 0≦返却値＜1.0 の範囲にして返します。このため弧度は，返却値とπの積算で求めます。

[制　限]

引数は long fixed 型（0x8000000(－1.0) - 0x7FFFFFF(0.9999...)の範囲）のため，値 1.0 は与えられません。また，返却値も同様に値 1.0 を返せませんので注意してください。

たとえば，値－1.0 の逆余弦は値πになりますが，この関数では値 1.0 をかえすところを 0x7FFFFFFF としています。このため引数や返却値のチェックを行う必要があります。

[プログラマーズ・ノート]

引数　　：R0H/L

返却値　　：R0H/L

使用レジスタ　　：R0, R1, R2, R3, R4, DP1

呼び出し関数　　：lfxasin（関数 lfxasin の内部では，関数 lfxsqrt と関数 lfxatan を使用しています）

関数内部では，関数 lfxasin を呼び出し，次の演算を行っています。

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x)$$

・アセンブル条件

FX_SIGNEXT の定義は，関数 lfxasin で行うため，この関数での定義は必要ありません。関数 lfxasin で定義してください。

FX_ERRCK の定義は必要ありません。

・実行サイクル数

サイクル数	引数
17	x = －1
126	0≦x＜1.53 E-5
260	1.53 E-5≦|x|＜0.447
334	0.447≦|x|＜0.707
333	0.707≦|x|＜0.949
407	0.949≦|x|＜1.0

・誤差

	誤差	備考
最大	1.174 E-9	すべての誤差のうち 99%は約 4 E-10 を下回っています
平均	1.590 E-10	－
中間	1.160 E-10	－

	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.75
最下位ビットの誤差率(%)	43.50	74.08	93.03	99.32	99.90	99.92	99.95	100

3.28　lfxasin

[書　式]

#include <fxlib.h>

long fixed　lfxasin(long fixed x);

[機　能]

引数 x **の逆正弦を求める関数です。**

通常，逆正弦は－π/2～π/2 **ラジアンになりますが，これは，**long fixed **型の範囲を外れるため，この関数では**－1.0≦**返却値**＜1.0 **の範囲にして返します。このため弧度は，返却値と**π/2(1.5707...)**の積算で求めます。**

[制　限]

引数は long fixed **型（**0x8000000(－1.0) - 0x7FFFFFF(0.9999...)**の範囲）のため，値** 1.0 **は与えられません。また，返却値も同様に** 1.0 **を返せませんので注意してください。**

たとえば，値 1.0 **の逆正弦は値**π/2 **となりますが，この関数では値** 1.0 **を返すところを** 0x7FFFFFFF **としています。このため引数や返却値のチェックを行う必要があります。**

[プログラマーズ・ノート]

引数　　**：**R0H/L

返却値　　**：**R0H/L

使用レジスタ　**：**R0, R1, R2, R3, R4, DP1

呼び出し関数　**：**lfxsqrt, lfxatan

関数内部では，関数 fxatan **や関数** fxsqrt **を呼び出し，次の演算を行います。**

$$\arcsin(x) = \arctan\left(\frac{x}{sqrt(1-x^2)}\right)$$

ただし，引数が 1.0 **に近似すると誤差が生じるため，**0.707≦x＜1 **の範囲では，次の式**

$$\arcsin(x) = \frac{\pi}{4} + \arcsin\left(\frac{x - sqrt(1-x^2)}{sqrt(2)}\right)$$

を置換した演算を行います。

$$\arcsin(x) = \frac{\pi}{4} + \arctan\left(\frac{(x - sqrt(1-x^2))}{sqrt(1+2\times x\times sqrt(1-x^2))}\right)$$

・アセンブル条件

FX_SIGNEXT **を定義すると，引数の符号拡張を行う** 2 **命令が追加されます。**

FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
8	x＝－1
117	0≦x＜1.53 E-5
251	1.53 E-5≦|x|＜0.447
325	0.447≦|x|＜0.707
324	0.707≦|x|＜0.949
398	0.949≦|x|＜1.0

・誤差

	誤差	備考
最大	2.240 E-9	すべての誤差のうち 99%は約 6 E-10 を下回っています
平均	1.672 E-10	－
中間	－3.675 E-13	－

	0.25	0.50	0.75	1.00	1.25	1.50	1.75	5.00
最下位ビットの誤差率(%)	42.02	73.55	91.27	97.73	99.37	99.73	99.82	100

3.29　lfxatan

[書　式]

#include <fxlib.h>

long fixed　lfxatan(accum x);

[機　能]

引数 x の逆正接を求める関数です。

引数 x は accum 型（40 ビット長の固定小数点）で，その範囲は－256.0～＋255.999... (0x8000000000 - 0x7FFFFFFFFF)になります。

通常，逆正接は－π/2～π/2 ラジアンになりますが，これは long fixed 型の範囲を外れるため，この関数では－1.0 ≦返却値＜1.0 の範囲にして返します。このため弧度は，返却値とπ/2(1.5707...)の積算で求めます。

[制　限]

引数は accum 型を設定します。

返却値は，±π/2 に相当する±1.0 を返せませんので注意してください。

この場合，返却値は 1.567（約±89.7747 度）に相当する約±0.9975 を返します。

[プログラマーズ・ノート]

引数　　：R0

返却値　　：R0H/L

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：なし

引数が 0≦|x|＜0.5 のときは，arctan(x)の演算を行い，0.5≦|x|≦1 の場合は次の演算を行います。

$$\arctan(x) = \arctan(0.5) + \arctan\left(\frac{x-0.5}{1+0.5\times x}\right)$$

1＜|x|≦2 の範囲（0 から 0.5 を含め）は次の演算を行います。

$$\arctan(x) = \frac{\pi}{2} - \left\{\arctan(0.5) + \arctan\left(\frac{2-x}{2x+1}\right)\right\}$$

|x|＞2 の範囲は次の演算を行います。

$$\arctan(x) = \frac{\pi}{2} - \arctan\left(\frac{1}{x}\right)$$

・アセンブル条件

FX_SIGNEXT および FX_ERRCK の定義は必要ありません。

・実行サイクル数

サイクル数	引数
103	0≦|x|＜0.5
177	0.5≦|x|≦1
175	1＜|x|≦2
178	2＜|x|≦4
181	4＜|x|

・誤差

	誤差
最大	7.541 E-10
平均	1.454 E-10
中間	0

	0.25	0.50	0.75	1.00	1.25	1.50	1.75
最下位ビットの誤差率（%）	45.18	82.59	95.19	99.4	99.98	99.99	100

3.30　lfxatan2

[書　式]

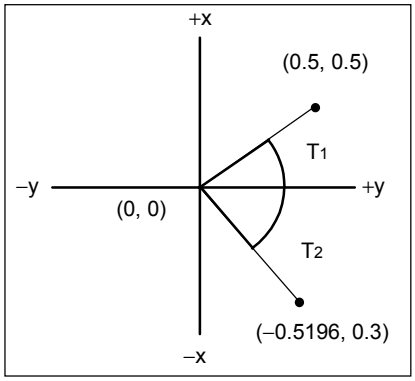
```
#include <fxlib.h>
long fixed    lfxaatan2(long fixed x, long fixed y);
```

[機　能]

引数 x/y **の逆正接を求める関数です。**

この関数は，引数 x, y **をそれぞれ平面の垂直軸，水平軸の座標**(x, y)**とし，＋側の** y **軸線と，原点**(0,0)**，座標**(x, y)**を結ぶ線の成す角をラジアンとする** T **を返却値として返します。**

通常，逆正接は－π～＋πの値をとりますが，これは long fixed 型の範囲を外れるため，この関数では
－1.0≦返却値＜1.0 の範囲にして返します。このため弧度は，返却値とπの積算で求めます。
たとえば，引数 x=0.5, y=0.5 を設定すると，返却値に 0.25 を返します。これにπを積算して得られた値π/4 が角 T_1 となります。
引数 x=－0.5196, y=0.3 を設定とすると，返却値に－0.333...を返します。これにπを積算して得られた値－π/3 となり，この結果角 T_2 にあたり，＋側の y 軸線と原点(0.0)，座標(－0.5196, 0.3)を結ぶ線の作る角と同一になります。



この関数は，引数 x=0, y≧0 の場合に，値 0 を返し，引数 x=0, y＜0 の場合に，値 1.0(π)ではなく 0x7FFFFFFF(0.999....)を返します。
また，引数 x≧0, y=0 の場合値，0.5(π/2)を，引数 x＜0, y=0 の場合，値－0.5(－π/2)を返します。

[制　限]

引数 x および y には値 0 を設定しないでください。
引数 x=0, y=0 を設定した場合，FX_ERRCK が定義されていないと，返却値は不定になります。

[プログラマーズ・ノート]

引数	：R0H/L(垂直軸:x)
	R1H/L(水平軸:y)
返却値	：R0H/L(1/π×arctan(x/y))
使用レジスタ	：R0, R1, R2, R3, R4, DP1
呼び出し関数	：なし

この関数では引数は絶対値で演算し，その座標を固定象限上に置きます。内部の演算では引数が 0≦|x|/|y|＜0.5 の場合は arctan(x/y)の演算を，0.5≦|x|/|y|≦1 の場合は次の演算を行います。

$$\arctan\left(\frac{x}{y}\right)=\arctan(0.5)+\arctan\left(\frac{2x-y}{x+2y}\right)$$

1＜|x|/|y|≦2 の場合は次の演算を行います。

$$\arctan\left(\frac{x}{y}\right)=\frac{\pi}{2}-\left\{\arctan(0.5)+\arctan\left(\frac{2x-y}{x+2y}\right)\right\}$$

2＜|x|/|y|の範囲では次の演算を行います。

$$\arctan\left(\frac{x}{y}\right)=\frac{\pi}{2}-\arctan\left(\frac{y}{x}\right)$$

・アセンブル条件

FX_SIGNEXT を定義すると，引数の符号拡張を行う 2 命令が追加されます。

FX_ERRCK を定義すると，関数のエラー・チェックをする命令が追加されます。

たとえば引数 x=0, y=0 の場合では，_fx_errno に FX_ERR_DIV0 を設定し，返却値に値 0 を返すようになります。この場合，実行時間に 6 サイクルが加わります。

・実行サイクル数

サイクル数	引数
12	x＝y＝0（エラー）
198	0≦|x/y|＜0.5
201	0.5≦|x/y|＜1
197	1≦|x/y|＜2
194	2≦|x/y|

・誤差

	誤差
最大	5.211 E-10
平均	1.811 E-10
中間	－3.708 E-12

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率(%)	38.26	64.70	88.14	99.47	100	100

3.31　lfxcos

[書　式]

```
#include <fxlib.h>
long fixed    lfxcos(long fixed x);
```

[機　能]

引数 x の余弦を求める関数です。返却値は cos(π/2×x)を返します。

引数－π/2～＋π/2 の範囲は，long fixed 型として表現できないため，引数に 2/π(0.6366...)を積算してください。これにより得られる返却値は積算する前の引数の余弦が得られます。

[制　限]

値 0 の余弦は＋1.0 です。long fixed 型で表現できないため，この場合，返却値には 0x7FFFFFFF(0.99999...)を返します。このとき最下位ビットに誤差(2^{-31})が発生します。

[プログラマーズ・ノート]

引数　　：R0H/L

返却値(指数)　：R0H/L

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：なし

関数内部では，引数に関係なく次の演算を行います。

$$\cos(x) = 1 + c_1 x^2 + c_2 x^4 + c_3 x^6 + c_4 x^8 + c_5 x^{10}$$

・アセンブル条件

FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
93	**すべての引数**

・誤差

	誤差
最大	8.137 E-10
平均	2.168 E-10
中間	－6.924 E-11

	0.25	0.50	0.75	1.00	1.25	1.50	1.75
最下位ビットの誤差率（%）	32.75	59.75	79.73	91.37	97.37	99.65	100

3.32　lfxcosh

[書　式]

#include <fxlib.h>

accum　lfxcosh(accum x);

[機　能]

引数 x **の双曲余弦を求める関数です。**

引数 x=－1.0 **で返却値は最大となり，約** 1.543 **を返します。引数は，関数** fxcosh **と同じ** accum **型です。**

[制　限]

引数 x **には，**|x|＜6.24(arccosh(256))**の条件を満たす値を設定してください。**

引数 x **に，**6.24(arccosh(256))≦|x|**になる値を設定した場合，**accum **型の最大値** 0x7FFFFFFFFF（＋255.999...)
を返却します。

[プログラマーズ・ノート]

引数　　：R0

返却値　　：R0

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：lfxexp

関数内部では，次の演算を行います。

$$\cosh(x) = \frac{e^{x} + e^{-x}}{2} = \frac{e^{x}}{2} + \frac{e^{-x}}{2}$$

$$\frac{e^{x}}{2} = e^{(x-\ln(2))} \qquad \frac{e^{-x}}{2} = \frac{1}{2 \times e^{x}} = \frac{1}{4 \times \frac{e^{x}}{2}} = \frac{0.25}{e^{(x-\ln(2))}}$$

$$\cosh(x) = \frac{e^{(x-\ln(2))} + 0.25}{e^{(x-\ln(2))}}$$

cosh(x)=cosh(－x)**であることから，引数** x **は正の値として演算します。**

引数 x **から** ln(2)**を減算し，上記演算の** $e^{(x-\ln(2))}$**を求めるため，関数** lfxexp **を呼び出します。その後，**0.25/ $e^{(x-n(2))}$ **の演算の解と** $e^{(x-\ln(2))}$**との和を双曲余弦の値とします。**

・アセンブル条件

FX_SIGNEXT **の定義は必要ありません。**

FX_ERRCK **を定義すると，|引数** x|≧6.24 **の場合，**_fx_errno **に** FX_ERR_RANGE **を設定し，エラーがない場合，**_fx_errno **に値** 0 **が設定されます。これは関数** lfxexp **によって設定されます。**

・実行サイクル数

サイクル数	引数
13	x=－256
109	6.238≦|x|＜256
261	1.693≦|x|＜6.238 (ln(2)＋1≦|x|＜ln(512))
228	0.693≦|x|＜1.693(ln(2)≦|x|＜ln(2)＋1)
254	0≦|x|＜0.693(0≦|x|＜ln(2))

・誤差

引数 x＜1.0 **の場合，ほぼ** 1LSB($\pm 2^{-31}$)**の精度にあります。**

1≦**引数** x＜1 **の場合，演算（**x－ln(2)**）の値が増加するのにともない誤差も増加します。**

引数 x＜1.0

	誤差
最大	5.800 E-10
平均	1.383 E-10
中間	2.772 E-11

	0.25	0.50	0.75	1.00	1.25	1.50	1.75
最下位ビットの誤差率（%）	46.83	84.57	96.87	99.70	100	100	100

－6.25≦引数 x≦6.25

	誤差
最大	4.264 E-8
平均	2.812 E-9
中間	3.423 E-10

そのほかの範囲

	誤差					
	1≦x＜2	2≦x＜3	3≦x＜4	4≦x＜5	5≦x＜6	6≦x＜6.25
最大	1.354 E-9	2.183 E-9	5.914 E-9	1.629 E-8	4.257 E-8	4.264 E-8
平均	2.530 E-10	5.127 E-10	1.079 E-9	3.092 E-9	8.437 E-9	1.403 E-8
中間	1.350 E-10	4.277 E-10	4.109 E-10	1.878 E-9	5.486 E-9	1.171 E-8

・備考

関数内部の変更では次に示す 2 点があります。

（1）現行では演算 $0.25/ e^{(x-\ln(2))}$では 1 ループ中で 4 ビットづつ演算しています。この命令を 1 ループ中でより多いビットを対象に行い，より少ないループ回数で行うことで実行時間を省きます。しかしこれにより，演算の精度は実行時間に反比例して低くなります。

（2）現行では関数 lfxexp の返却値をチェックしていません。これに次に示すソース・リストのように返却値のチェック（0x7FFFFFFFFF であるかチェック）をする命令 2 行を挿入することで，引数が 6.238≦|x|＜256 の範囲では約 35 クロック（現在 109 クロック）を省くことができます。
しかし，このクロック数は関数 lfxexp が 0x7FFFFFFFFF を返すときのみに有効であることから，返却値をチェックしない現行の方法をお勧めします。

```
call lfxexp;
/* insert the following two lines */
r1 = r0 + 1;                        /* check for 0x7FFFFFFFFF */
if (r1 < 0) jmp _lfxcosh_toobig;            /* if goes to 0x8000000000 */
/* end of insert */
r4 = r0;
```

3.33　lfxdiv

[書　式]

#include <fxlib.h>

long fixed lfxdiv(long fixed x, long fixed y);

[機　能]

引数 x **を被除数，**y **を除数とする商および余りを求める関数です。**

引数 x, y **は** long fixed **型（**31 **ビット＋**1 **サイン・ビット），返却値は** long fixed **型です。**

[制　限]

引数 y **には** 0 **を設定しないでください。**

引数 x, y **には**|x|＜|y|**の条件を満たす値を設定してください。**

[プログラマーズ・ノート]

引数　　**：**R0H/L(**引数** x)

　　R1H/L(**引数** y)

返却値　　**：**R0H/L

使用レジスタ　：R0, R1, R2, R3, R4

呼び出し関数　：なし

関数内部では，DIV **命令（演算ごとに** 1 **ビットずつ求める，除算命令）を用い，**long fixed **型の演算を行います。**

この DIV **命令は，値** 128 **未満である間実行し，これにより引数** x, y **は** 7 **ビット左シフトされます。**

はじめの DIV **命令は，商の** 4 **ビットを求めます。そのあとのループは** 7 **つの** DIV **命令を記述し** 4 **回ループで** 28 **ビット，合計** 32 **ビットの商を得ます。**

この演算のあと，DIV **命令により商の次のビットを求めています。これは商の値を** long fixed **型にするための丸め処理にあたります。この結果，余りのない商を求めています。**

・アセンブル条件

FX_SIGNEXT **を定義すると，引数の符号拡張を行う** 4 **命令が追加されます。**

内部ではレジスタ R0, R1 **の全ビットを使用するため，符号拡張は有効にしておいてください。**

FX_SIGNEXT **を無効にした場合は，**R0, R1 **に符号拡張を行った値と同等の値を設定してください。**

FX_ERRCK **を定義すると，関数のエラー・チェックを行う** 13 **命令が追加されます。**

これにより，エラーなしでも 11 **クロック多い実行時間がかかります。**

引数にエラーがある場合，返却値に値 0 **が返され，**_fx_errno **に次に示す値が設定されます。**

エラー条件	内容
y=0	FX_ERR_DIV0
y≦x	FX_ERR_ARGS
エラーなし	0

・実行サイクル数

サイクル数	引数
12	y=0（エラー時）
20	|y|≦|x|（エラー時）
77	その他

・誤差

	誤差
最大	2.327 E-10
平均	1.019 E-10
中間	0

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率(%)	55.61	100	100	100	100	100

3.34　lfxexp

[書　式]

#include <fxlib.h>

accum　lfxexp(accum x);

[機　能]

引数 x を正規化するのに必要なシフト量を求める関数です。

返却値は e^x となります。

[制　限]

引数 x には，|x|＜5.545 (ln(256))の条件を満たす値を設定してください。

引数 x に，5.545(ln(256))≦|x|になる値を設定した場合，accum 型の最大値 0x7FFFFFFFFF(＋255.999...)を返却します。

[プログラマーズ・ノート]

引数　　：R0

返却値　　：R0

使用レジスタ　：R0, R1, R2, R3, DP1

呼び出し関数　：なし

関数内部の評価は引数 x の符号（±）に依存しています。

0≦x＜1 の範囲では，次の演算を行います。

$$e^x = 1 + c_1x^1 + c_2x^2 + c_3x^3 + c_4x^4 + c_5x^5 + c_6x^6 + c_7x^7 + c_8x^8$$

引数 x が負の場合，まず，x＜－22 の範囲では，値 0 を返します。

引数 x が－22＜x＜0 の範囲では，次の演算を行います。

$$y = \frac{x}{\ln(2)} \quad yi = floor(y) \quad yf = y - yi$$ （x/ln(2)の整数および小数部）

$$e^x = 2^y = 2^{(yi+yf)} = 2^{yi} \times 2^{yf} \quad z = y^f \times \ln(2)$$

$$e^x = 2^{yi} \times e^z$$

これら演算は，y, yi, yf そして z(y^f×ln(2))を得るために行われます。上記多項式は e^z を求め，x が x＜0 のとき yi は常に負の整数になるため，2^{yi} の乗算式は yi の値だけ右シフトすることになります。

引数 x が正のとき，まず x＞5.545(ln(256))の範囲では，accum 型の最大値を返し，_fx_errno に FX_ERR_RANGE を設定します。
引数 x が 0＜x≦5.545(ln(256))の範囲では，次の演算を行います。

$$xi = floor(x) \quad xf = x - xi$$ （x の整数および小数部）

$$e^x = e^{(xi+xf)} = e^{xi} \times e^{xf}$$

まず，引数 x を xi(0 - 5)と xf に分け，e^{xf} を求める演算を行います。xi=0 の場合，e^{xf} のみを求め，1≦xi≦5 の場合，e^{xi} の値はデータ・テーブルより得ます。
これにより e^{xi}×e^{xf} の演算を行います。

・アセンブル条件

FX_SIGNEXT の定義は必要ありません。

FX_ERRCK が定義されている場合，引数 x が約 5.545 を越える値では，_fx_errno に FX_ERR_RANGE が設定され，エラーがない場合は_fx_errno をクリアします。

・実行サイクル数

この関数の性能は引数 x により変化します。

サイクル数	引数
11	x≦－22
167	－22＜x＜0
141	0≦x＜1
174	1≦x＜5.545
17	x≧5.545(ln(256))

・誤差

引数 x が x＜1.0 の範囲にある場合，多くは±1/2LSB(±2^{-32})の精度です。

引数 x が 1≦x＜5.545 の範囲にある場合，引数が増えるに従って精度は低くなります。これは e^{xi} と積算する多項式の演算に起因します。

引数 x＜1.0

	誤差
最大	4.968 E-10
平均	1.309 E-10
中間	－7.151 E-12

	0.25	0.50	0.75	1.00	1.25	1.50	1.75
最下位ビットの誤差率（%）	48.07	86.72	98.75	99.95	100	100	100

そのほかの範囲

	誤差					
	1≦x＜2	2≦x＜3	3≦x＜4	4≦x＜5	5≦x＜5.545	－256≦x＜256
最大	1.140 E-9	2.886 E-9	6.981 E-9	1.785 E-8	3.613 E-8	3.613 E-8
平均	3.211 E-10	6.907 E-10	1.705 E-9	4.609 E-9	9.246 E-9	2.050 E-11 注
中間	2.326 E-10	－2.126 E-10	9.426 E-11	7.201 E-10	1.631 E-9	0 注

注　この範囲では，たいていの誤差が 0 **あるいは** 0 **に近くなります。条件** x＜－22（2.79 E-10 **に満たない誤差）か，条件** x＞5.545(**返却値** 255.999...**かつ** FX_ERR_RANGE **状態)では誤差** 0 **としています。**

[注　意]

この関数は，lfxcosh, lfxsinh, lfxtanh, lfxpow **から呼ばれるため，この関数の機能または演算精度の変更は行わないでください。それら呼び出す関数の精度および性能に影響を与えることになります。**

3.35　lfxldexp

[書　式]

```
#include <fxlib.h>
accum    lfxldexp(long fixed x, int exp);
```

[機　能]

引数 x, exp **から，演算（**$x \times 2^{exp}$**）の結果を得る関数です。**

引数 exp **の値に応じた引数** x **のビット・シフトを行います。この場合，正のときは左シフト，負のときは右シフトを行います。**

返却値は 1.0 **以上の値になるため** accum **型になります。**

関数 fxldexp **では，引数それぞれが** fixed **型と** int **型，返却値が** accum **型となっています。詳しくは，3.12 fxldexp を参照してください。**

[制　限]

－63≦exp≦63

$-256 \leqq x \times 2^{exp} < 256$

引数 x **は** long fixed **型を設定します。**

引数 exp **は** int **型として定義されていますが，下位** 6 **ビットはシフト・カウント，最上位** 1 **ビットは符号ビットになります。**|exp|＞40 **は，返却値の全ビットが** 0 **または** 1 **になりますので注意してください。**

FX_ERRCK **を定義し，**|**返却値**|＞256(**正では** 0x7FFFFFFFFF(255.999...)**，負では** 0x8000000000(－256))**になる引数を与えると，**_fx_errno **に** FX_ERR_RANGE **が設定されます。**

[プログラマーズ・ノート]

引数　　：R0H/L(**引数** x)
　　R1H(**引数** exp)

返却値(指数)　：R0

使用レジスタ　：R0, R1, R2, R4

呼び出し関数　：**なし**

引数 exp **の符号（正：左シフト，負：右シフト）により，引数** x **のシフトする方向が決まります。**

・アセンブル条件

FX_SIGNEXT **を定義すると，引数** x, exp **の符号拡張のため** 3 **命令が追加されます。**

FX_ERRCK **を定義すると，**_fx_errno **にエラーの状態を表す値が設定されます。**

FXLDEXP_ROUND **を定義すると，処理に** 2 **命令が追加され，右ビット・ローテートされた結果が得られます。**

FXLDEXP_ROUND **を未定義にすると，定義時に比べ，返却値の精度が劣ります。**

このラベルはソース・ファイル中に定義されています。

・実行サイクル数

サイクル数	引数
9	x = 0
20/18	x != 0, exp < 0（round/no round）
19	x != 0, exp≧0, **オーバフローなし**
21	x != 0, exp≧0, **オーバフローあり**

・誤差

	誤差	
	round	no round
最大	2.328 E-10	4.657 E-10
平均	3.706 E-11	6.318 E-11
中間	0	0

		0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	round	82.80	97.90	100	100	100	100
	no round	77.90	85.70	94.60	100	100	100

3.36　lfxlog2

[書　式]

#include <fxlib.h>

long fixed　lfxlog2(long fixed x);

[機　能]

底を 2 **とする引数** x **の対数を求める関数です。**

[制　限]

0.5≦**引数** x＜1 **の範囲を設定してください。**

引数 x＜0.5 **または，**1≦**引数** x **を設定した場合，**FX_ERRCK **が定義されていないと，返却値は不定になります。**

[プログラマーズ・ノート]

引数　　**：**R0H/L

返却値　　**：**R0H/L

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：なし

関数内部では，2 **つの多項式の比により** $\log_2 x$ **を演算しています。**

$$\log_2(x) = \frac{a + cx + ex^2 + gx^3 + ix^4}{1 + bx + dx^2 + fx^3 + hx^4}$$

・アセンブル条件

FX_SIGNEXT **の定義は必要ありません。**

FX_ERRCK **を定義すると，引数** x＜0.5 **または，**1≦**引数** x **の場合，**_fx_errno **に** FX_ERR_ARGS **が設定され，返却値は** 0 **となります。**

・実行サイクル数

サイクル数	引数
12	x<0.5（エラー）
244	0.5≦x＜1

・誤差

	誤差
最大	4.419 E-10
平均	1.286 E-10
中間	5.892 E-11

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率(%)	50.25	86.5	99.22	100	100	100

3.37　lfxmod

[書　式]

#include <fxlib.h>

long fixed　　lfxmod(long fixed x, long fixed y);

[機　能]

x/y **の余りを求める関数です。**

x/y **の余りは，**q**（商の整数部）を用い，**x－q×y **で求められます。**

例　x=0.25, y=0.1 の場合

x/y=2.5 **で，この商の整数部は** 2 **です。**

余りは，0.25－2×0.1=0.05 **です。この値がこの関数の返却値になります。**

例　x=0.3, y=0.1 の場合

返却値は 0 **です。**

例　x=0.2, y=0.3 の場合

返却値は 0.2 **です。**

[制　限]

引数 y **に** 0 **を設定しないでください。**

[プログラマーズ・ノート]

引数　　**：**R0H/L(x)

　　：R1H/L(y)

返却値　　**：**R0H/L(x mod y)

使用レジスタ　：R0, R1, R2, R3, R4

呼び出し関数　：なし

内部アルゴリズムは x **と** y **の大きさを調べます。**|x| = |y|**の場合，**0 **が返却値です。**|x|＜|y|**の場合，**x **が返却値です。**|x|＞|y|**の場合，余りが返却値です。**

- **アセンブル条件**

  FX_SIGNEXT **を定義すると，引数の符号拡張を行うための４命令が追加されます。**

  FX_ERRCK **を定義すると，関数** fxmod **は** 0 **を返し，引数** y=0 **で，**_fx_errno **に** FX_ERR_DIV0 **が設定されます。**FX_ERRCK **を定義すると，引数** y=0 **で，返却値は不定になります。**

・実行サイクル数

サイクル数	引数
12	y = 0（エラー）
18	|x|＜|y|
19	|x| = |y|
35~66	|x|＞|y|

・誤差

誤差はありません。

3.38　lfxpow

[書　式]

#include < fxlib.h>

accum　lfxpow(long fixed x, long fixed y);

[機　能]

x^y(x の y 乗)を求める関数です。

－1.0≦y＜1.0 の範囲なので，引数 x の分数べきが求められます。たとえば，y=0.5 の場合，返却値は $x^{0.5}$（x の平方根）です。y が正の場合，0 - 1 までの値になります。y が負の場合，1 より大きい値になります。そのため，返却値は accum 型です。

x，y の設定によって，最大値（255.999...）を越える場合があります。これは，x の値が小さく，y の値が負に大きい場合に発生します。たとえば，x=0.001，y=－0.9 の場合，x^y は約 501.2 となり，accum 型として表現することができません。そのような場合，FX_ERRCK の定義に関係なく，最大値（0x7FFFFFFFFF または 255.999...）が返却値になります。

[制　限]

0≦x＜1.0

x^y＜256.0　　つまり ln(x)×y＜5.545　　5.545 = ln(256)

引数 x は，必ず正の数値にしてください。負の数値を引数にすると，結果は複素数になります。x が負の数値で FX_ERRCK が定義されていると，_fx_errno に FX_ERR_ARGS を設定し，0 を返却します。引数 y には有効な long fixed 型を設定してください。

このとき倍数 y は約 5.545(ln(256))より小さい値にしてください。y を 5.545 以上の値に設定すると，x^y は 256 より大きい値になります。

ln(x)×y≧ln(256)の場合，最大可能数値（0x7FFFFFFFFF または 255.999...）が返却値になります。FX_ERRCK が定義されている場合，_fx_errno に FX_ERR_RANGE が設定されます。

[プログラマーズ・ノート]

引数　　：R0H/L(x)

　　：R1H/L(y)

返却値　　：R0H/L(x^y)

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：fxlog, lfxexp

関数内部では，次の演算を行います。

$$x^y = (e^{\ln(x)})^y = e^{(\ln(x)*y)}$$

まず，x **を引数とする関数** fxlog **の返却値に** y **を積算し，**e **の乗数を求めます。その値を関数** lfxexp **に渡し，その演算結果がこの関数の返却値になります。**

・アセンブル条件

FX_SIGNEXT **を定義すると，引数の符号拡張を行う** 4 **命令が追加されます。**

FX_ERRCK **を定義すると，引数** x **は負の値の場合，**_fx_errno **に** FX_ERR_ARGS **が設定されます。**

また，ln(x)*y **の値が約** 5.545 **以上の場合（オーバフローが原因となる），**_fx_errno **に** FX_ERR_RANGE **が設定されます。**FX_ERRCK **の設定に関係なく正の最大値が返却されます。**

・実行サイクル数

サイクル数	引数
12	x＜0（エラー）
14	x = 0
470-540	0＜x, 0＜y
444-514	0＜x, y = 0
444-547	x＞0, y＜0, 0≦ln(x)×y＜5.545
346-390	x＞0, y＜0, 5.545≦y×ln(x)（エラー）

・誤差

	誤差
最大	4.522 E-8
平均	2.954 E-10
中間	－2.353 E-13

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率(%)	35.67	65.69	83.25	90.61	93.84	95.28

この関数は，内部で 2 **つの関数を使用しているため，誤差はそれらの関数に依存します。引数** x **が** 0 **に近似（**x≠0**）し，引数** y **が負に大きい場合（**x^y＜256.0**）に誤差は最大となります。**

3.39　lfxrand

[書　式]

#include < fxlib.h>

long fixed　　lfxrand(void);

[機　能]

疑似乱数を得る関数です。

返却値は，0x00000001(4.65 E-10) - 0x7FFFFFFE(0.999...)**の** long fixed **型で，整数値の** 1-2147483646($2^{31}-2$) **に相当します。返却値の乱数は，関数** lfxsrand **の引数（**seed **値）の設定に依存します。たとえば，関数** lfxsrand **の引数に** 0x00000001 **を設定すると，初回返却値は，**0x000041A7 **になります。**

乱数の数列は 2147483646 **を** 1 **周期とし，同一値が発生することはありません。**

[制　限]

引数はありません。

[プログラマーズ・ノート]

引数　　：なし

返却値(指数)　：R0H/L

使用レジスタ　：R0, R1, R2, R3

呼び出し関数　：なし

関数内部では，次に示す式で求められます。

$$rand_{i+1} = (rand_i \times a) \bmod m$$

この場合，乗数 a **は** 16807 **注，係数** m **は** 2147483647($2^{31}-1$)**注です。**

係数 m **は，**$2^{31}-1$ **で，**2^{31} **より小さい最大素数になります。**

注　Stephen K. Park & Keith W. Miller : Random Number Generators: Good Onews Are Hard To File , Communications of the ACM, October 1988, Volume 31, Number 10. (1988)

・アセンブル条件

FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

FXRAND_CKRAND **は，現行では定義されていません。定義されている場合，乱数が生成されるたびに** seed **値をチェックする命令が追加されます。** seed **値が** $0<seed<2^{31}-1$ **の範囲でなければならないため，**seed **値が設定されるとき，または乱数が生成されるたびにチェックが行われます。**

seed **値の設定は，3.23　fxsrand を参照してください。**

・実行サイクル数

サイクル数	引数	備考
23	seed **値のチェックなし（**LFXRAND_CKRAND **未定義）**	**ディフォールト**
24	seed **値のチェックあり（**LFXRAND_CKRAND **定義）**	－

・誤差

誤差はありません。

3.40　lfxrexp

[書　式]

```
#include < fxlib.h>
long fixed          lfxrext( accum x, int *eptr );
```

[機　能]

引数 x を 0.5≦|x|≦1.0 の範囲に正規化した値を求める関数です。
eptr によって示される YRAM に x を戻すために必要な，底を 2 とする指数を格納します。返却値は，long fixed 型の値に丸め処理されます。関数 fxrexp の返却値は fixed 型です。

[制　限]

引数 eptr には有効な Y メモリのアドレスを設定してください。

[プログラマーズ・ノート]

引数　　：R0(x)
　　：R1H(eptr)
返却値(指数)　：R0H/L(正規化された x)
　　：*eptr(指数)
使用レジスタ　：R0, R1, R2, DP5
呼び出し関数　：なし

関数内部では，x の値により 2 つの処理が実行されます。−1≦x＜1 の場合，EXP 命令が使用され，左に数ビット・シフトします。x＜−1，1≦x の場合，x を右に 8 ビット・シフトし，EXP 命令を使用します。それから，8 をひくことによってその値を調整します。返却値は，x を適切なビット数で右シフトし，long fixed 型に調整されます。

・アセンブル条件

FX_SIGNEXT および FX_ERRCK の定義は必要ありません。
LFXREXP_ROUND シンボルを定義すると，この関数の返却値に丸め処理を行う 4 命令が追加されます（ディフォールト設定）。その結果，精度が向上します。

・実行サイクル数

サイクル数	引数	
13	−1≦x＜1	
17	x＜−1 または x≧1	round
13		no round

・誤差

	誤差	
	round	no round
最大	2.328 E-10	4.638 E-10
平均	3.424 E-12	3.424 E-12
中間	0	0

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	99.52	99.87	100	100	100	100

3.41　lfxsin

[書　式]

#include < fxlib.h>

long fixed　　lfxsin(long fixed x);

[機　能]

引数 x **の正弦を求める関数です。**

引数－π/2～π/2 **の範囲は** long fixed **型として表現できないため，引数** x **に** 2/π**を積算してください。この場合，関数** lfxsin **は積算前の引数** x **の正弦を返します。**

[制　限]

引数および返却値で 1.0 **を表現できないため，プログラムでチェックを行う必要があります。**

[プログラマーズ・ノート]

引数　　：R0H/L(x)

返却値　　：R0H/L(sin(π/2 * x))

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：なし

関数内部では，次の演算を行います。

$$\sin(x) = c_1x + c_2x^3 + c_3x^5 + c_4x^7 + c_5x^9 + c_6x^{11}$$

・アセンブル条件

FX_SIGNEXT **を定義すると，符号拡張のための** 2 **命令が追加されます。**

FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
109	すべての引数

・誤差

	誤差
最大	4.850 E-10
平均	1.385 E-10
中間	－2.919 E-12

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	47.62	82.30	97.33	99.98	100	100

3.42　lfxsinh

[書　式]

#include < fxlib.h>

accum　　lfxsinh(accum x);

[機　能]

引数 x **の双曲正弦を求める関数です。**

[制　限]

|x|＜6.24 (arcsinh(256))

引数 x **に約** 6.24(arcsinh(256))**以上の値が与えられると，双曲正弦の** sinh(x)**の結果は** 256 **以上の値になり，この場合，**accum **型の正の最大値（**0x7FFFFFFFFF **つまり** 255.999...**）が返却されます。**

[プログラマーズ・ノート]

引数　　**：**R0(x)

返却値(指数)　**：**R0(sinh)

使用レジスタ　**：**R0, R1, R2, R3, R4, DP1

呼び出し関数　**：**lfxexp

関数内部では，次の式で演算を行います。

$$\sinh(x) = \frac{e^{x} - e^{-x}}{2} = \frac{e^{x}}{2} - \frac{e^{-x}}{2}$$

$$\frac{e^{x}}{2} = e^{(x-\ln(2))} \qquad \frac{e^{-x}}{2} = \frac{1}{2 \times e^{x}} = \frac{1}{4 \times \frac{e^{x}}{2}} = \frac{0.25}{e^{(x-\ln(2))}}$$

$$\sinh(x) = e^{(x-\ln(2))} - \frac{0.25}{e^{(x-\ln(2))}}$$

・アセンブル条件

FX_SIGNEXT **の定義は必要ありません。**

FX_ERRCK **を定義すると，引数にエラーがない場合，**_fx_errno **をクリアします。引数** x **が約** 6.24 **以上の場合，**_fx_errno **に** FX_ERR_RANGE **が設定されます。これらの条件は関数** lfxexp **関数で定義されている場合のみ有効です。**

・実行サイクル数

サイクル数	引数
15	x = －256
37	6.238≦|x|＜256
265	1.693≦|x|＜6.238(ln(2)＋1≦|x|＜ln(512))
232	0.693≦|x|＜1.639(ln(2)≦|x|＜ln(2)＋1)
258	0≦|x|＜0.639(0≦|x|＜ln(2))

・誤差

引数 x **の値が** 1.0 **より小さい場合，約**±1.5LSB(±1.5×2^{-31})**の誤差が生じます。**1≦x＜5.545 **の場合，**(x－ln(2))**が増すにつれ精度が落ちます。**

・x＜1.0

	誤差
最大	8.088 E-10
平均	2.120 E-10
中間	－4.801 E-12

	0.25	0.50	0.75	1.00	1.25	1.50	1.75
最下位ビットの誤差率（%）	32.08	60.48	80.62	93.22	98.65	99.85	100

・－6.25≦x≦6.25

	誤差
最大	4.228 E-8
平均	2.832 E-9
中間	1.139 E-11

・そのほかの範囲

	誤差					
	1≦x＜2	2≦x＜3	3≦x＜4	4≦x＜5	5≦x＜6	6≦x＜6.25
最大	1.326 E-9	2.037 E-9	5.781 E-9	1.592 E-8	4.297 E-8	4.228 E-8
平均	2.684 E-10	5.261 E-10	1.083 E-9	3.095 E-9	8.432 E-9	1.403 E-8
中間	1.564 E-10	4.402 E-10	4.237 E-10	1.853 E-9	5.463 E-9	1.167 E-8

・備考

現行より精度を上げる，または処理速度を向上するためには，関数 lfxexp の変更が不可欠になります。

処理速度を向上する場合，この関数を変更する必要があります。$0.25/e^{(x-\ln(2))}$の除算は，1 ループ内で 4 ビットが対象となる処理を行いますが，今よりも多いビットを処理することで，処理速度を向上することができます。ただし，演算精度は変わりません。

3.43　lfxsqrt

[書　式]

#include < fxlib.h>

long fixed　　lfxsqrt(long fixed x);

[機　能]

引数 x の正の平方根を求める関数です。

[制　限]

引数 x≧0 を設定してください。

[プログラマーズ・ノート]

引数　　：R0H/L(x)

返却値　　：R0H/L(sqrt)

使用レジスタ　：R0, R1, R2, R3, R4, DP1

内部アルゴリズム[注]は，引数を 0.25 - 1 の範囲に正規化します。そして，シフト量を保存します。次に示す演算で，最初の評価を行います。

注 Mikami, Kobayashi & Yokoyama : A New DSP-Oriented Algorithm for Calculation of the Square Root Using a Nonlinear Digital Filter , *IEEE Transactions on Signal Processing*, Vol.40, No.7, July 1992.(1992)

$$Beta = ax^2 + bx + c$$
$$sqrt_0 = dx + e$$

ベータ値(0.5/sqrt(x))の評価は，次の式に変形されます。

$$Beta' = 2 \times Beta \times \left(\frac{3}{4} - \frac{x}{Beta^2} \right)$$

平方根の評価は，Beta を用いて次の演算を 3 回行います。

$$sqrt_{i+1} = sqrt_i + Beta' \times (x - sqrt_i^2)$$

この結果，シフト値/2 だけ右シフトし，正規化され，丸め処理が行われます。このアルゴリズムは，32 ビット演算に拡張されています。

・アセンブル条件

FX_ERRCK が定義されていると，引数はチェックされます。引数が 0 未満の場合，_fx_errno に FX_ERR_ARGS が設定され，0 が返却されます。引数が 0 以上の場合，_fx_errno に 0 が設定されます。FX_ERRCK が定義されていない場合，返却値は不定になります。

FX_SIGNEXT を定義すると，引数を符号拡張するための 2 命令が追加されます。この関数に定義されている ROUND_RSLT は，結果に対し，正規化や丸め処理といった処理方法を定義するシンボルです。このシンボルは LFXSQRT.ASM ファイル内に定義されています。その値と処理内容は次のとおりです。

値	速度	精度	命令数	備考
0	速い	欠ける	2	－
1	普通	普通	6	－
2	遅い	正確	12	ディフォールト

・実行サイクル数

サイクル数	引数
10	x＜0 （エラー）
13	x = 0
60	x＞0

・誤差

	誤差
最大	3.733 E-10
平均	1.169 E-10
中間	－6.029 E-13

	0.25	0.50	0.75	1.00	1.25	1.50
最下位ビットの誤差率（%）	49.72	99.90	99.997	100	100	100

3.44　lfxsrand

[書　式]

```
#include < fxlib.h>
void    lfxsrand( long fixed seed );
```

[機　能]

疑似乱数の初期値を設定する関数です。

引数（seed 値）を 0x00000001 に設定してください。引数（seed 値）をほかの値に設定すると，その値が疑似乱数の初期値となります。

[制　限]

0x00000000＜seed＜0x7FFFFFFF の値にしてください。関数 lfxrand の内部アルゴリズムは，0x7FFFFFFF(2^{31}－1)を係数として使用しています。

[プログラマーズ・ノート]

引数　　：R0H/L(seed)
返却値　　：なし
使用レジスタ　：R0, R1
呼び出し関数　：なし

lfxsrand ルーチンは，LFXRAND.ASM ファイル内の関数 lfxrand とともに記述されています。

・アセンブル条件

FX_SIGNEXT および FX_ERRCK の定義は必要ありません。

LFXRAND_CKSEED を定義すると（ディフォールト設定時），引数 seed 値について範囲のチェックが行われます。範囲を越えた場合，0x00000001 に設定します。

この関数の引数が，必ず範囲内であると保証している場合，または乱数生成時にチェックする LFXRAND_CKSEED が定義されている場合は，LFXRAND_CKSEED の定義は必要ありません。

・実行サイクル数

サイクル数	引数	備考
13	LFXRAND_CKSEED 定義	ディフォールト
6	LFXRAND_CKSEED 未定義	－

・誤差

返却値はありません。

3.45　lfxtan

[書　式]

```
#include < fxlib.h>
accum   lfxtan( long fixed x );
```

[機　能]

引数 x の正接を求める関数です。

引数－π/2～＋π/2 の範囲は long fixed 型として表現できないため，引数に 2/π(0.6366..)を積算してください。この場合，関数 lfxtan は積算前の引数 x の正接を返します。

[制　限]

－0.9975 (0x80517CA7)≦x≦0.9975 (0x7FAE8359)

返却値は±π/2 に相当する±1.0 を返せませんので注意してください。この場合，約±1.567 ラジアン（約±89.7748 度）に相当する約±0.9975 を返します。

[プログラマーズ・ノート]

引数　　：R0H/L(x)

返却値　　：R0(tan)

使用レジスタ　：R0, R1, R2, R3, R4, DP1

呼び出し関数　：lfxsin, lfxcos

関数内部では，次の演算を行います。

$$\tan(x) = \frac{\sin(x)}{\cos(x)}$$

・アセンブル条件

FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

サイクル数	引数
285	|x|＜0.5
307	0.5≦|x|≦0.9975
238	0.9975＜|x|＜1

・誤差

関数内部の演算 sin(x)/cos(x)**では，**cos(x)**が値** 0 **に近似すると誤差が大きくなるなどの理由から，誤差率は次の条件で表現しています。**

	誤差	
	|tan|≧1	|tan|＜1
最大	2.138 E-5	9.730 E-10
平均	4.085 E-8	1.022 E-10
中間	0	0
最大/tan	1.152 E-7	1.069 E-6
平均/tan	7.432 E-10	9.651 E-10

	1	2	3	4	5	6	7
最下位ビットの誤差率（%）	60.03	71.98	77.08	79.95	81.93	83.37	84.32

3.46　lfxtanh

[書　式]

```
#include < fxlib.h>
long fixed     lfxtanh( accum x );
```

[機　能]

引数 x **の双曲正接を求める関数です。**

[制　限]

引数 x **は** accum **型に設定してください。**

[プログラマーズ・ノート]

引数　　：R0(x)

返却値　　：R0H/L(tanh)

使用レジスタ　　：R0, R1, R2, R3, R4, DP1

呼び出し関数　　：lfxexp

tanh(x)=－tanh(－x)**になるため，内部で引数** x **を正の値として扱っています。**

関数内部では，次の演算を行います。

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{e^{x} - \dfrac{1}{e^{x}}}{e^{x} + \dfrac{1}{e^{x}}}$$

引数 x **の値が** 0-**約** 12.5 **間の** e^{x} **を計算する必要があります。**

$e^{12.5}$ **が約** 268,337 **なので，**accum **型を越えてしまいます。そこで，次の式のように** 8 **の指数により，分子と分母の値を小さくします。**

$$\tanh(x) = \frac{e^{x} - \dfrac{1}{e^{x}}}{e^{x} + \dfrac{1}{e^{x}}}$$

$$= [(e^{x} - 1/e^{x})/8^{n}]/[(e^{x} + 1/e^{x})/8^{n}]$$

$$= [(e^{x}/8^{n}) - (1/e^{x} \times 8^{n})]/[(e^{x}/8^{n}) + (1/e^{x} \times 8^{n})]$$

$$= [(e^{x}/8^{n}) - (1/8^{2n})/(e^{x}/8^{n})]/[(e^{x}/8^{n}) + (1/8^{2n})/(e^{x}/8^{n})]$$

$e^{x}/8^{n}$ **で双曲正接を求めます。これを，次に示す形に変換しています。**

$$\frac{e^{x}}{8^{n}} = e^{(x - n*\ln(8))}$$

これにより，e^{x} **の指数を** 0 **から** ln(8)**，つまり約** 2.08 **で求めます。結果は，**0-8 **の動的範囲になります。**0-12.5 **の** x **の範囲の演算を，演算可能な範囲にまで落としました。アルゴリズムは次に示します。**

t = n×ln(8)**の場合，**n **は** t **が正の値となるような** 0≦n＜5 **の最大整数の範囲です。**

t＞ln(8)，x＞12.47 **の場合，**－1 **か** 0.999...**を返します。**

別の方法では，u = lfxexp(t) = e^{t} = $e^{(x-n\times n(8))}$ = $e^{x}/8^{n}$ **を行い，**v = 8^{2n}/u = $8^{2n}/e^{(x-n\times n(8))}$ = $1/(e^{x}\times 8^{n})$**を計算します。**

$$\tanh(x) = (u - v)/(u + v)$$
$$= [e^x/8^n - 1/(e^x \times 8^n)]/[e^x/8^n + 1/(e^x \times 8^n)]$$
$$= [(e^x - 1/e^x)/8^n]/[(e^x + 1/e^x)/8^n]$$
$$= (e^x - 1/e^x)/(e^x + 1/e^x)$$

・アセンブル条件

FX_SIGNEXT **および** FX_ERRCK **の定義は必要ありません。**

・実行サイクル数

関数 lfxtanh **は，**$e^{(x-n\times\ln(8))}$**を得るために関数** lfxexp **を呼び出しています。そのため，この関数の実行時間は，関数** lfxexp **の実行時間に依存しています。次に実行時間を示します。**

サイクル数	引数
42	x = －256
311	x = 0
344	|x| = 1
344	|x| = 2
315	|x| = 3
348	|x| = 4
319	|x| = 5
352	|x| = 6
323	|x| = 7
356	|x| = 8
327	|x| = 9
360	|x| = 10
329	|x| = 11
362	|x| = 12
42	|x|＞12.47 (6×ln(8))

・誤差

lfxtanh **の精度は次の範囲に依存します。**

	誤差		
	－1≦x＜1	－12.5＜x＜12.5	－256≦x＜256
最大	6.806 E-10	6.806 E-10	6.806 E-10
平均	1.512 E-10	1.616 E-10	2.291 E-10
中間	1.2181 E-11	1.911 E-11	1.752 E-11

最下位ビットの誤差率（%）	0.25	0.50	0.75	1.00	1.25	1.50
－1≦x＜1	44.03	78.37	94.93	99.30	99.90	100
－12.5＜x＜12.5	41.48	75.23	92.26	98.51	99.89	100
－256≦x＜256	49.52	51.31	52.11	53.64	100	100

引数 x **が** 12.5 **以上の場合，双曲正接は約** 1.0 **に近似する値になります。この関数の返却値は，**long fixed **型最大値** 0x7FFFFFFF **になります。誤差は** 2^{-31}**，つまり** 1LSB **を生じます。そのため，**－256～256 **の範囲の平均誤差は他の範囲に比べて大きくなることに注意してください。**

3.47　normalize

[書　式]

#include < fxlib.h>

accum　　normalize(accum x, int y);

[機　能]

引数 x **を引数** y **だけシフトする関数です。**

引数 y **が負の場合，符号拡張を伴う右シフトを行い，**y **が正の場合，**LSB **に** 0 **を挿入し，左シフトします。次のように指数関数とともに使用されるとき，関数** normalize **は，**fixed **型，または** long fixed **型で表現される最大値を求めるために使用することができます。**

normalize(x, exponent(x))

－1.0**～**0.999...**の範囲に適した** x **を返します。**

[制　限]

引数 y **のビット** 5 - **ビット** 0 **はシフト・カウント，ビット** 15 **は符号ビットとして使用されます。**40 **以上のシフトを実行すると，返却値は** 0x0000000000 **か，**0xFFFFFFFFFF **を返しますので注意してください。**

[プログラマーズ・ノート]

引数　　**：**R0(x)

　　：R1H(y)

返却値　　**：**R0　　x <<|y|(y=**正**),　x >> |y|(y=**負**)

使用レジスタ　：R0, R1

呼び出し関数　：なし

・アセンブル条件

FX_SIGNEXT **を定義すると，引数** y **の符号拡張を行う命令が追加されます。**

・実行サイクル数

サイクル数	引数
8	y≧0
11	y＜0

・誤差

右にシフトすると，下位ビットがシフト量分だけ失われます。

・備考

マクロ定義を行うと，CALL **命令，**RET **命令の** 4 **サイクルを省くことができます。マクロはユーザが定義してください。**

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：info@lsi.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部
東　京　(03)3798−6106, 6107, 6108
大　阪　(06)6945−3178, 3200, 3208, 3212
広　島　(082)242−5504
仙　台　(022)267−8740
郡　山　(024)923−5591
千　葉　(043)238−8116

第二販売事業部
東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部
東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
前　橋　(027)243−6060
鳥　取　(0857)27−5313
太　田　(0276)46−4014
名古屋　(052)222−2170, 2190
福　岡　(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【NECエレクトロンデバイス　ホームページ】

NECエレクトロンデバイスの情報がインターネットでご覧になれます。　URL（アドレス）　http://www.ic.nec.co.jp/

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］ μPD77016 ファミリ　アプリケーション・ノート　ライブラリ編
（U12021JJ2V0AN00（第 2 版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）　（　　）
ご住所　（　　）
お電話番号　（　　）
お仕事の内容　（　　）
お名前　（　　）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは
NEC 販売員，特約店販売員， その他（　　）

ご協力ありがとうございました。
下記あてに FAX で送信いただくか，最寄りの販売員にコピーをお渡しください。

日本電気(株) NEC エレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6

キリトリ