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

78K/II シリーズ

8ビット・シングルチップ・マイクロコンピュータ

浮動小数点演算プログラム編

μPD78214 シリーズ
μPD78218A シリーズ
μPD78224 シリーズ
μPD78234 シリーズ
μPD78244 シリーズ

資料番号　IEA-686D（第5版）
発行年月　July 1997　N

〔メ　モ〕

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戦略物資等（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，後日変更する場合があります。
○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所有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。
○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。
○この製品は耐放射線設計をしておりません。

本版で改訂された主な箇所

箇　所	内　容
全　般	μPD78P244に関する記述削除

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・コーナーを設けております。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，78K/IIシリーズ製品の機能を理解し，78K/IIシリーズ製品を用いた浮動小数点演算プログラムを設計するエンジニアを対象とします。

78K/IIシリーズ製品

- μPD78214シリーズ　：μPD78212，78213，78214，78P214
  μPD78212(A)，78213(A)，78214(A)，78P214(A)
- μPD78218Aシリーズ：μPD78217A，78218A，78P218A，78218A(A)
- μPD78224シリーズ　：μPD78220，78224，78P224
- μPD78234シリーズ　：μPD78233，78234，78237，78238，78P238
  μPD78234(A)，78238(A)
- μPD78244シリーズ　：μPD78243，78244

目　的 このアプリケーション・ノートは，78K/IIシリーズ製品の浮動小数点演算応用プログラムについてユーザに理解して頂くことを目的とします。なお，掲載のプログラムは例示的に示したものであり，量産設計を対象とするものではありません。

構　成 このアプリケーション・ノートでは，次の内容について説明しています。

- 計算アルゴリズム
- 四則演算
- 関数（数学，座標変換，型変換）
- プログラム・リスト

なお，次の２つのアプリケーション・ノートも別に用意しています。

- 基礎編（IEA-607）
- 応用編（IEA-700）

品質水準　μPD78212(A), 78213(A), 78214(A), 78P214(A)はμPD78212, 78213, 78214, 78P214の, μPD78218A(A)はμPD78218Aの, またμPD78234(A), 78238(A)はμPD78234, 78238の「特別」品質水準品です。

> **このアプリケーション・ノート中の使用例は，一般電子機器用の「標準」品質水準品用に作成してあります。「特別」品質水準品を要求する用途にこのアプリケーション・ノート中の使用例を使用する場合は，実際に使用する各部品および回路について，その品質水準についてご検討のうえご使用ください。**

品質水準とその応用分野の詳細については当社発行の資料「NEC 半導体デバイスの品質水準」(IEI-620)をご覧ください。

応用分野　○標準品

- ●プリンタ
- ●カメラ
- ●タイプライタ
- ●PPC
- ●電子楽器
- ●エアコンなど

○特別品

- ●自動車電装
- ●燃料制御など

関連資料

○78K/IIシリーズ共通資料

資　料　名		資料番号
ユーザーズ・マニュアル　命令編		U10288
SBI　ユーザーズ・マニュアル		IEM-5040
アプリケーション・ノート	基礎編	IEA-607
	応用編	IEA-700
	浮動小数点演算プログラム編	このアプリケーション・ノート
セレクション・ガイド		IF-304
インストラクション活用表		IEM-5101
インストラクション・セット		IEM-5102
開発ツール　セレクション・ガイド		EF-231

○個別資料

●μPD78214シリーズ

品名 / 資料名	μPD78212	μPD78213	μPD78214	μPD78P214
パンフレット	IB-5036			
データ・シート	U12483			IC-7732
ユーザーズ・マニュアル　ハードウエア編	IEM-5119			
モード・レジスタ活用表	IEM-5100			

品名 / 資料名	μPD78212(A)	μPD78213(A)	μPD78214(A)	μPD78P214(A)
データ・シート	IC-8234			IC-8589
ユーザーズ・マニュアル　ハードウエア編	IEM-5119			
モード・レジスタ活用表	IEM-5100			

●μPD78218Aシリーズ

品名 / 資料名	μPD78217A	μPD78218A	μPD78P218A	μPD78218A(A)
パンフレット	IF-288			
データ・シート	IC-8132	IC-8131	IC-8133	IC-8685
ユーザーズ・マニュアル　ハードウエア編	IEU-755			
特殊機能レジスタ活用表	IEM-5532			

●μPD78224シリーズ

品名 / 資料名	μPD78220	μPD78224	μPD78P224
パンフレット	IB-5011		
データ・シート	IC-5457		IC-7757
ユーザーズ・マニュアル　ハードウエア編	IEU-5019		
特殊機能レジスタ活用表	IEM-999		

●μPD78234シリーズ

品名 / 資料名	μPD78233	μPD78234	μPD78237	μPD78238	μPD78P238
パンフレット	IF-207				
データ・シート	IC-7902		IC-8348	IC-8023	IC-8030
ユーザーズ・マニュアル　ハードウエア編	IEU-718				
特殊機能レジスタ活用表	IEM-5515				

品名 / 資料名	μPD78234(A)	μPD78238(A)
パンフレット	—	
データ・シート	IC-8416	
ユーザーズ・マニュアル　ハードウエア編	IEU-718	
特殊機能レジスタ活用表	—	

●μPD78244シリーズ

品名 / 資料名	μPD78243	μPD78244
パンフレット	IF-6339	
データ・シート	IC-8355	IC-8070
ユーザーズ・マニュアル　ハードウエア編	IEU-747	
特殊機能レジスタ活用表	IEM-5528	

目　次　要　約

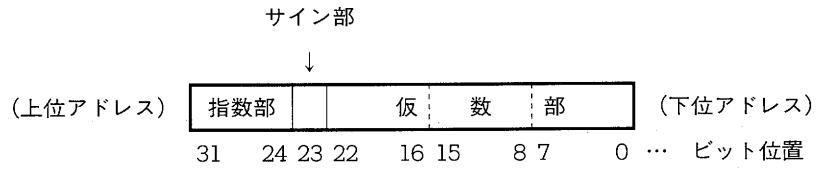
目　　次

第１章　概　　要

１.１　浮動小数点形式

この演算プログラムでは，浮動小数点数を４バイトで表現します。内訳は次のとおりです（下図参照)。

- ●仮数部　：23ビット
- ●指数部　：８ビット
- ●サイン部：１ビット



この形式の数値は，次のようになります。

$$(-1)^{(\text{サイン部値})} \times (\text{仮数部値}) \times 2^{(\text{指数部値})}$$

次に，各部の詳細について説明します。

(1) 仮 数 部

仮数部は絶対値で表現され，仮数部のビット位置22－0が２進数の小数点第２位-第24位に相当します。

また，仮数部の値は０の場合を除いて常に0.５－１の範囲になるように指数部の値が調整されます。小数点第１位(0.5の値を意味する)は常に１となり，この形式では省略した形で表現されています。

備考　最上位ビット（MSB）を常に１にする操作を正規化と言います。

(2) サイン部

正数の場合は０，負数の場合は１とします。

(3) 指 数 部

底2の指数を1バイトの整数型（負の場合2の補数）で表し，この値に，さらに，80Hのバイアスを加えた値を用いています。これらの関係は，具体的には下表のようになります。

指数部（16進）	指数部の値
FF	127
FE	126
⋮	⋮
81	1
80	0
7F	−1
⋮	⋮
01	−127

注意　指数部が0のときのみ，浮動小数点値は0を表します。この場合，仮数部，およびサイン部は無視されます。

(4) 数値の表現範囲

浮動小数点値Xは，次の範囲の値と，“0”を表現することが可能です。

$$約2.9387\times10^{-39}\leqq|X|\leqq約1.7014\times10^{38}$$

1.2　提供関数

この演算プログラムの提供関数について概説します。

提供関数として，次の４つを用意しています。

- 四則演算
- 数学関数
- 座標変換関数
- 数値型変換関数

提供関数	関　　数	関数名
四則演算	加算	LADD
	減算	LSUB
	乗算	LMLT
	除算	LDIV
数学関数	三角関数 { sin関数	LSIN
	三角関数 { cos関数	LCOS
	三角関数 { tan関数	LTAN
	自然対数関数（log）	LLOG
	常用対数関数（log10）	LLOG10
	指数関数（底＝e）	LEXP
	指数関数（底＝10）	LEXP10
	べき乗　（ab）	LPOW
	平方根	LSQRT
	逆三角関数 { arcsin関数	LASIN
	逆三角関数 { arccos関数	LACOS
	逆三角関数 { arctan関数	LATAN
	双曲線関数 { sinh関数	LHSIN
	双曲線関数 { cosh関数	LHCOS
	双曲線関数 { tanh関数	LHTAN
	絶対値	LABS
	逆数	LRCPN
座標変換関数	極座標→直交座標への変換	POTORA
	直交座標→極座標への変換	RATOPO
数値型変換関数	文字列→浮動小数点値への変換	ATOL
	浮動小数点値→文字列への変換	LTOA
	２バイト整数型→浮動小数点値への変換	FTOL
	浮動小数点値→２バイト整数型への変換	LTOF
	IEEE-754形式→78K/II形式への浮動小数点形式変換	LTO78
	78K/II形式　→IEEE-754形式への浮動小数点形式変換	LTOIE

1.3 ファイル構成

この演算プログラムは，次の４種類のファイルにより構成されています。

- ●オブジェクト・ソース・ファイル[注]：26
- ●共通サブルーチン・ファイル[注]　：２
- ●共通データ・ファイル　：１
- ●インクルード・ファイル　：４

注 オブジェクト・ソース・ファイルは，構造化アセンブラ言語で記述されています。

(1) オブジェクト・ソース・ファイル

ソース・ファイル名	提　供　関　数
LFLT1　.SRC	四則演算
LSIN　.SRC	sin関数
LCOS　.SRC	cos関数
LTAN　.SRC	tan関数
LLOG　.SRC	自然対数関数
LLOG10.SRC	常用対数関数
LEXP　.SRC	指数関数（底＝e）
LEXP10 .SRC	指数関数（底＝10）
LPOW　.SRC	べき乗関数
LSQRT　.SRC	平方根
LASIN　.SRC	arcsin関数
LACOS .SRC	arccos関数
LATAN .SRC	arctan関数
LHSIN　.SRC	sinh関数
LHCOS .SRC	cosh関数
LHTAN .SRC	tanh関数
LABS　.SRC	絶対値
LRCPN .SRC	逆数
POTORA.SRC	極座標→直交座標への変換
RATOPO.SRC	直交座標→極座標への変換
ATOL　.SRC	文字列→浮動小数点値への変換
LTOA　.SRC	浮動小数点値→文字列への変換
FTOL　.SRC	２バイト整数型→浮動小数点値への変換
LTOF　.SRC	浮動小数点値→２バイト整数型への変換
LTO78　.SRC	IEEE-754形式→78K/II形式への浮動小数点形式変換
LTOIE　.SRC	78K/II形式→IEEE-754形式への浮動小数点形式変換

(2) 共通サブルーチン・ファイル

ソース・ファイル名	提　供　機　能
LFLT2 .SRC	多項式計算関数
LLD　.SRC	浮動小数点レジスタ間ロード／置換関数

(3) 共通データ・ファイル

ソース・ファイル名	定　義　内　容
DFLT .SRC	浮動小数点レジスタ定義

(4) インクルード・ファイル

ソース・ファイル名	定　義　内　容
EQU　.INC	EQU定義
REF1　.INC	浮動小数点レジスタ参照宣言
REF2　.INC	浮動小数点レジスタ間ロード／置換関数使用宣言
ASCII .INC	アスキーコード定義

各関数の使用に当たって，リンケージをとるべきオブジェクト・モジュール・ファイルについては，各関数の解説の項に記述しています。

備考　当社の78K/IIシリーズ用アセンブラ・パッケージには，ライブラリアンが付属しており，このライブラリアンを使用してライブラリ・ファイルを作成することができます。上記の全プログラムをライブラリ・ファイルに登録することで，リンク時にそのライブラリ・ファイルを指定するだけで必要なモジュールを自動的にリンクすることができるので便利です。
なお，ライブラリ・ファイルへの登録順序には，特に制限はありません。

1.4 プログラムの特性

1.4.1 プログラムの配置アドレス

この演算プログラムで使用するワーク・エリアはすべて，ショート・ダイレクト・アドレシング・エリアに浮動小数点レジスタとして定義しています（実際には，単にグローバル変数というだけですが，使用方法が浮動小数点演算用に限定されるため，ここではレジスタと呼びます）。

これら浮動小数点レジスタの領域確保はDFLT（リンケージ対象オブジェクト・モジュール）で行っています。それ以外のモジュールは浮動小数点レジスタを外部参照していますので，ROM内に配置される必要がありますが，その他の制限はありません。

1.4.2 リエントラント性

リエントラント性は持っていません。

多重タスクの処理系では，1タスクのみがこの関数を呼ぶことができるようにするなどの資源管理が必要になります。

1.4.3 スタック

各関数の解説のところでスタックの最大消費サイズを示しています。各関数の使用に当たって最低限，示されたサイズ分のスタックを用意する必要があります。

1.4.4 レジスタ，フラグなどの保存

汎用レジスタ，およびフラグの保存はしていません。必要であれば，この関数を呼び出す前に，スタックなどに退避しておいてください。

1.4.5 レジスタ・バンク

この演算プログラムでは，レジスタ・バンクの選択命令は使用していません。呼び出された時点で選択されているレジスタ・バンクを用いています。

1.4.6 処理時間

各関数に記載されている処理時間は，μPD78213を外部クロック12 MHz（内部システム・クロックf_{CLK}=6 MHz）で動作させ，スタック・エリアを0FE00H-0FE20Hにとった場合の処理時間です。スタック・エリアの場所やROM内蔵品で，内部ROMからの高速フェッチ機能を使用した場合，実行時間は変化します。

1.5 データの受け渡し方法

1.5.1 パラメータおよび返値

先に述べた浮動小数点レジスタにより受け渡しを行います。

この演算プログラムでは統一して，被演算値と返値の格納レジスタを共通にしています。その意味からこのマニュアル中では以後，被演算値をデスティネーション（d）と呼び，演算値をソース(s) と呼びます。

パラメータおよび返値の設定の詳細については，各関数の解説のところで説明しています。

浮動小数点レジスタは，この演算プログラムのワーク用のレジスタでもあります。したがって，汎用レジスタ同様，その内容は保存されません。

1.5.2 演算結果の報告

終了状態を細かく分けると，次の５種類になります。

(1) 正常終了
(2) アンダフロー
(3) オーバフロー
(4) 虚数化
(5) 演算不能（log（－1）など）

(2)のアンダフローは返値０を正常終了として返します。

終了状態の報告は，エラー((3)，(4)，(5)) 状態を区別しません。正常終了か異常終了かだけを報告します。

終了状態	Aレジスタ内容	CYフラグ
正常終了	0	0
異常終了	81H	1

1.6 浮動小数点レジスタ

このアプリケーション・ノート中では，演算プログラムで使用するワーク・エリアを浮動小数点レジスタと呼びます。ここでは，各レジスタの役割について簡単に説明します。

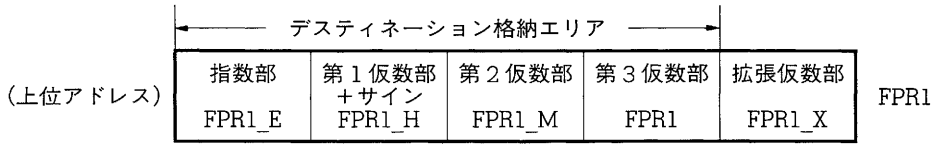
下の図で，1つの [　] が1バイトを，また，左側が上位アドレスを表します(以降，このアプリケーション・ノート中では共通)。

1.6.1 浮動小数点レジスタ1（FPR1）

ほとんどの関数で，デスティネーションの格納用に使用するレジスタです。

また，演算の中心となるレジスタでもあります。

FPR1は，下図に示す連続した5バイトのレジスタです。



レジスタは5バイトで構成されており，その1つ1つに広域名が付けられています。

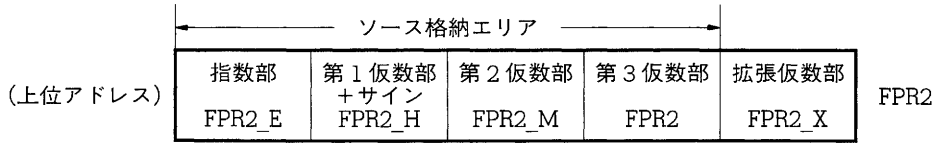
FPR1-FPR1_Eの4バイトが，デスティネーションを格納するエリアです。

FPR1_Xは，仮数部を内部的に1バイト拡張して，第4仮数部（小数点第25位-第32位）として計算するためのエリアです。

1.6.2 浮動小数点レジスタ2（FPR2）

ソースの格納用に使用するレジスタです。

FPR2は，下図に示す連続した5バイトのレジスタです。



レジスタの構成は，FPR1と同様です。

1.6.3 浮動小数点レジスタ3-5（FPR3，4，5）

数学関数などで，一時的なワークとして使用するレジスタです。

FPR1，FPR2同様，連続した5バイトのレジスタです。

(1) 浮動小数点レジスタ3（FPR3）

（上位アドレス）	指数部	第1仮数部	第2仮数部	第3仮数部	拡張仮数部 FPR3 FPR3_1	FPR3

(2) 浮動小数点レジスタ4（FPR4）

（上位アドレス）	指数部	第1仮数部	第2仮数部	第3仮数部 FPR4_2	拡張仮数部 FPR4 FPR4_1	FPR4

(3) 浮動小数点レジスタ5（FPR5）

（上位アドレス）	指数部	第1仮数部	第2仮数部	第3仮数部 FPR5_2	拡張仮数部 FPR5 FPR5_1	FPR5

注意　FPR3，FPR4，FPR5レジスタは，すべての関数で使用するわけではありません。各関数で何番レジスタまで使用するかは，関数解説の項で説明しています。

1.6.4 浮動小数点レジスタ間のロード／置換

数学関数や型変換関数では，頻繁にレジスタのロード／ストア／置換を行います。そのため，浮動小数点レジスタ間のロード／ストア関数，および置換関数を用意しています。

ロード／ストア関数および置換関数の一覧を示します。

関　数　名	機　　能
LLD21, LLD21X	1番レジスタ内容を2番レジスタへロード
LLD31, LLD31X	1番レジスタ内容を3番レジスタへロード
LLD41, LLD41X	1番レジスタ内容を4番レジスタへロード
LLD51, LLD51X	1番レジスタ内容を5番レジスタへロード
LLD52	2番レジスタ内容を5番レジスタへロード
LLD13	3番レジスタ内容を1番レジスタへロード
LLD23, LLD23X	3番レジスタ内容を2番レジスタへロード
LLD14, LLD14X	4番レジスタ内容を1番レジスタへロード
LLD24, LLD24X	4番レジスタ内容を2番レジスタへロード
LLD15	5番レジスタ内容を1番レジスタへロード
LLD25, LLD25X	5番レジスタ内容を2番レジスタへロード
LLD1C, LLD1CX	定数データを1番レジスタへロード
LLD2C, LLD2CX	定数データを2番レジスタへロード
LXC13, LXC13X	1番レジスタと3番レジスタ内容を置換
LXC14X	1番レジスタと4番レジスタ内容を置換
LXC15, LXC15X	1番レジスタと5番レジスタ内容を置換

注意　関数名の最後が，“X”で終わっているものは，拡張仮数部を含めたロード／置換関数です。

第２章　計算アルゴリズム

ここでは，計算の基本となるアルゴリズムについて簡単に説明します。

2.1　関数の展開方法

３種類の展開法を使用します。

- ●平方根　　：Newton-Raphson法
- ●逆三角関数：最良近似法
- ●その他　　：Taylor展開法

2.2　丸め方法

０に向かって丸めを行います。

2.3　桁落ち防止方法

Taylor展開の展開多項式が階差級数となった場合，被演算子の値の範囲によっては極端な桁落ちが起こる場合があります。このような桁落ちを防止するために，この演算プログラムでは，展開式の第１項から第N項までの各項の値が，単調かつ急速に０に近付くような展開式を使用し，さらに仮数部を32ビットとしています。

2.4　多項式加算／乗算時の誤差

有効桁24ビットの数体系で，０に向って丸めを行った８項の多項式の加算を行った場合，最大４ビットの誤差が含まれます。また，x^8を７回の乗算で求めるのと同様の誤差が含まれます。

このような四則演算の誤差（加算と乗算）を少なくするために，この演算プログラムは，内部では仮数部を32ビット（拡張仮数部を８ビット付加）として計算しています。

第３章　四則演算

四則演算関数として，次の４つを用意しています。

(1) 浮動小数点加算（LADD）

FPR1の値を被加数，FPR2の値を加数として加算を行います。

(2) 浮動小数点減算（LSUB）

FPR1の値を被減数，FPR2の値を減数として減算を行います。

(3) 浮動小数点乗算（LMLT）

FPR1の値を被乗数，FPR2の値を乗数として乗算を行います。

(4) 浮動小数点除算（LDIV）

FPR1の値を被除数，FPR2の値を除数として除算を行います。

演算結果は，FPR1に格納されます。

3.1　浮動小数点加算（LADD）

(1) 処理内容

FPR1の値をx，FPR2の値をyとして，x+yをFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1

(3) 消費スタック・サイズ

2（LADDからの戻り番地のみ）

(4) 使用レジスタ

FPR1，FPR2

(5) 処理時間（内部システム・クロック：6 MHz）

平均：142 μs

最大：296 μs（0.5+（−0.500000060））

(6) 処理手順

(a) xまたは，yが０の場合，０でない数を結果として演算を終了します。

(b) 指数部差が32以上の場合，大きい方を結果として演算を終了します。

(c) 指数部の小さい方の仮数部を，大きい方に合わせて桁調整します。

(d) 仮数部の加減算を行います。元の符号が同じ場合は加算し，不一致の場合は減算します。

(e) 加算時は桁上がり，減算時はゼロ判定処理を行います。

(f) 正規化を行い，演算結果をFPR1に格納します。

(7) 作業エリア

(a) 仮数部の演算処理用に，FPR1に対してDEHLレジスタを，FPR2に対してAXBCレジスタをワークとして使用します。

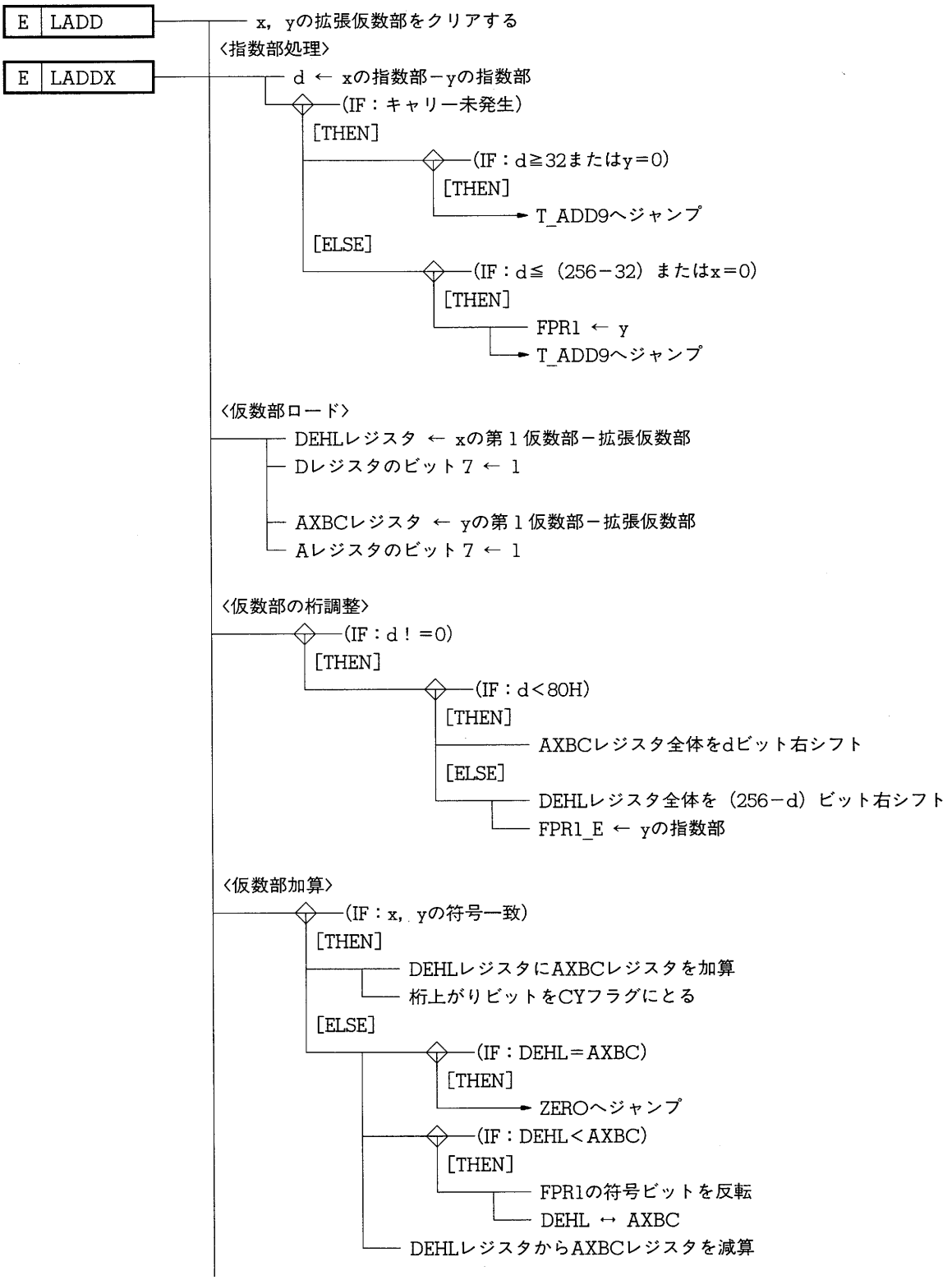
D ← FPR1_H,　　A ← FPR2_H
E ← FPR1_M,　　X ← FPR2_M
H ← FPR1　,　　B ← FPR2
L ← FPR1_X,　　C ← FPR2_X

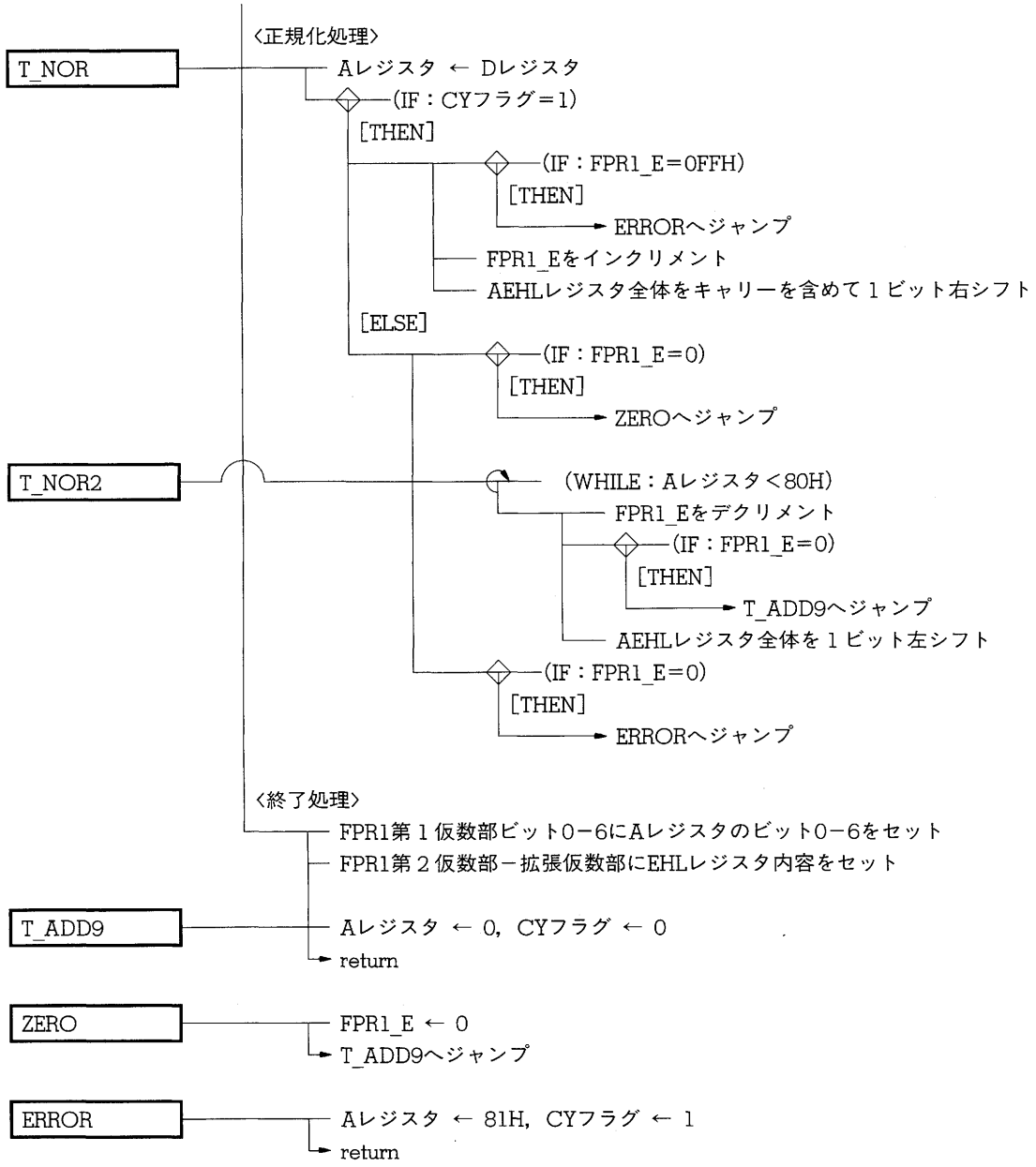
D，Aレジスタのビット７には，サイン・ビットの代わりにMSBを立てます。

(b) 正規化処理部では，第１仮数部-拡張仮数部に対して，AEHLレジスタを使用します。

(c) FPR1_Xを指数部差の退避エリアとして使用します（処理図では，便宜上，dという変数名で表しています)。

(8) 処 理 図


E LADD
E LADDX
x, yの拡張仮数部をクリアする
〈指数部処理〉
d ← xの指数部－yの指数部
(IF：キャリー未発生)
[THEN]
(IF：d≧32またはy=0)
[THEN]
T_ADD9へジャンプ
[ELSE]
(IF：d≦（256－32）またはx=0)
[THEN]
FPR1 ← y
T_ADD9へジャンプ
〈仮数部ロード〉
DEHLレジスタ ← xの第1仮数部－拡張仮数部
Dレジスタのビット7 ← 1
AXBCレジスタ ← yの第1仮数部－拡張仮数部
Aレジスタのビット7 ← 1
〈仮数部の桁調整〉
(IF：d！=0)
[THEN]
(IF：d<80H)
[THEN]
AXBCレジスタ全体をdビット右シフト
[ELSE]
DEHLレジスタ全体を（256－d）ビット右シフト
FPR1_E ← yの指数部
〈仮数部加算〉
(IF：x, yの符号一致)
[THEN]
DEHLレジスタにAXBCレジスタを加算
桁上がりビットをCYフラグにとる
[ELSE]
(IF：DEHL=AXBC)
[THEN]
ZEROへジャンプ
(IF：DEHL<AXBC)
[THEN]
FPR1の符号ビットを反転
DEHL ↔ AXBC
DEHLレジスタからAXBCレジスタを減算



備考1. ラベル E | | は広域名を示します。

2. ラベル T_NOR　T_NOR2　ZERO　ERROR は，LMLT，およびLDIV関数でも参照されます。

3. ラベル E | LADDX は，数学関数などで，拡張仮数部を使用した加算を実行するための内部的な広域名です。

3.2　浮動小数点減算（LSUB）

(1) 処理内容

FPR1の値をx，FPR2の値をyとして，x－yをFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1

(3) 消費スタック・サイズ

2（LSUBからの戻り番地のみ）

(4) 使用レジスタ

FPR1，FPR2

(5) 処理時間（内部システム・クロック：6 MHz）

平均：150 μs

最大：300 μs（0.5－0.500000060）

(6) 処理手順

yの符号を反転し，LADDへジャンプします。

(7) 処 理 図

3.3　浮動小数点乗算（LMLT）

(1) 処理内容

FPR1の値をx，FPR2の値をyとして，x×yをFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1

(3) 消費スタック・サイズ

2（LMLTからの戻り番地のみ）

(4) 使用レジスタ

FPR1，FPR2

(5) 処理時間（内部システム・クロック：6 MHz）

平均：125 μs

最大：133 μs（1×0.5）

(6) 処理手順

(a) xまたはyが０の場合，０を返します。

(b) 指数部を加算し，オーバフローおよびアンダフローの判定をします。

(c) 符号のXORをとり，結果の符号とします。

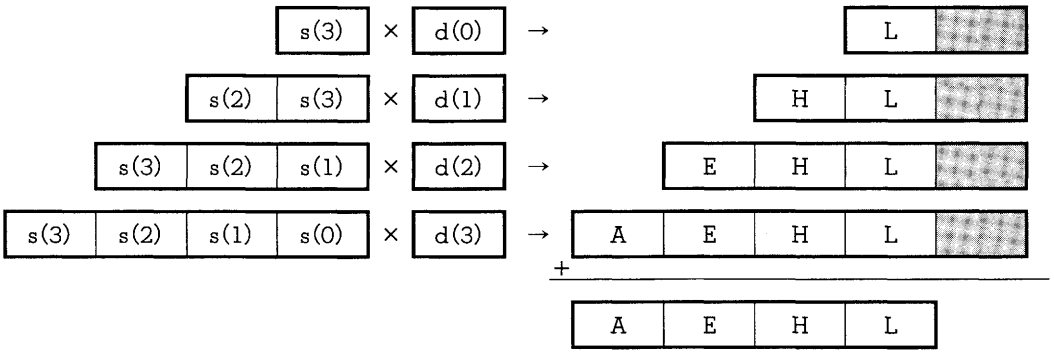
(d) 仮数部の乗算を次に示す方法で行います。

仮数部の演算処理用に指数部を除くFPR1，FPR2を便宜上，要素数４のBYTE型配列d(4)，s(4)と見なします（添字は０-３の値をとるものとします）。

FPR1_H	FPR1_M	FPR1	FPR1_X	を　d(4)
FPR2_H	FPR2_M	FPR2	FPR2_X	を　s(4)

d(3)，s(3)のビット７には，サイン・ビットの代わりにMSBを立てます。

下図に示す手順でAEHLレジスタに乗算結果を積み上げていきます(▒▒ の部分は切り捨てています)。

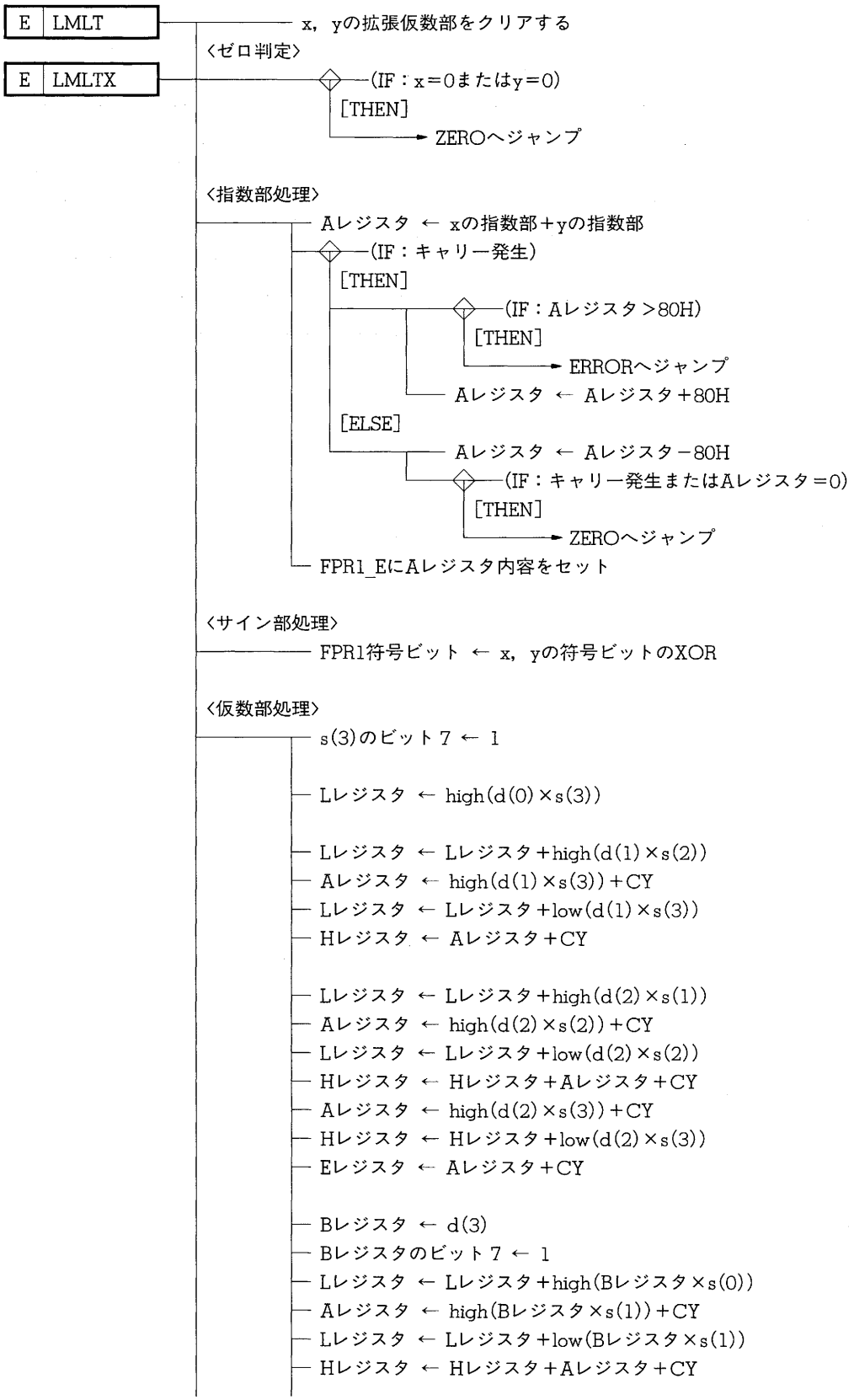


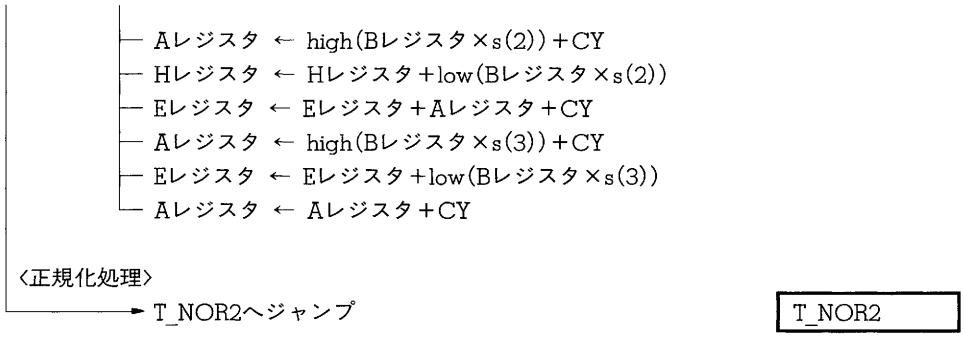
(e) 正規化ルーチンへジャンプします。

(7) 作業エリア

仮数部の乗算結果の積み上げ用に，AEHLレジスタを使用します。

(8) 処 理 図


E LMLT
x，yの拡張仮数部をクリアする
〈ゼロ判定〉
E LMLTX
(IF：x＝0またはy＝0)
[THEN]
ZEROへジャンプ
〈指数部処理〉
Aレジスタ ← xの指数部＋yの指数部
(IF：キャリー発生)
[THEN]
(IF：Aレジスタ＞80H)
[THEN]
ERRORへジャンプ
Aレジスタ ← Aレジスタ＋80H
[ELSE]
Aレジスタ ← Aレジスタ－80H
(IF：キャリー発生またはAレジスタ＝0)
[THEN]
ZEROへジャンプ
FPR1_EにAレジスタ内容をセット
〈サイン部処理〉
FPR1符号ビット ← x，yの符号ビットのXOR
〈仮数部処理〉
s(3)のビット7 ← 1
Lレジスタ ← high(d(0)×s(3))
Lレジスタ ← Lレジスタ＋high(d(1)×s(2))
Aレジスタ ← high(d(1)×s(3))＋CY
Lレジスタ ← Lレジスタ＋low(d(1)×s(3))
Hレジスタ ← Aレジスタ＋CY
Lレジスタ ← Lレジスタ＋high(d(2)×s(1))
Aレジスタ ← high(d(2)×s(2))＋CY
Lレジスタ ← Lレジスタ＋low(d(2)×s(2))
Hレジスタ ← Hレジスタ＋Aレジスタ＋CY
Aレジスタ ← high(d(2)×s(3))＋CY
Hレジスタ ← Hレジスタ＋low(d(2)×s(3))
Eレジスタ ← Aレジスタ＋CY
Bレジスタ ← d(3)
Bレジスタのビット7 ← 1
Lレジスタ ← Lレジスタ＋high(Bレジスタ×s(0))
Aレジスタ ← high(Bレジスタ×s(1))＋CY
Lレジスタ ← Lレジスタ＋low(Bレジスタ×s(1))
Hレジスタ ← Hレジスタ＋Aレジスタ＋CY



備考1. high（ ）は，乗算結果の上位1バイトを示します。

2. low（ ）は，乗算結果の下位1バイトを示します。

3. ラベル E | LMLTX は数学関数などで，拡張仮数部を使用した乗算を実行するための内部的な広域名です。

3.4　浮動小数点除算（LDIV）

(1) 処理内容

FPR1の値をx，FPR2の値をyとして，x／yをFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1

(3) 消費スタック・サイズ

2（LDIVからの戻り番地のみ）

(4) 使用レジスタ

FPR1，FPR2

(5) 処理時間（内部システム・クロック：6 MHz）

平均：510 μs

最大：609 μs（−0.999999940／−0.5）

(6) 処理手順

(a) yが0の場合，異常終了を返します。

(b) xが0の場合，0を返します。

(c) xの指数部からyの指数部を減算し，オーバフローとアンダフローの判定をします。

(d) 符号のXORをとり，結果の符号とします。

(e) 仮数部の除算を次に示す方法で行います。

xの仮数部に対してCYフラグとX，B，Cレジスタで置き換えたものを25ビットのd，sまた，yの仮数部に対して第1仮数部をLレジスタで置き換えたものを24ビットの整数型（符号なし）d，sと見なします。

```
CY ←        0,
X  ← FPR1_H,     L ← FPR2_H
B  ← FPR1_M,
C  ← FPR1   ,
```

X，Lレジスタのビット7には，サイン・ビットの代わりにMSBを立てておきます。

CY	Xレジスタ	Bレジスタ	Cレジスタ	を　d

Lレジスタ	FPR2_M	FPR2	を　s

通常の手計算の方法により，CYフラグとDEHレジスタに商を積み上げていきます。

(i) CYフラグ，X，B，C，Lレジスタの初期設定とHレジスタのクリアを行います。

(ii) dとsを比較し，d≧sならdからsを引き，キャリー・フラグをセット(1)します。
d<sならキャリー・フラグをリセット(0)にします。

(iii) DEHレジスタ全体をキャリー・フラグを含めて1ビット左シフトします。

(iv) dを1ビット左シフトします。

(v) (ii)-(iv)を25回繰り返します（ただし25回目は(iv)を実行しません)。

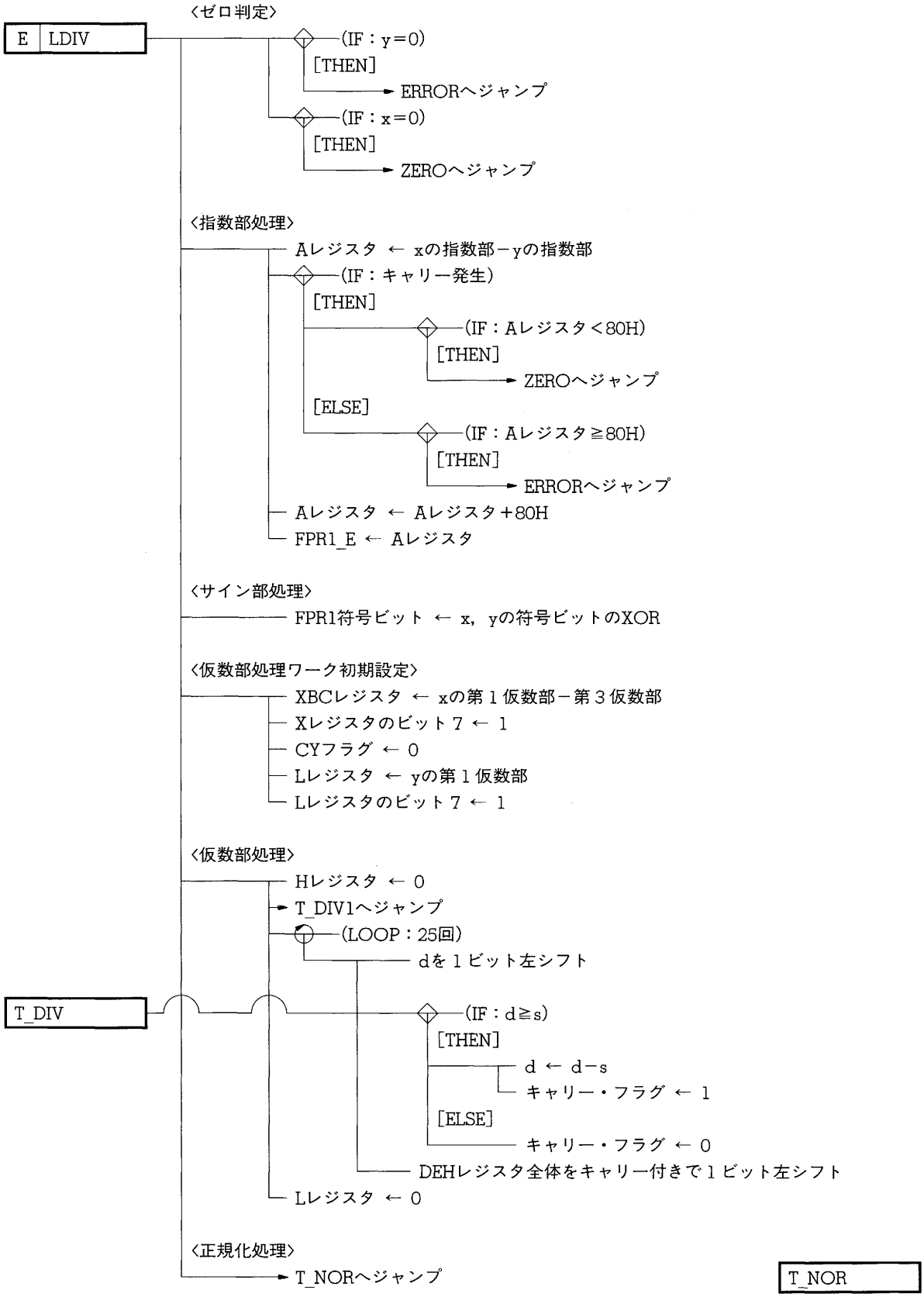
(vi) 最後の処理(iii)で発生したキャリーを仮数部の桁上がりビットとします。

(f) 正規化ルーチンへジャンプします。

(7) 作業エリア

FPR1（1バイト）を仮数部除算時のループ・カウンタとして使用しています。

(8) 処 理 図


E LDIV
〈ゼロ判定〉
(IF：y=0)
[THEN]
ERRORへジャンプ
(IF：x=0)
[THEN]
ZEROへジャンプ
〈指数部処理〉
Aレジスタ ← xの指数部－yの指数部
(IF：キャリー発生)
[THEN]
(IF：Aレジスタ<80H)
[THEN]
ZEROへジャンプ
[ELSE]
(IF：Aレジスタ≧80H)
[THEN]
ERRORへジャンプ
Aレジスタ ← Aレジスタ+80H
FPR1_E ← Aレジスタ
〈サイン部処理〉
FPR1符号ビット ← x, yの符号ビットのXOR
〈仮数部処理ワーク初期設定〉
XBCレジスタ ← xの第1仮数部－第3仮数部
Xレジスタのビット7 ← 1
CYフラグ ← 0
Lレジスタ ← yの第1仮数部
Lレジスタのビット7 ← 1
〈仮数部処理〉
Hレジスタ ← 0
T_DIV1へジャンプ
(LOOP：25回)
dを1ビット左シフト
T_DIV
(IF：d≧s)
[THEN]
d ← d−s
キャリー・フラグ ← 1
[ELSE]
キャリー・フラグ ← 0
DEHレジスタ全体をキャリー付きで1ビット左シフト
Lレジスタ ← 0
〈正規化処理〉
T_NORへジャンプ
T_NOR

第４章　数学関数

数学関数として，次のものを用意しています。

(1) sin関数（LSIN）

FPR1の値の正弦を求めます。

(2) cos関数（LCOS）

FPR1の値の余弦を求めます。

(3) tan関数（LTAN）

FPR1の値の正接を求めます。

(4) 自然対数関数（LLOG）

FPR1の値の自然対数をとります。

(5) 常用対数関数（LLOG10）

FPR1の値の常用対数をとります。

(6) 指数関数（底＝e）（LEXP）

FPR1の値を指数値，底をeとした指数関数解を求めます。

(7) 指数関数（底＝10）（LEXP10）

FPR1の値を指数値，底を10とした指数関数解を求めます。

(8) べき乗関数（LPOW）

FPR1の値とFPR2の値のべき乗を求めます。

(9) 平方根関数（LSQRT）

FPR1の値の平方根を求めます。

(10) arcsin関数（LASIN）

FPR1の値の逆正弦を求めます。

(11) arccos関数（LACOS）

FPR1の値の逆余弦を求めます。

(12) arctan関数（LATAN）

FPR1の値の逆正接を求めます。

(13) sinh関数（LHSIN）

FPR1の値のhyperbolic sine関数解を求めます。

(14) cosh関数（LHCOS）

FPR1の値のhyperbolic cosine関数解を求めます。

(15) tanh関数（LHTAN）

FPR1の値のhyperbolic tangent関数解を求めます。

(16) 絶対値関数（LABS）

FPR1の値の絶対値をとります。

(17) 逆数関数（LRCPN）

FPR1の値の逆数を求めます。

演算結果は，FPR1に格納されます。

4.1　共通サブルーチン

先に述べたように，平方根関数を除く他の数学関数は，Taylor近似式，または最良近似式を用いて計算しています。これら近似式は高次の多項式で表され，かつ，Taylor展開，および最良近似の性格上共通のパターンを持っています。

そこで，２つのタイプの多項式の計算関数（LPLY，LPLY2）を共通サブルーチンとして用意しています。

(1) 処理内容

多項式の計算結果をFPR1へ返します。

(2) アルゴリズム

次の２つのタイプの多項式計算が可能です。

①	$X+k_1XY+k_1k_2XY^2+k_1k_2k_3XY^3+\cdots+k_1k_2\cdots k_nXY^n$
②	$Z+k_1XY+k_1k_2XY^2+k_1k_2k_3XY^3+\cdots+k_1k_2\cdots k_nXY^n$

備考　X　：多項式第１項

Y　：各項の次数変化に対する乗算定数（X^2など）

(k_1，k_1k_2，…，$k_1k_2\cdots k_n$)　：各項の係数

多項式	使　用　条　件	計算関数
①	各項が共通約数Xを持っている場合	LPLY
②	第１項が定数の場合	LPAY2

誤差を最小限におさえるために，X，Y，Z，k_1，k_2，…k_nは拡張仮数部を持つ浮動小数点数を使用しています。

注意　Taylor近似の性格上，｜第１項｜＞｜第２項｜＞…｜第ｎ項｜が成立している必要があります。

(3) 処理手順

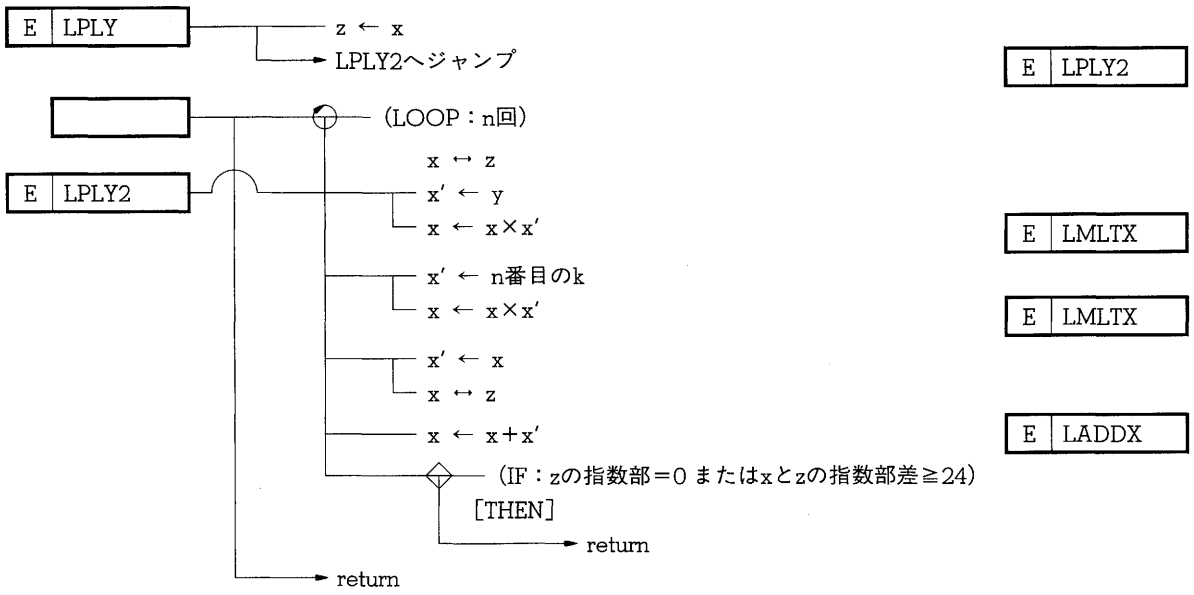
(a) 第N項を第N－第１項とY，k_nの乗算によって求めます。

(b) 第N－第１項までの和に第N項の値を加算します。

(c) 第N項が第N項までの和に比して十分に小さければ，計算を終了します。

(d) (a)－(b)を第n項まで繰り返します。

(4) 処 理 図



備考　x ：FPR1

x′：FPR2

z ：FPR3

y ：FPR4

n ：Cレジスタ

kの先頭アドレスはDEレジスタにより受け渡します。

4.2　sin関数（LSIN）

(1) 処理内容

FPR1の値をxとして，sin(x)をFPR1に返します。

●単位：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSIN,FTOL，LTOF

(3) 消費スタック・サイズ

10(LSINからの戻り番地２バイトを含む)

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4

(5) 処理時間（内部システム・クロック：6 MHz）

平均：2.69 ms

最大：7.10 ms（sin(1.701412e+38)）

(6) アルゴリズム

次のTaylor展開近似式を用いて求めます。

$$SIN(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-\frac{x^{11}}{11!}$$

(7) 処理手順

(a) SIN(−x)＝−SIN(x)よりxが負なら符号を記憶し，xの絶対値をとります。

(b) xを$0\leqq x<\pi/2$の範囲にスケーリングします。

x′を$0\leqq x'<\pi/2$とすると，次のようになります。

$$SIN(\pi/2+x')=SIN(\pi/2-x')$$
$$SIN(2\pi/2+x')=SIN(-x')$$
$$SIN(3\pi/2+x')=SIN(x'-\pi/2)$$

したがって，xを$\pi/2$で割った商をn，余りをx′とすると，SIN(x)は，次のようになります。

n/4の余り	SIN(x)	x''
0	SIN(x')	x'
1	SIN(π/2−x')	π/2−x'
2	SIN(−x')	−x'
3	SIN(x'−π/2)	x'−π/2

(c) 記憶していた符号をx''にかけます。

(d) SIN(x'')をTaylor近似式により求めます。

近似多項式の第1項(x'')，各項の係数を除く比(x''^2)，および第2項−第6項係数の前項係数に対する比(−1/3!，−3!/5!，−5!/7!，−7!/9!，−9!/11!)を引数に多項式計算関数をCALLします。

(8) 作業エリア

π/2による除算商と余り算出処理部で，次のように使用します。

(a) FPR4_1を，除算商の積み上げ用として使用します（処理図では便宜上，nという変数名で表しています)。

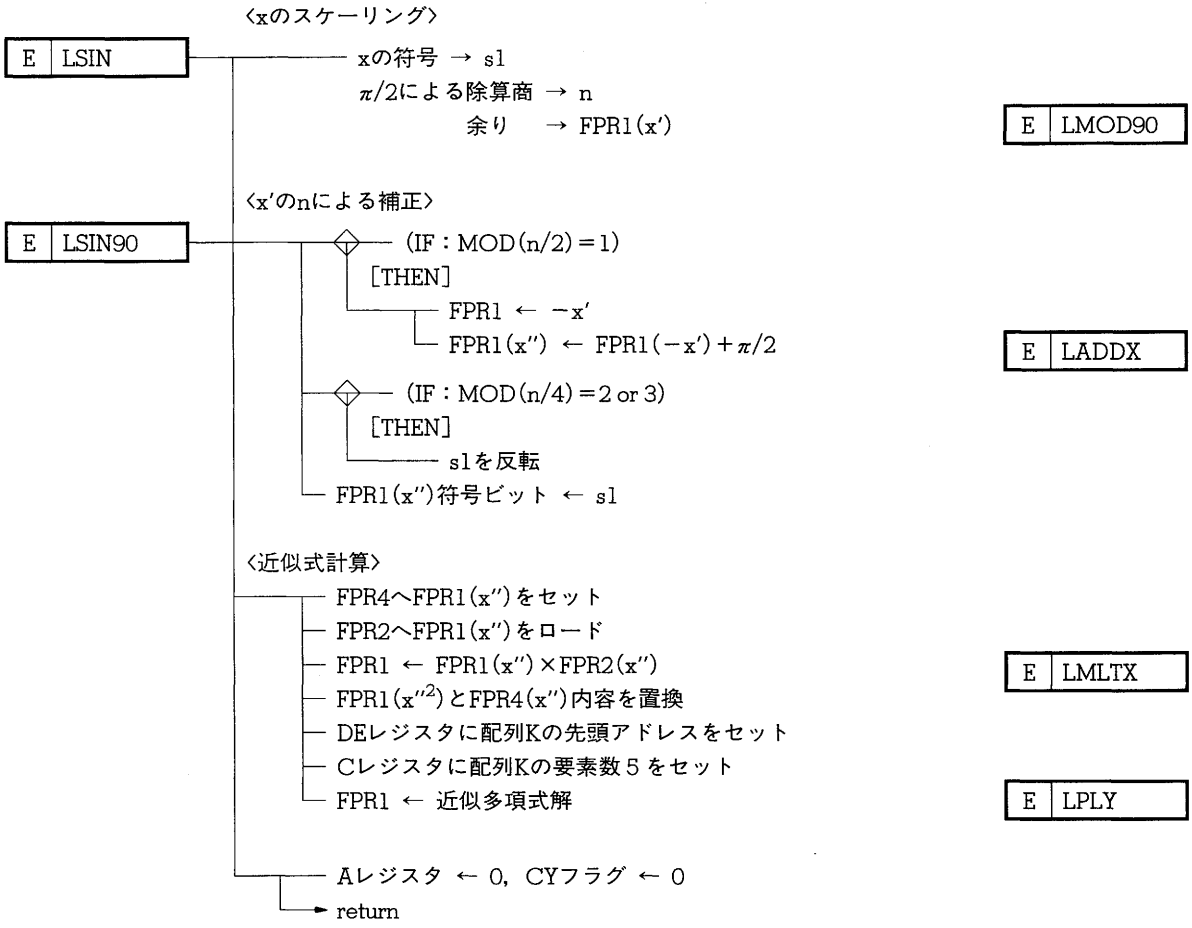
(b) FPR4_2を，xの符号退避用ワークとして使用します（処理図では便宜上，s1という変数名で表しています)。

(9) 浮動小数点定数データ

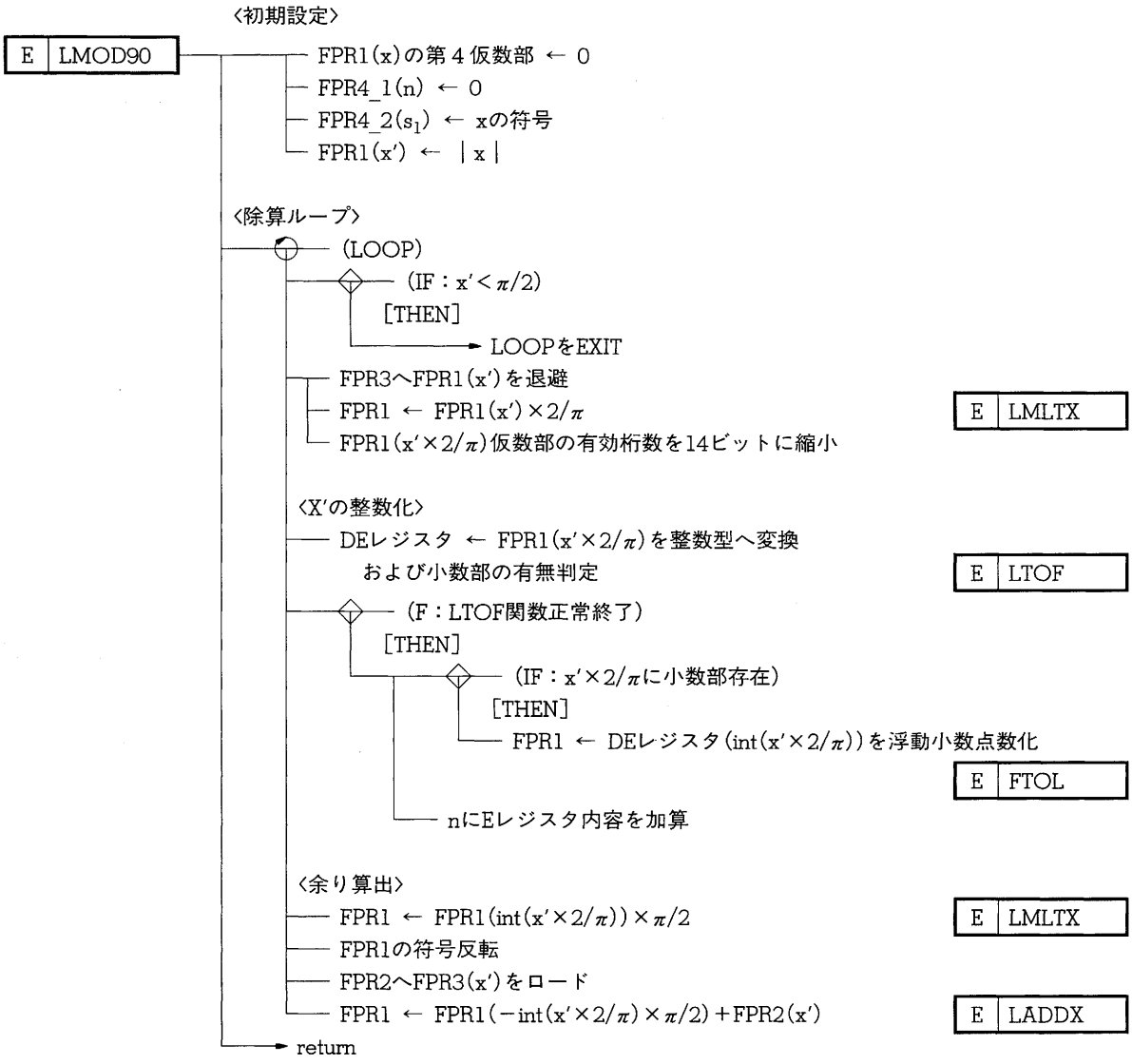
(a) 0−π/2のスケーリング用に拡張仮数部を持つ定数データ2/πとπ/2を用いています。

(b) Taylor近似式の係数列に対して，拡張仮数部を持つ定数データ−1/3!，−3!/5!，−5!/7!，−7!/9!，−9!/11!を，要素数5の配列Kとして使用しています。

(10) 処 理 図


〈xのスケーリング〉
E LSIN
xの符号 → s1
π/2による除算商 → n
余り → FPR1(x')
E LMOD90
〈x'のnによる補正〉
E LSIN90
(IF : MOD(n/2)＝1)
[THEN]
FPR1 ← −x'
FPR1(x'') ← FPR1(−x')＋π/2
E LADDX
(IF : MOD(n/4)＝2 or 3)
[THEN]
s1を反転
FPR1(x'')符号ビット ← s1
〈近似式計算〉
FPR4へFPR1(x'')をセット
FPR2へFPR1(x'')をロード
FPR1 ← FPR1(x'')×FPR2(x'')
E LMLTX
FPR1(x''2)とFPR4(x'')内容を置換
DEレジスタに配列Kの先頭アドレスをセット
Cレジスタに配列Kの要素数5をセット
FPR1 ← 近似多項式解
E LPLY
Aレジスタ ← 0, CYフラグ ← 0
return

（π/2による除算商，および余り算出サブルーチン）



備考　int(X)はXの整数部を表します。

4.3　cos関数（LCOS）

(1) 処理内容

FPR1の値をxとして，cos(x)をFPR1に返します。

●単位：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSIN，LCOS，FTOL，LTOF

(3) 消費スタック・サイズ

10(LCOSからの戻り番地２バイトを含む)

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4

(5) 処理時間（内部システム・クロック：６MHz）

平均：2.67 ms

最大：7.74 ms（cos(1.701412e+38)）

(6) アルゴリズム

次の式により，SIN(X')を求めます。

$$COS(x)=COS(|x|)=SIN(|x|+\pi/2)$$
$$x'=|x|+\pi/2$$

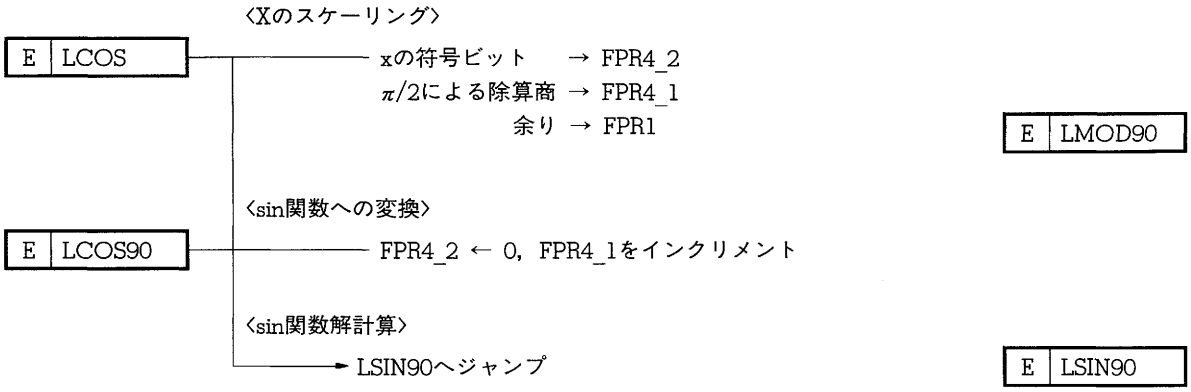
(7) 処理手順

(a) LMOD90サブルーチンにより，$\pi/2$による除算商(FPR4_1)と余り(FPR1)を求めます。

(b) LMOD90が退避した符号(FPR4_2)をクリアし，FPR4_1に１を加えます。

(c) LSIN90へジャンプします。

(8) 処 理 図


〈Xのスケーリング〉
E LCOS
xの符号ビット　→ FPR4_2
π/2による除算商 → FPR4_1
余り → FPR1
E LMOD90
〈sin関数への変換〉
E LCOS90
FPR4_2 ← 0, FPR4_1をインクリメント
〈sin関数解計算〉
LSIN90へジャンプ
E LSIN90

4.4　tan関数（LTAN）

(1) 処理内容

FPR1の値をxとして，tan(x)をFPR1に返します。

●単位：ラジアン

(2) リンケージ対象オブジェト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSIN，LCOS，LTAN，FTOL，LTOF

(3) 消費スタック・サイズ

14(LTANからの戻り番地２バイトを含む)

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：６MHz）

平均：5.54 ms

最大：10.29 ms（tan(1.701412e+38)）

(6) アルゴリズム

次の式により求めます。

$$TAN(x)=\frac{SIN(x)}{COS(x)}$$

(7) 処理手順

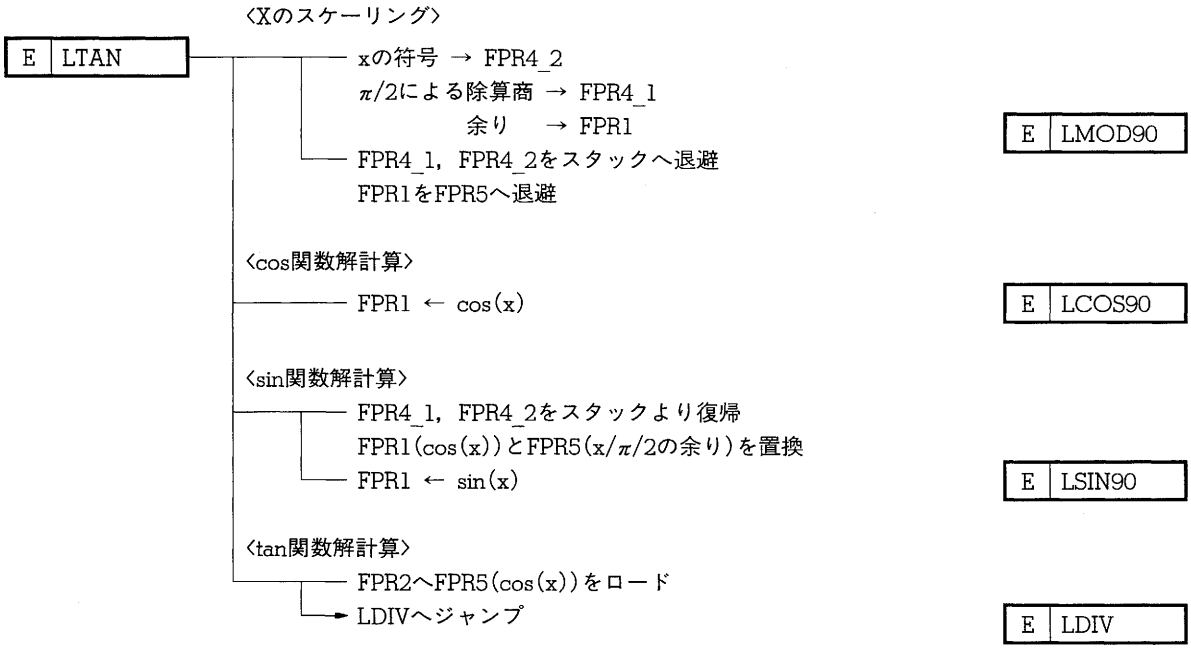
(a) LMOD90サブルーチンにより，$\pi/2$による除算商（FPR4_1），余り(FPR1)，およびxの符号(FPR4_2)を得ます。

(b) LCOS90サブルーチンにより，cos(x)を求めます。

(c) LSIN90サブルーチンにより，sin(x)を求めます。

(d) sin(x)／cos(x)を解とします。

(8) 処 理 図


〈Xのスケーリング〉
E LTAN
xの符号 → FPR4_2
π/2による除算商 → FPR4_1
余り → FPR1
FPR4_1, FPR4_2をスタックへ退避
FPR1をFPR5へ退避
E LMOD90
〈cos関数解計算〉
FPR1 ← cos(x)
E LCOS90
〈sin関数解計算〉
FPR4_1, FPR4_2をスタックより復帰
FPR1(cos(x))とFPR5(x/π/2の余り)を置換
FPR1 ← sin(x)
E LSIN90
〈tan関数解計算〉
FPR2へFPR5(cos(x))をロード
LDIVへジャンプ
E LDIV

4.5　自然対数関数（LLOG）

(1) 処理内容

FPR1の値をxとして，log(x)をFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LLOG，FTOL

(3) 消費スタック・サイズ

10(LLOGからの戻り番地2バイトを含む)

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4

(5) 処理時間（内部システム・クロック：6 MHz）

平均：3.08 ms

最大：4.36 ms（log(3.27680039e+10)）

(6) アルゴリズム

次の式により求めます。

●Xの指数部＝X_e，仮数部＝X_fと置くと，log(x)は次のとおりです。

$$\log(x)=X_e\times\log(2)+\log(X_f)$$

●$\log(X_f)$ の求め方

① $X'=\dfrac{X_f-1}{X_f+1}$とする

② $\log(X_f)=\log(1+X')-\log(1-X')$のTaylor展開近似式を使用

③ $\log(x)=X_e\times\log(2)+2X'+\dfrac{2}{3}X'^3+\dfrac{2}{5}X'^5+\dfrac{2}{7}X'^7+\dfrac{2}{9}X'^9+\dfrac{2}{11}X'^{11}+\dfrac{2}{13}X'^{13}$

(a) x≦0ならエラーを返します。

(b) x＝1なら0を返します。

(c) $X_e\times\log(2)$を求めます。

(d) X'を$(X_f-1)/(X_f+1)$により求めます。

(e) log(x)をTaylor近似式により求めます。

指数部の対数と近似多項式の第1項の加算値($X_e \times \log(2)+2X'$)，第2項の計算初期値として2X'，各項の係数を除く比(X'2)，および第2項－第7項係数の前項係数に対する比(1/3，3/5，5/7，7/9，9/11，11/13)を引数に多項式計算関数をCALLします。

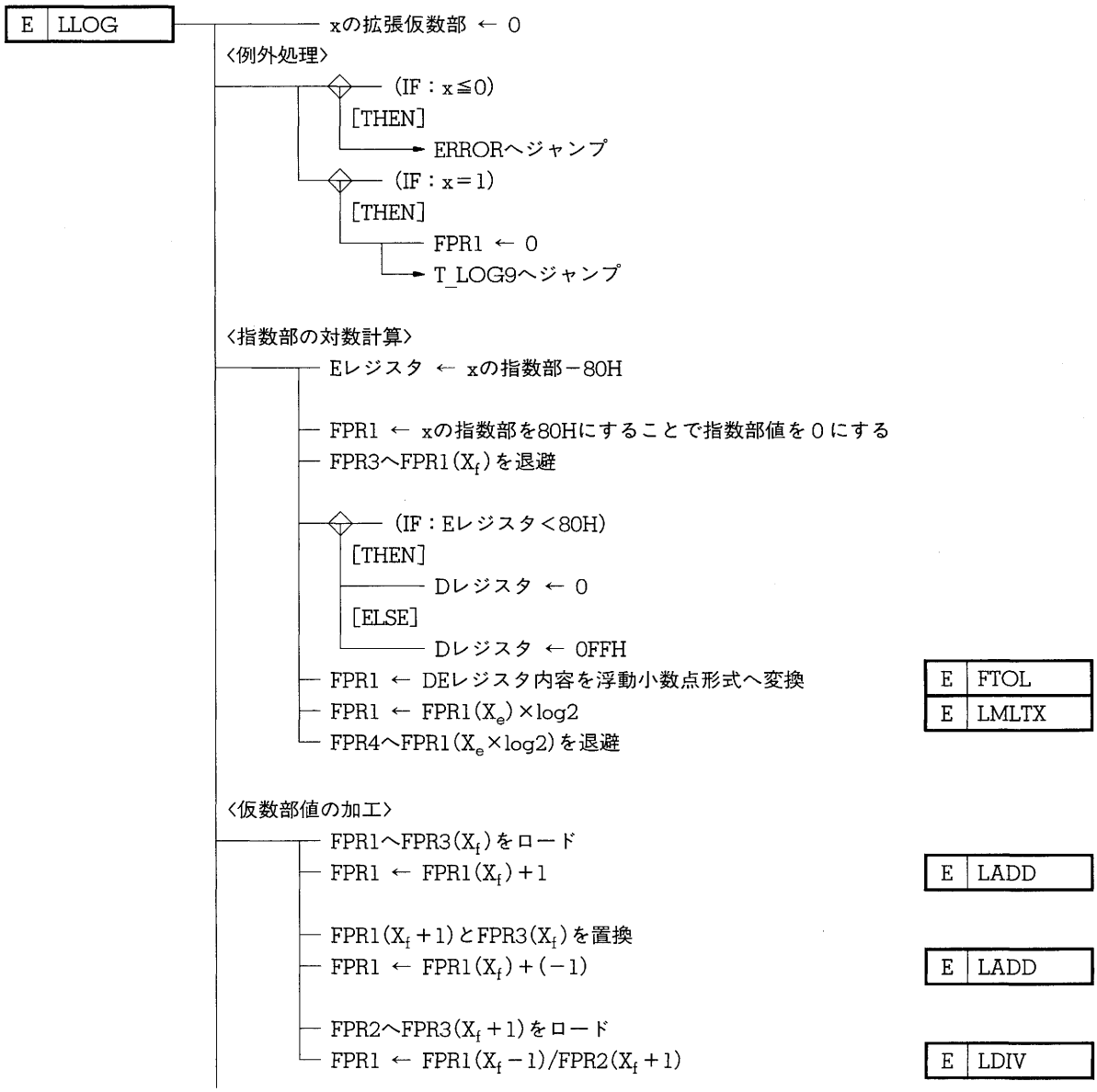
(7) 浮動小数点定数データ

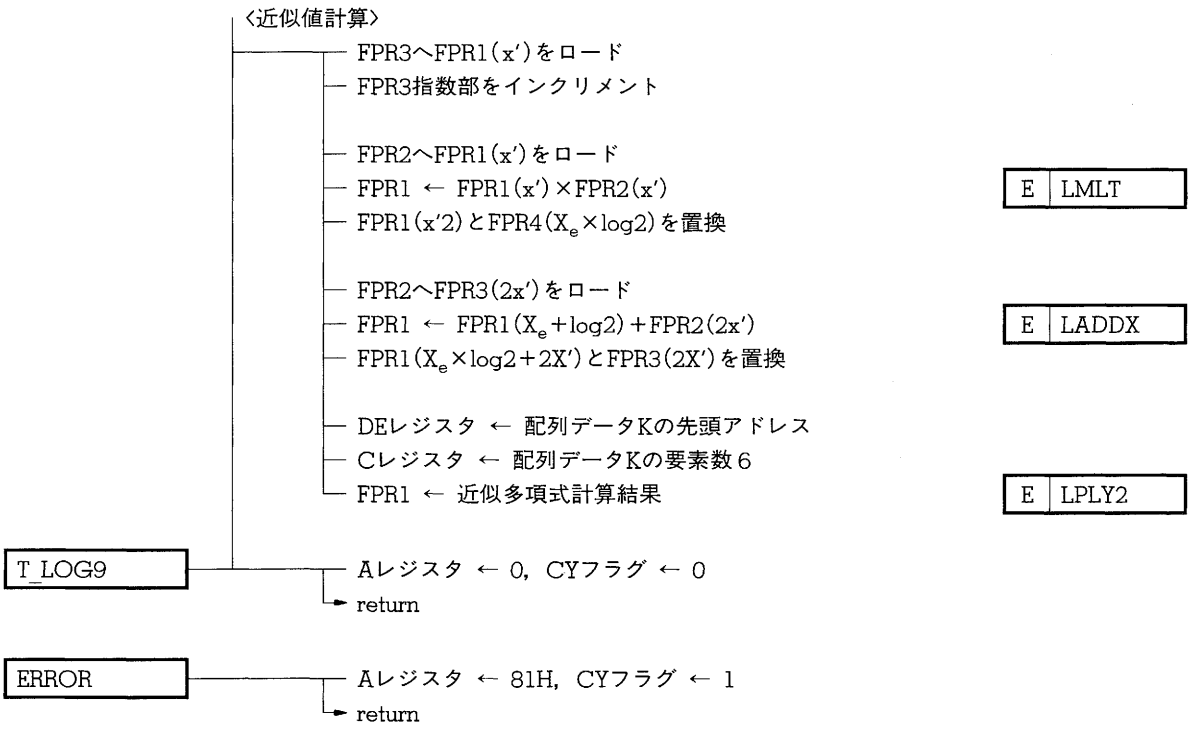
(a) x＝1の判定部および，X'の計算部において定数データ1を用いています。

(b) 指数部値の対数計算部において拡張仮数部を持つ定数データlog(2)を用いています。

(c) 近似多項式の係数列に対して，拡張仮数部を持つ浮動小数点定数データ1/3，3/5，5/7，7/9，9/11，11/13を要素数6の配列Kとして定義しています。

(8) 処 理 図


E LLOG
xの拡張仮数部 ← 0
〈例外処理〉
(IF : x≦0)
[THEN]
ERRORへジャンプ
(IF : x=1)
[THEN]
FPR1 ← 0
T_LOG9へジャンプ
〈指数部の対数計算〉
Eレジスタ ← xの指数部−80H
FPR1 ← xの指数部を80Hにすることで指数部値を0にする
FPR3へFPR1(Xf)を退避
(IF : Eレジスタ<80H)
[THEN]
Dレジスタ ← 0
[ELSE]
Dレジスタ ← 0FFH
FPR1 ← DEレジスタ内容を浮動小数点形式へ変換
E FTOL
FPR1 ← FPR1(Xe)×log2
E LMLTX
FPR4へFPR1(Xe×log2)を退避
〈仮数部値の加工〉
FPR1へFPR3(Xf)をロード
FPR1 ← FPR1(Xf)+1
E LADD
FPR1(Xf+1)とFPR3(Xf)を置換
FPR1 ← FPR1(Xf)+(−1)
E LADD
FPR2へFPR3(Xf+1)をロード
FPR1 ← FPR1(Xf−1)/FPR2(Xf+1)
E LDIV


〈近似値計算〉
FPR3へFPR1(x')をロード
FPR3指数部をインクリメント
FPR2へFPR1(x')をロード
FPR1 ← FPR1(x')×FPR2(x')
FPR1(x'2)とFPR4(Xe×log2)を置換
E
LMLT
FPR2へFPR3(2x')をロード
FPR1 ← FPR1(Xe+log2)+FPR2(2x')
FPR1(Xe×log2+2X')とFPR3(2X')を置換
E
LADDX
DEレジスタ ← 配列データKの先頭アドレス
Cレジスタ ← 配列データKの要素数6
FPR1 ← 近似多項式計算結果
E
LPLY2
T_LOG9
Aレジスタ ← 0, CYフラグ ← 0
return
ERROR
Aレジスタ ← 81H, CYフラグ ← 1
return

4.6　常用対数関数（LLOG10）

(1) 処理内容

FPR1の値をxとして，log10(x)をFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LLOG，LLOG10，FTOL

(3) 消費スタック・サイズ

12(LLOG10からの戻り番地2バイトを含む)

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4

(5) 処理時間（内部システム・クロック：6 MHz）

平均：3.24 ms

最大：4.47 ms（log10(3.27680039e+04)）

(6) アルゴリズム

次の式により求めます。

$$\log10(x)=\frac{\log(x)}{\log(10)}$$

(7) 処理手順

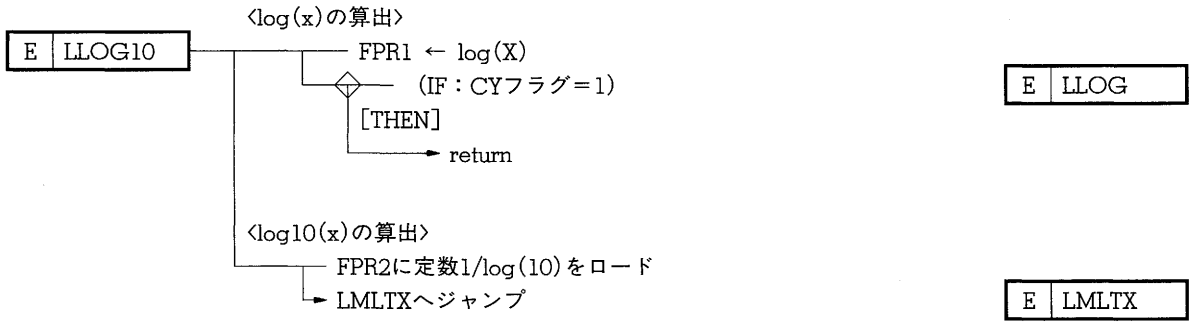
(a) LLOG関数によりlog(x)を求めます。

(b) LLOG関数の結果がエラーの場合，そのままエラーを返します。

(8) 浮動小数点定数データ

拡張仮数部を持つ定数データ1/log(10)を用います。

(9) 処 理 図


E LLOG10
<log(x)の算出>
FPR1 ← log(X)
(IF：CYフラグ＝1)
[THEN]
return
<log10(x)の算出>
FPR2に定数1/log(10)をロード
LMLTXへジャンプ
E LLOG
E LMLTX

4.7　指数関数（底＝e）（LEXP）

(1) 処理内容

FPR1の値をxとして，e^xをFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LEXP，FTOL，LTOF

(3) 消費スタック・サイズ

10(LEXPからの戻り番地２バイトを含む)

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：６MHz）

平均：3.43 ms

最大：4.54 ms（exp(−5.67776156)）

(6) アルゴリズム

指数底を２に変換することにより先に解の指数部を求める方法をとっています。

次の式により求めます。

$$e^x = 2^{x/\log 2} = 2^{\mathrm{int}(x/\log 2)} \times 2^{\mathrm{dec}(x/\log 2)}$$

備考1. int(X)はXの上位整数(int(X)−1≦X<int(X))を示します。

2. dec(X)はXの小数部(dec(X)＝X−int(X)；−1≦dec(X)<0)を示します。

$2^{\mathrm{dec}(x/\log 2)}$ は0.5−1の範囲となりますから，int(x/log2)は解の指数部そのものとなります。

仮数部は，$2^{\mathrm{dec}(x/\log 2)+1}$により，次のTaylor近似式により求めます(２をかけても得られる仮数部は同じため，次式に最も有利な0−1の範囲を使います)。

X′＝dec(x/log2)＋1として，

$$2X' = 1 + \frac{\log 2}{1!}X' + \frac{(\log 2)^2}{2!}X'^2 + \frac{(\log 2)^3}{3!}X'^3 + \cdots + \frac{(\log 2)^9}{9!}X'^9$$

最後に，2X′は1−2の範囲の数ですが，丸めをゼロへ向かって行っていることから2X′<1となる場

合があります。この場合int(X/log2)－1を解の指数部とします。

(7) 処理手順

(a) x/log2を求めます。

(b) オーバフローの場合

- x<0なら0を解として返します。
- x>0ならエラーを解として返します。

(c) int(x/log2)を求めます。

(d) int(x/log2)<－7FHなら0を解として返します。

int(x/log2)>7FHならエラーを解として返します。

(e) dec(x/log2)＋1を求めX'とします。

(f) $2^{x'}$をTaylor近似式により求めます。

近似多項式の第2項までの和（1＋log2・X'），第3項の計算初期値としてlog2・X'，各項の係数を除く比(X')，および第3項－第10項係数の前項係数に対する比(log2/2，log2/3，log2/4，log2/5，log2/6，log2/7，log2/8，log2/9)を引数に多項式計算関数をCALLします。

(g) $2^{x'}$<1の場合，解の指数部(int(x/log2))から1を引きます。

(h) 近似式の計算結果に指数部値int(x/log2)を埋め込み，解とします。

(8) 作業エリア

(a) FPR3_1のビット7をxの符号の退避エリアとして使用します(処理図では，記号Sで表しています)。

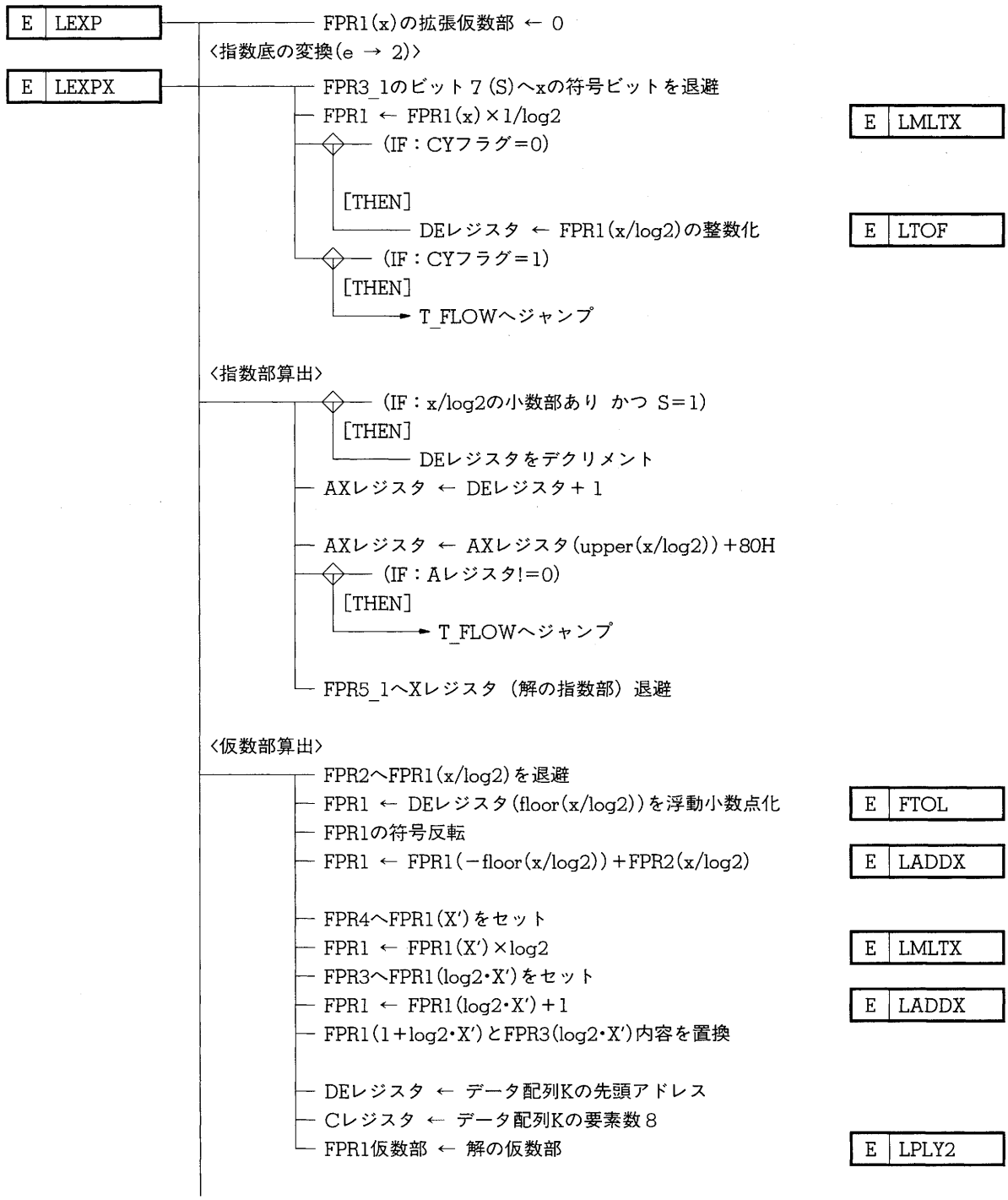
(b) FPR5_1を指数部値int(x/log2)の退避エリアとして使用します。

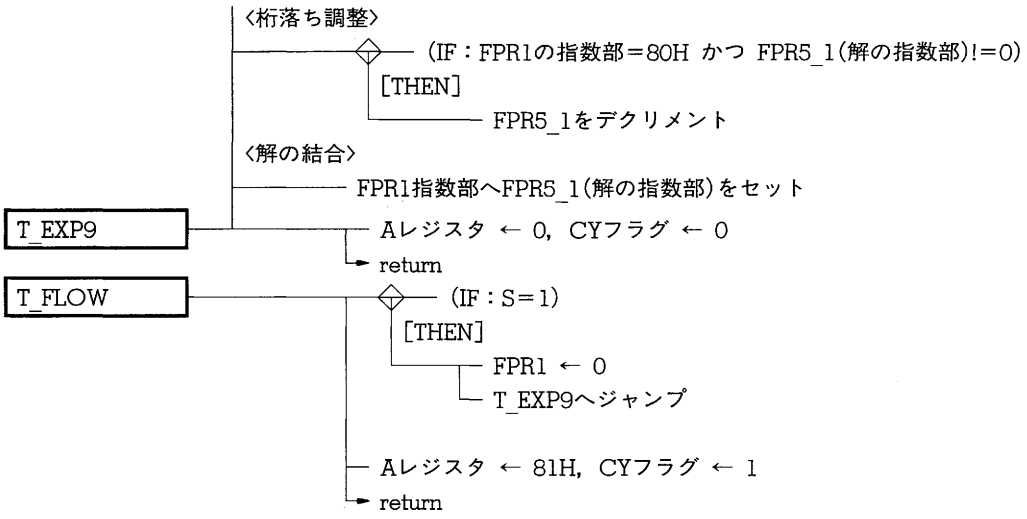
(9) 浮動小数点定数データ

(a) 拡張仮数部を持つ定数データ1/log2，log2，1を使用します。

(b) 近似多項式の係数列に対して，拡張仮数部を持つ定数データlog2/2，log2/3，log2/4，log2/5，log2/6，log2/7，log2/8，log2/9を要素数8の配列Kとして使用します。

(10) 処 理 図


E LEXP
FPR1(x)の拡張仮数部 ← 0
〈指数底の変換(e → 2)〉
E LEXPX
FPR3_1のビット7(S)へxの符号ビットを退避
FPR1 ← FPR1(x)×1/log2
E LMLTX
(IF：CYフラグ=0)
[THEN]
DEレジスタ ← FPR1(x/log2)の整数化
E LTOF
(IF：CYフラグ=1)
[THEN]
T_FLOWへジャンプ
〈指数部算出〉
(IF：x/log2の小数部あり かつ S=1)
[THEN]
DEレジスタをデクリメント
AXレジスタ ← DEレジスタ+1
AXレジスタ ← AXレジスタ(upper(x/log2))+80H
(IF：Aレジスタ!=0)
[THEN]
T_FLOWへジャンプ
FPR5_1へXレジスタ（解の指数部）退避
〈仮数部算出〉
FPR2へFPR1(x/log2)を退避
FPR1 ← DEレジスタ(floor(x/log2))を浮動小数点化
E FTOL
FPR1の符号反転
FPR1 ← FPR1(−floor(x/log2))+FPR2(x/log2)
E LADDX
FPR4へFPR1(X')をセット
FPR1 ← FPR1(X')×log2
E LMLTX
FPR3へFPR1(log2･X')をセット
FPR1 ← FPR1(log2･X')+1
E LADDX
FPR1(1+log2･X')とFPR3(log2･X')内容を置換
DEレジスタ ← データ配列Kの先頭アドレス
Cレジスタ ← データ配列Kの要素数8
FPR1仮数部 ← 解の仮数部
E LPLY2



備考1. upper(X)は，Xの上位整数(upper(X)－1≦X＜upper(X))を示します。

2. floor(X)は，Xの下位整数(floor(X)≦X＜floor(X)＋1)を示します。

3. ラベル | E | LEXPX | は他の数学関数などで，拡張仮数部を使用した指数計算を実行するための内部的な広域名です。

4.8　指数関数（底＝10）（LEXP10）

(1) 処理内容

FPR1の値をxとして，10^xをFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LEXP，LEXP10，FTOL，LTOF

(3) 消費スタック・サイズ

10(LEXP10からの戻り番地２バイトを含む)

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：3.53 ms

最大：4.76 ms（exp10(−0.979693294)）

(6) アルゴリズム

次の式により求めます。

$$10^x = e^{x \times \log(10)}$$

(7) 処理手順

(a) x×log(10)を求めます。

(b) (a)の乗算結果がエラーの場合，そのままエラーを返します。

(c) LEXP関数へジャンプします。

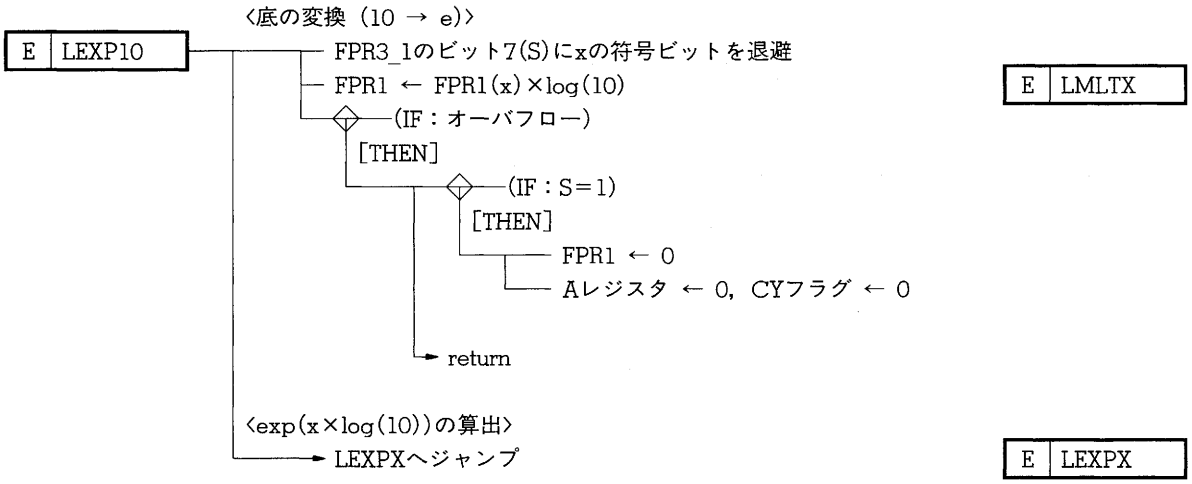
(8) 作業エリア

FPR3_1をxの符号ビットの退避エリアとして使用します(処理図では，Sという記号で表しています)。

(9) 浮動小数点定数データ

指数底の変換時に，拡張仮数部を持つ定数データlog(10)を用います。

(10) 処 理 図


E LEXP10
〈底の変換（10 → e)〉
FPR3_1のビット7(S)にxの符号ビットを退避
FPR1 ← FPR1(x)×log(10)
(IF：オーバフロー)
[THEN]
(IF：S=1)
[THEN]
FPR1 ← 0
Aレジスタ ← 0, CYフラグ ← 0
return
〈exp(x×log(10))の算出〉
LEXPXへジャンプ
E LMLTX
E LEXPX

4.9　べき乗関数（LPOW）

(1) 処理内容

FPR1の値をa，FPR2の値をbとして，a^bをFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LLOG，LEXP，LPOW，FTOL，LTOF

(3) 消費スタック・サイズ

14(LPOWからの戻り番地２バイトを含む)

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：6.50 ms

最大：8.72 ms（−0.5 ^ −3）

(6) アルゴリズム

a，b数値の組み合わせにより計算方法が異なります。

a，b	a^b
a=0，b≦0	エラー
a=0，b>0	0
a>0	$e^{b\log(a)}$
a<0，b=0	1
a<0，bは０以外の整数　bは偶数：	$e^{b\log(\lvert a\rvert)}$
bは奇数：	$-e^{b\log(\lvert a\rvert)}$
a<0，bは整数でない	エラー

(7) 処理手順

(a) a=0かつb≦0なら，演算結果としてエラーを返します。

(b) a=0かつb>0なら，演算結果として０を返します。

(c) a<0かつb=0なら，演算結果として１を返します。

(d) a<0かつb≠0なら，bの整数判定，および整数の場合，偶奇数判定を行います。
　bは整数でないなら演算結果としてエラーを返します。

(e) $e^{b\log(|a|)}$ をLLOG，LEXP関数により求めます。

(f) aが負，かつbが奇整数なら符号ビットをたてます。

(8) 作業エリア

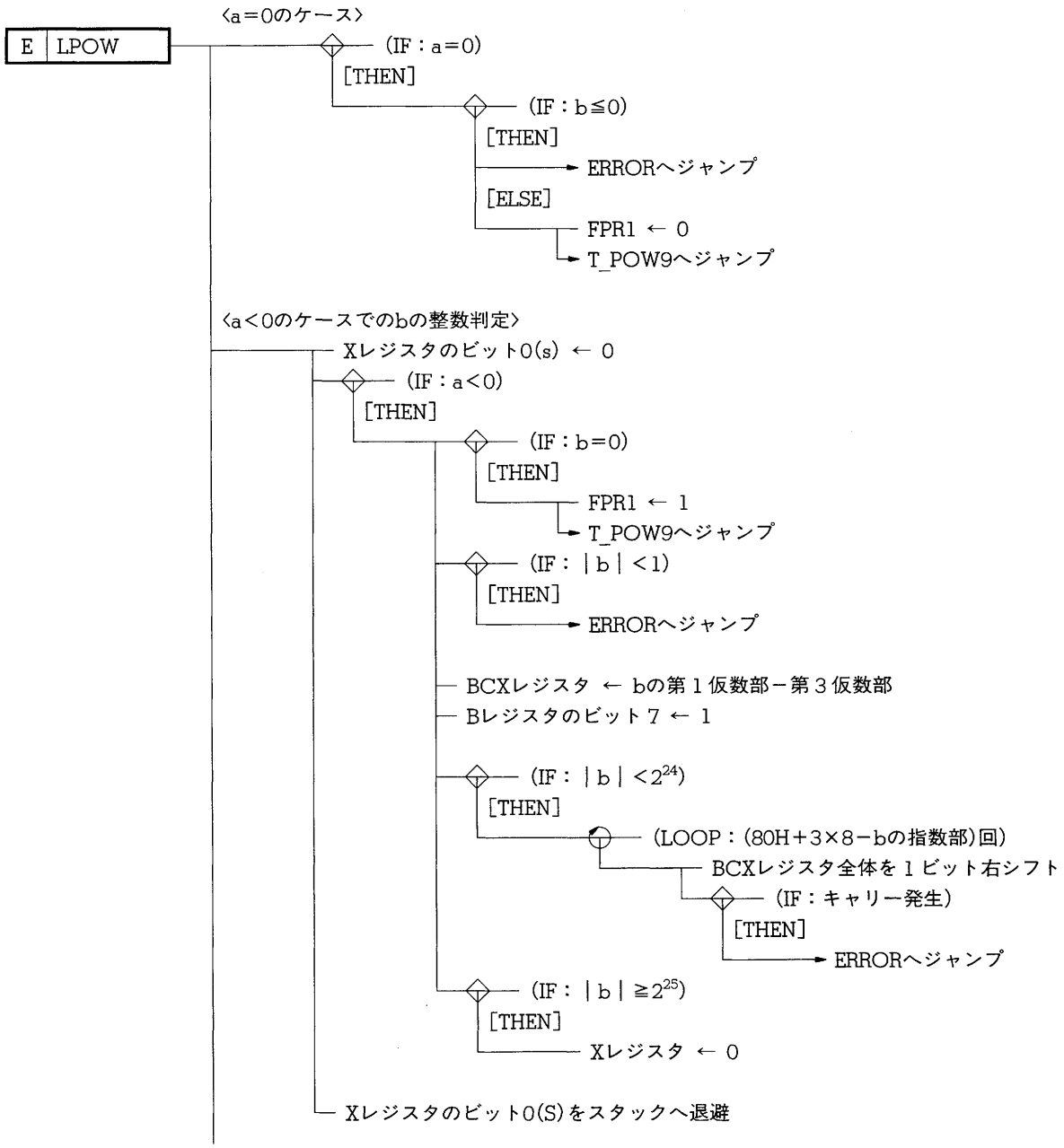
(a) Xレジスタ・ビット0を符号用ワークとして使用します（処理図では，変数名sで表しています)。

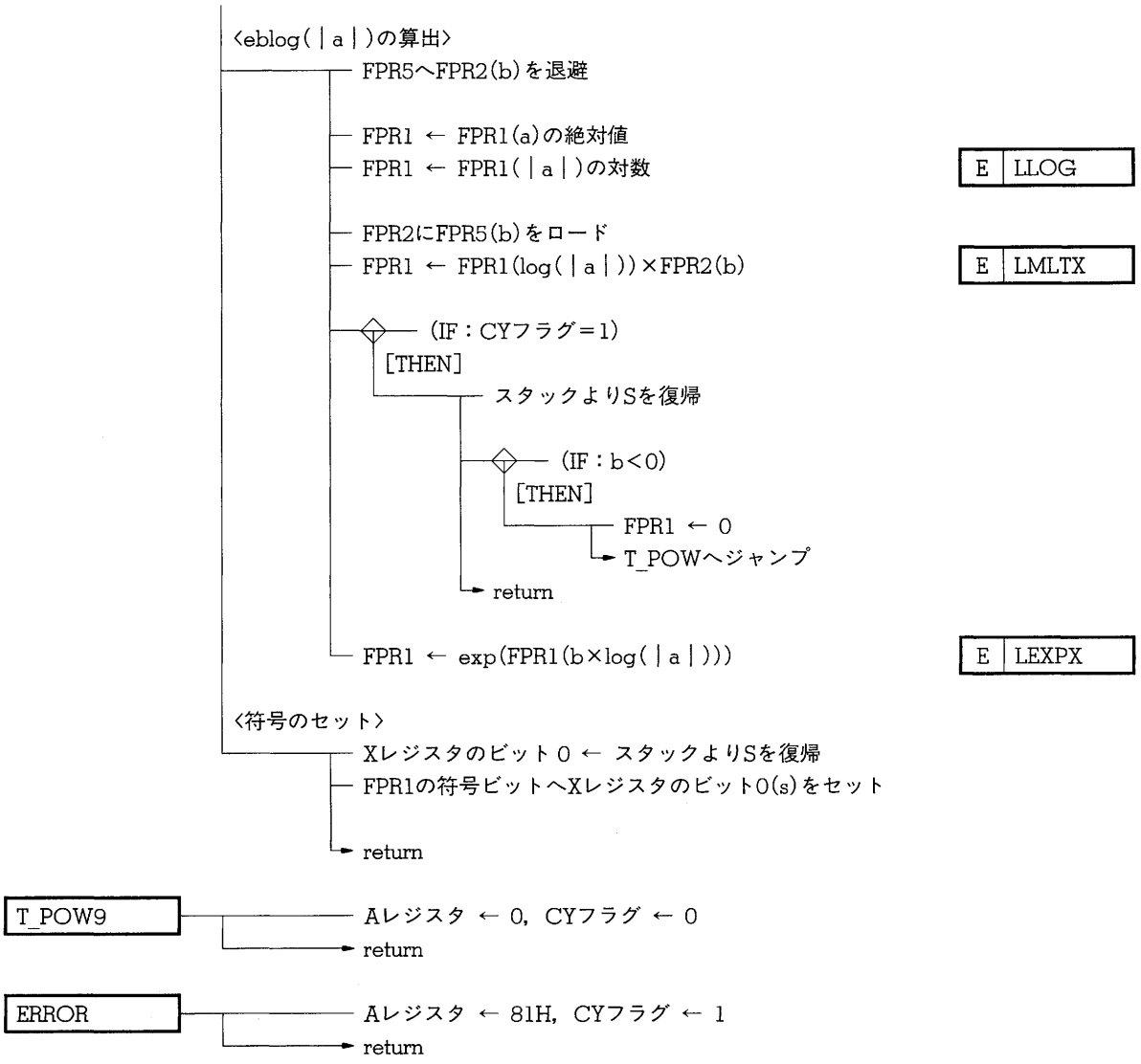
(b) bの整数判定部において，BCXレジスタをbの仮数部処理用ワークとして使用します。

(9) 浮動小数点定数データ

a<0かつb=0の場合の解として，定数データ1を使用します。

(10) 処 理 図


E | LPOW
〈a＝0のケース〉
(IF：a＝0)
[THEN]
(IF：b≦0)
[THEN]
ERRORへジャンプ
[ELSE]
FPR1 ← 0
T_POW9へジャンプ
〈a＜0のケースでのbの整数判定〉
Xレジスタのビット0(s) ← 0
(IF：a＜0)
[THEN]
(IF：b＝0)
[THEN]
FPR1 ← 1
T_POW9へジャンプ
(IF：|b|＜1)
[THEN]
ERRORへジャンプ
BCXレジスタ ← bの第1仮数部－第3仮数部
Bレジスタのビット7 ← 1
(IF：|b|＜2^24)
[THEN]
(LOOP：(80H＋3×8－bの指数部)回)
BCXレジスタ全体を1ビット右シフト
(IF：キャリー発生)
[THEN]
ERRORへジャンプ
(IF：|b|≧2^25)
[THEN]
Xレジスタ ← 0
Xレジスタのビット0(S)をスタックへ退避


〈eblog(｜a｜)の算出〉
FPR5へFPR2(b)を退避
FPR1 ← FPR1(a)の絶対値
FPR1 ← FPR1(｜a｜)の対数
E LLOG
FPR2にFPR5(b)をロード
FPR1 ← FPR1(log(｜a｜))×FPR2(b)
E LMLTX
(IF：CYフラグ＝1)
[THEN]
スタックよりSを復帰
(IF：b<0)
[THEN]
FPR1 ← 0
T_POWへジャンプ
return
FPR1 ← exp(FPR1(b×log(｜a｜)))
E LEXPX
〈符号のセット〉
Xレジスタのビット0 ← スタックよりSを復帰
FPR1の符号ビットへXレジスタのビット0(s)をセット
return
T_POW9
Aレジスタ ← 0，CYフラグ ← 0
return
ERROR
Aレジスタ ← 81H，CYフラグ ← 1
return

4.10　平方根関数（LSQRT）

（1）処理内容

FPR1の値をaとして，$\sqrt{a}$ をFPR1に返します。

（2）リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LLD，LSQRT

（3）消費スタック・サイズ

4（LSQRTからの戻り番地２バイトを含む）

（4）使用レジスタ

FPR1，FPR2，FPR3，FPR4

（5）処理時間（内部システム・クロック：6 MHz）

平均：2.56 ms

最大：2.76 ms（$\sqrt{(6.30915472e+36)}$）

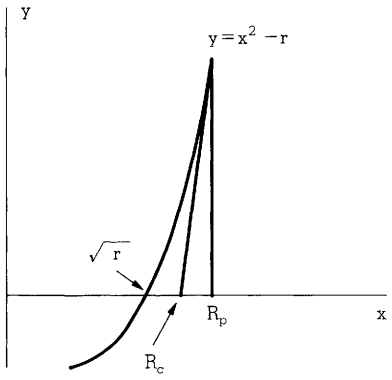
（6）アルゴリズム

$$\sqrt{a} = \sqrt{(r\times 2^{2n})} = 2^n \times \sqrt{r} \quad (0.25 \leqq r < 1)$$

と置くことで，最初から指数部（2^n）と，仮数部（$\sqrt{r}$）とに分けて計算します。

$\sqrt{r}$ の計算には，Newton-Raphson法を用います。

下図に示すように $\sqrt{r}$ は，２次関数$y=x^2-r$とx軸との交点のx座標です。



R_pを $\sqrt{r}$ の過大近似値として，直線$x=R_p$と$y=x^2-r$の交点から$y=x^2-r$に対して接線を引きます。

接線とx軸との交点のx座標をR_cとすると，R_cは，R_pよりさらに近い$\sqrt{r}$の近似値となります。

R_cは，R_pにより，次の式で求めます。

$$R_c=\frac{R_p}{2}+\frac{r}{2R_p}$$

この関数では，$\sqrt{r}<1$より，初期近似値$R_1=1$として最高R_6近似値を解とします。

(7) 処理手順

(a) a<0の場合エラー，a=0の場合，0を演算結果として返します。

(b) aの指数部aeに対して（ae+1)/2を求め，指数部の根nとします。

(c) nに対してaの指数部を調整し，r（=a/n）とします。

(d) 第2次近似値（R_2）r/2+1/2を求めます。

(e) 第2次近似値を最初のR_pとして第6次近似値まで計算します。

ただし途中で$R_c \geqq R_p$となった場合，その時点の近似値を解とします。

(f) 最後に，得られた仮数部近似解の指数部に指数部の根（n）を埋め込みます。

(8) 作業エリア

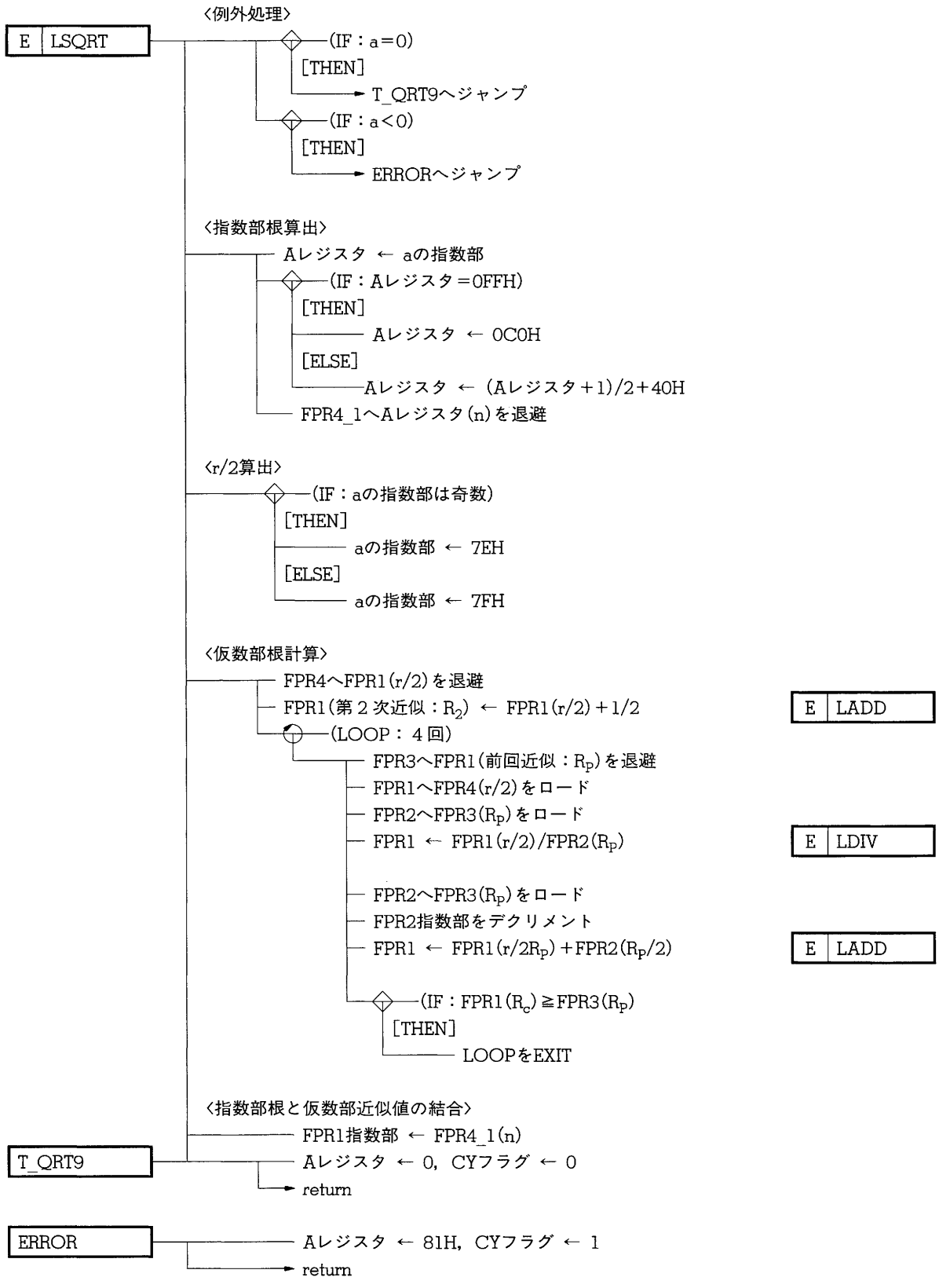
(a) FPR4_1を指数部の根（n）の退避エリアとして使用します。

(b) FPR3_1を仮数部近似値の計算部でのループ・カウンタとして使用します。

(9) 浮動小数点定数データ

第1次近似値の計算過程で定数データ1/2を使用しています。

(10) 処 理 図


E | LSQRT
〈例外処理〉
(IF：a＝0)
[THEN]
T_QRT9へジャンプ
(IF：a＜0)
[THEN]
ERRORへジャンプ
〈指数部根算出〉
Aレジスタ ← aの指数部
(IF：Aレジスタ＝0FFH)
[THEN]
Aレジスタ ← 0C0H
[ELSE]
Aレジスタ ← (Aレジスタ＋1)/2＋40H
FPR4_1へAレジスタ(n)を退避
〈r/2算出〉
(IF：aの指数部は奇数)
[THEN]
aの指数部 ← 7EH
[ELSE]
aの指数部 ← 7FH
〈仮数部根計算〉
FPR4へFPR1(r/2)を退避
FPR1(第2次近似：R_2) ← FPR1(r/2)＋1/2
E | LADD
(LOOP：4回)
FPR3へFPR1(前回近似：R_p)を退避
FPR1へFPR4(r/2)をロード
FPR2へFPR3(R_p)をロード
FPR1 ← FPR1(r/2)/FPR2(R_p)
E | LDIV
FPR2へFPR3(R_p)をロード
FPR2指数部をデクリメント
FPR1 ← FPR1(r/2R_p)＋FPR2(R_p/2)
E | LADD
(IF：FPR1(R_c)≧FPR3(R_p)
[THEN]
LOOPをEXIT
〈指数部根と仮数部近似値の結合〉
FPR1指数部 ← FPR4_1(n)
T_QRT9
Aレジスタ ← 0, CYフラグ ← 0
return
ERROR
Aレジスタ ← 81H, CYフラグ ← 1
return

4.11　arcsin関数（LASIN）

(1) 処理内容

FPR1の値をxとして，ARCSIN（x）をFPR1に返します。

- 入力値xの有効範囲：－1～1
- 返値の範囲　　：$-\pi/2 \sim \pi/2$
- 単位　　：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSQRT，LASIN，LATAN

(3) 消費スタック・サイズ

11（LASINからの戻り番地２バイトを含む）

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：4.41 ms

最大：7.95 ms（arcsin(0.999999940)）

(6) アルゴリズム

次の式により，逆正接関数に変換して解を求めます。

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

(7) 処理手順

(a) x＝０なら，演算結果として１を返します。

(b) x＝１なら$\pi/2$を，x＝－1なら$-\pi/2$を解として返します。

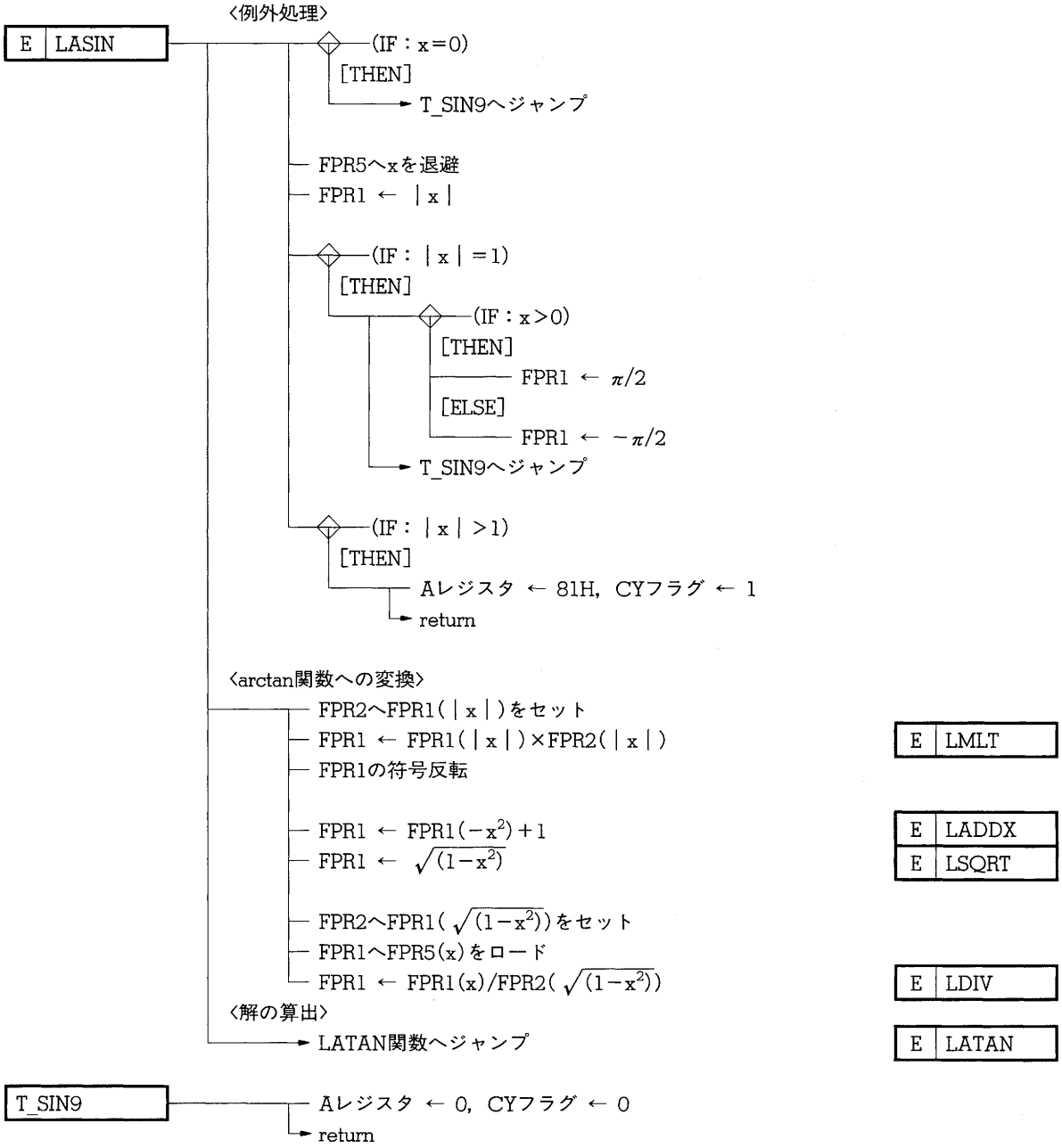
(c) |x|＞1なら，演算結果としてエラーを返します。

(d) $x/\sqrt{(1-X^2)}$ を求め，LATAN関数にジャンプします。

(8) 浮動小数点定数データ

拡張仮数部を持つ定数データ１および$\pi/2$を使用します。

(9) 処 理 図


〈例外処理〉
E LASIN
(IF：x=0)
[THEN]
T_SIN9へジャンプ
FPR5へxを退避
FPR1 ← |x|
(IF：|x|=1)
[THEN]
(IF：x>0)
[THEN]
FPR1 ← π/2
[ELSE]
FPR1 ← −π/2
T_SIN9へジャンプ
(IF：|x|>1)
[THEN]
Aレジスタ ← 81H，CYフラグ ← 1
return
〈arctan関数への変換〉
FPR2へFPR1(|x|)をセット
FPR1 ← FPR1(|x|)×FPR2(|x|)
FPR1の符号反転
FPR1 ← FPR1(−x²)+1
FPR1 ← √(1−x²)
FPR2へFPR1(√(1−x²))をセット
FPR1へFPR5(x)をロード
FPR1 ← FPR1(x)/FPR2(√(1−x²))
〈解の算出〉
LATAN関数へジャンプ
E LMLT
E LADDX
E LSQRT
E LDIV
E LATAN
T_SIN9
Aレジスタ ← 0，CYフラグ ← 0
return

4.12　arccos関数（LACOS）

(1) 処理内容

FPR1の値をxとして，ARCCOS（x）をFPR1に返します。

- 入力値xの有効範囲：－1～1
- 返値の範囲　　：0～π
- 単位　　：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSQRT，LASIN，LACOS，LATAN

(3) 消費スタック・サイズ

13（LACOSからの戻り番地２バイトを含む）

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：4.50 ms

最大：8.16 ms（arccos(0.99999940)）

(6) アルゴリズム

次の式により求めます。

$$ARCCOS(x) = \pi/2 - ARCSIN(x)$$

(7) 処理手順

(a) LASIN関数によりarcsin(x)を求めます。

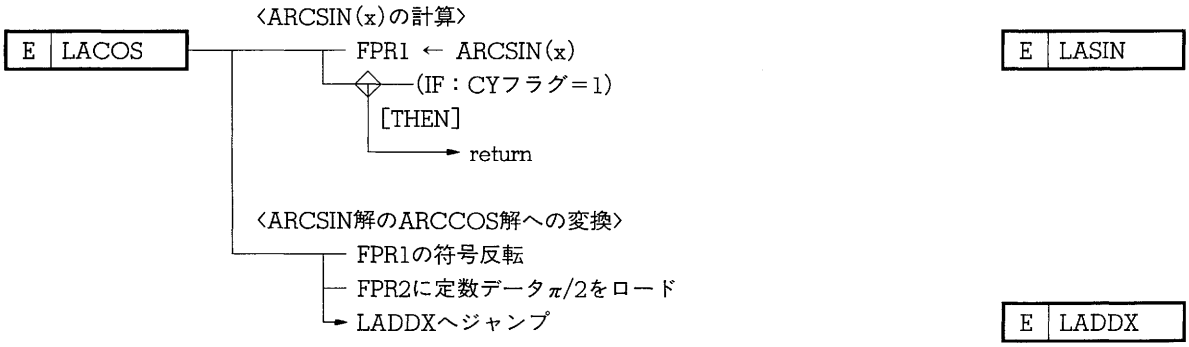
(b) LASIN関数の結果がエラーの場合，そのままエラーを返します。

(c) $\pi/2$－ARCSIN(x)を計算し，解とします。

(8) 浮動小数点定数データ

拡張仮数部を持つ定数データ$\pi/2$を使用します。

(9) 処 理 図


〈ARCSIN(x)の計算〉
E LACOS
FPR1 ← ARCSIN(x)
(IF：CYフラグ＝1)
[THEN]
return
E LASIN
〈ARCSIN解のARCCOS解への変換〉
FPR1の符号反転
FPR2に定数データπ/2をロード
LADDXへジャンプ
E LADDX

4.13　arctan関数（LATAN）

(1) 処理内容

FPR1の値をxとして，ARCTAN(x)をFPR1に返します。

- 返値の範囲：$-\pi/2 \sim \pi/2$
- 単位　　：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LATAN

(3) 消費スタック・サイズ

11（LATANからの戻り番地2バイトを含む）

(4) 使用レジスタ

FPR1, FPR2, FPR3, FPR4

(5) 処理時間（内部システム・クロック：6 MHz）

平均：3.33 ms

最大：4.47 ms（arctan(4)）

(6) アルゴリズム

Hastingsの最良近似式により求めます。

$$\arctan(x) \fallingdotseq \sum_{i=0}^{n} (A_i \times x^{2i+1})$$

この関数では，n=7，係数列A_0-A_7を次のように定めています。

$A_0 = 0.9999993329$　　$A_4 = 0.0964200441$

$A_1 = -0.3332985605$　　$A_5 = -0.0559098861$

$A_2 = 0.1994653599$　　$A_6 = 0.0218612288$

$A_3 = -0.1390853351$　　$A_7 = -0.0040540580$

注意　Hastingの最良近似式は $|x| \leq 1$ の範囲でしか使用できません。したがって，$|x| \geq 1$ の場合，次のようにして求めます。

$x'=\frac{|x|-1}{|x|+1}$としてxを置き換える

$\arctan(|x|)=\pi/4+\arctan(x')$

(7) 処理手順

(a) x=0の場合，演算結果として0を返します。

(b) xの符号および指数部を退避し，xの絶対値をとります。

(c) $|x|\geqq 1$の場合，Xを $(|x|-1)/(|x|+1)$ に変換します。

(d) arctan(x')を最良近似多項式により計算します。

近似多項式の第1項 ($A_0\times x'$)，第2項の計算初期値としてx'，各項の係数を除く比 (x'2)，および第2項係数と第3-8項係数の前項係数に対する比 (A_1，A_2/A_1，A_3/A_2，A_4/A_3，A_5/A_4，A_6/A_5，A_7/A_6) を引数に多項式計算関数をCALLします。

(e) $|x|>1$の場合，近似式解に$\pi/4$を加算します。

(f) xの元の符号をarctan(|x|)の演算結果に埋め込みます。

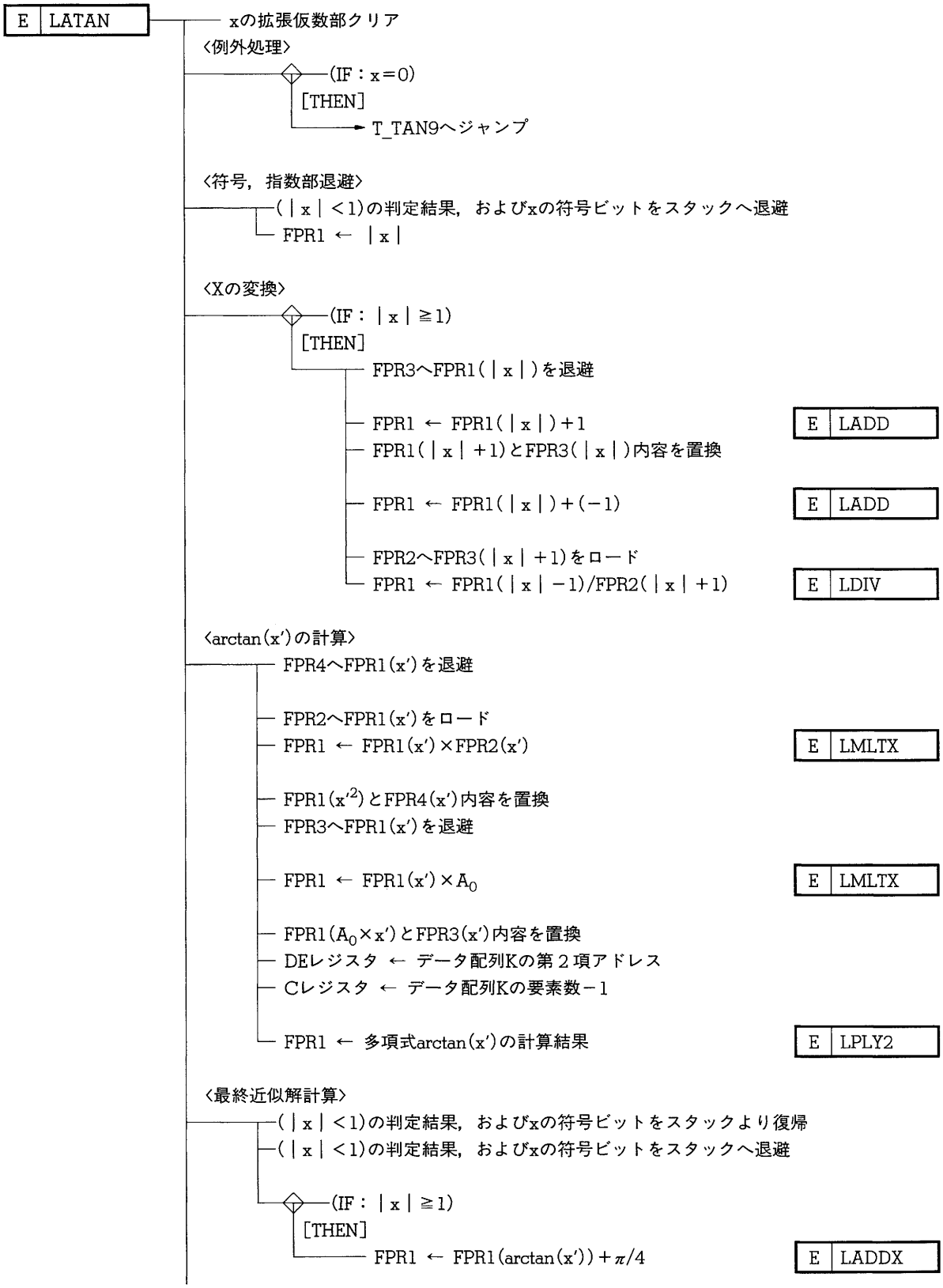
(8) 浮動小数点定数データ

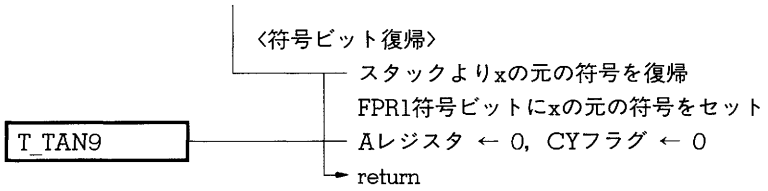
(a) xからx'への変換部において，定数データ1を使用しています。

(b) arctan(x')からarctan(|x|)への変換部において，拡張仮数部を持つ定数データ$\pi/4$を使用しています。

(c) 近似多項式の係数列に対して，拡張仮数部を持つ定数データA_0，A_1，A_2/A_1，A_3/A_2，A_4/A_3，A_5/A_4，A_6/A_5，A_7/A_6を要素数8の配列Kとして用いています。

(9) 処 理 図


E | LATAN
xの拡張仮数部クリア
〈例外処理〉
(IF：x=0)
[THEN]
T_TAN9へジャンプ
〈符号，指数部退避〉
(|x|<1)の判定結果，およびxの符号ビットをスタックへ退避
FPR1 ← |x|
〈Xの変換〉
(IF：|x|≧1)
[THEN]
FPR3へFPR1(|x|)を退避
FPR1 ← FPR1(|x|)+1
E | LADD
FPR1(|x|+1)とFPR3(|x|)内容を置換
FPR1 ← FPR1(|x|)+(−1)
E | LADD
FPR2へFPR3(|x|+1)をロード
FPR1 ← FPR1(|x|−1)/FPR2(|x|+1)
E | LDIV
〈arctan(x')の計算〉
FPR4へFPR1(x')を退避
FPR2へFPR1(x')をロード
FPR1 ← FPR1(x')×FPR2(x')
E | LMLTX
FPR1(x'^2)とFPR4(x')内容を置換
FPR3へFPR1(x')を退避
FPR1 ← FPR1(x')×A0
E | LMLTX
FPR1(A0×x')とFPR3(x')内容を置換
DEレジスタ ← データ配列Kの第2項アドレス
Cレジスタ ← データ配列Kの要素数−1
FPR1 ← 多項式arctan(x')の計算結果
E | LPLY2
〈最終近似解計算〉
(|x|<1)の判定結果，およびxの符号ビットをスタックより復帰
(|x|<1)の判定結果，およびxの符号ビットをスタックへ退避
(IF：|x|≧1)
[THEN]
FPR1 ← FPR1(arctan(x'))+π/4
E | LADDX


〈符号ビット復帰〉
スタックよりxの元の符号を復帰
FPR1符号ビットにxの元の符号をセット
T_TAN9
Aレジスタ ← 0, CYフラグ ← 0
return

4.14　sinh関数（LHSIN）

(1) 処理内容

FPR1の値をxとして，SINH(x)をFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LEXP，LHSIN，LRCPN，FTOL，LTOF

(3) 消費スタック・サイズ

14（LHSINからの戻り番地2バイトを含む）

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：3.09 ms

最大：5.03 ms（sinh(−86.6433868)）

(6) アルゴリズム

● |x| ≧0.5の場合

次の式により求めます。

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

● |x| <0.5の場合

Taylor展開近似式により求めます。

$$SINH\ (x)=x+\frac{1}{3!}x^{3}+\frac{1}{5!}x^{5}+\frac{1}{7!}x^{7}$$

備考　SINH(−x)＝−SINH(x)であることを利用しています。

(7) 処理手順

- $|x| \geqq 0.5$の場合

  (a) xの符号を記憶し，xの絶対値をとります。

  (b) $e^{|x|}$をLEXP関数により求めます。

  (c) $e^{|x|}$オーバフローの場合，そのままエラーを返します。

  (d) $(e^{|x|} - 1/e^{|x|})/2$を求め，xの元の符号を埋めこみ解とします。

- $|x| < 0.5$の場合

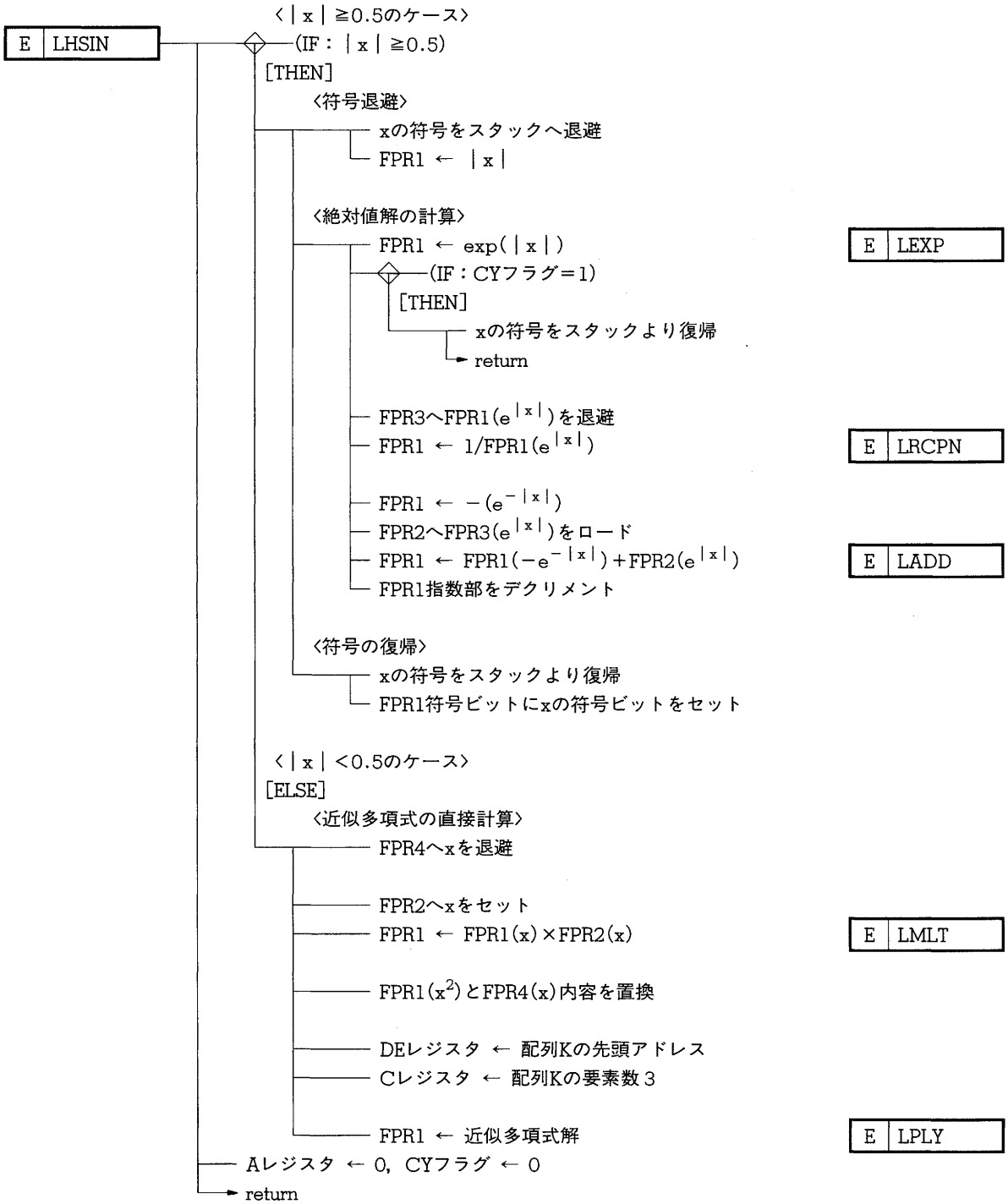
  SINH(x)をTaylor近似式により求めます。

  近似多項式の第1項(x)，各項の係数を除く比(x^2)，および第2項－第4項係数の前項係数に対する比（1/3！，3！/5！，5！/7！）を引数に多項式計算関数をCALLします。

(8) 浮動小数点定数データ

Taylor近似式の係数列に対して，拡張仮数部を持つ定数データ1/3！，3！/5！，5！/7！を要素数3の配列Kとして使用します。

(9) 処 理 図


E | LHSIN
〈|x|≧0.5のケース〉
(IF：|x|≧0.5)
[THEN]
〈符号退避〉
xの符号をスタックへ退避
FPR1 ← |x|
〈絶対値解の計算〉
FPR1 ← exp(|x|)
E | LEXP
(IF：CYフラグ=1)
[THEN]
xの符号をスタックより復帰
return
FPR3へFPR1(e^|x|)を退避
FPR1 ← 1/FPR1(e^|x|)
E | LRCPN
FPR1 ← −(e^−|x|)
FPR2へFPR3(e^|x|)をロード
FPR1 ← FPR1(−e^−|x|)+FPR2(e^|x|)
E | LADD
FPR1指数部をデクリメント
〈符号の復帰〉
xの符号をスタックより復帰
FPR1符号ビットにxの符号ビットをセット
〈|x|<0.5のケース〉
[ELSE]
〈近似多項式の直接計算〉
FPR4へxを退避
FPR2へxをセット
FPR1 ← FPR1(x)×FPR2(x)
E | LMLT
FPR1(x^2)とFPR4(x)内容を置換
DEレジスタ ← 配列Kの先頭アドレス
Cレジスタ ← 配列Kの要素数3
FPR1 ← 近似多項式解
E | LPLY
Aレジスタ ← 0, CYフラグ ← 0
return

4.15　cosh関数（LHCOS）

(1) 処理内容

FPR1の値をxとして，COSH(x)をFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LEXP，LHCOS，LRCPN，FTOL，LTOF

(3) 消費スタック・サイズ

12（LHCOSからの戻り番地２バイトを含む）

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：3.56 ms

最大：5.02 ms（cosh(86.6433868)）

(6) アルゴリズム

次の式により求めます。

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

備考　COSH(−x)＝COSH(x)であることを利用しています。

(7) 処理手順

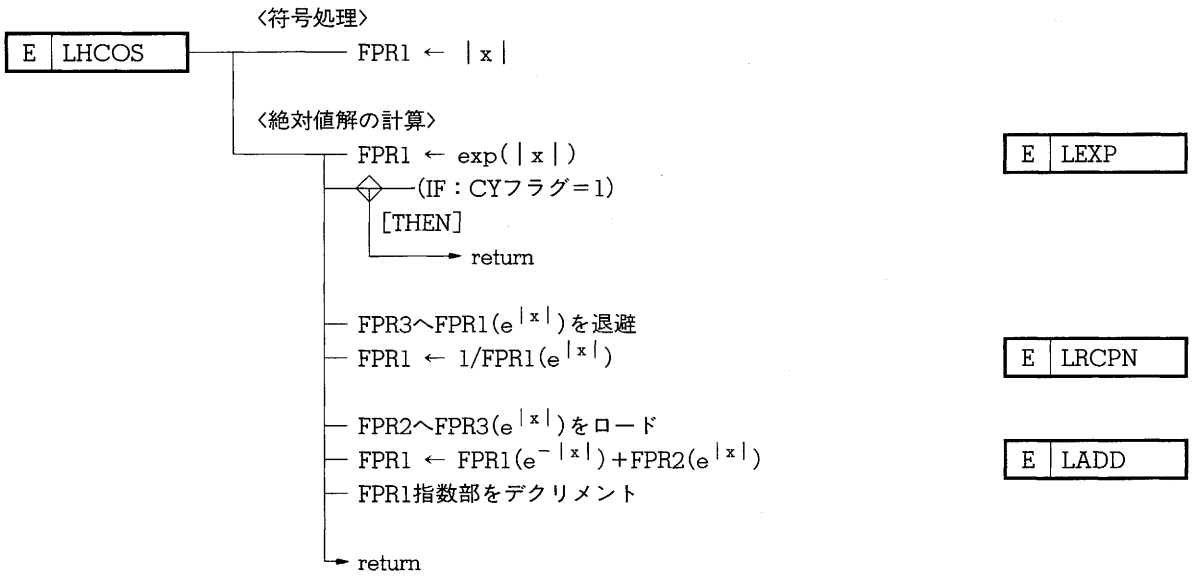
(a) Xの絶対値をとります。

(b) $e^{|x|}$をLEXP関数により求めます。

(c) $e^{|x|}$オーバフローの場合，そのままエラーを返します。

(d) $(e^{|x|}+1/e^{|x|})/2$を求め，解とします。

(8) 処 理 図


E LHCOS
〈符号処理〉
FPR1 ← |x|
〈絶対値解の計算〉
FPR1 ← exp(|x|)
E LEXP
(IF：CYフラグ＝1)
[THEN]
return
FPR3へFPR1(e^|x|)を退避
FPR1 ← 1/FPR1(e^|x|)
E LRCPN
FPR2へFPR3(e^|x|)をロード
FPR1 ← FPR1(e^-|x|)+FPR2(e^|x|)
E LADD
FPR1指数部をデクリメント
return

4.16　tanh関数（LHTAN）

(1) 処理内容

FPR1の値をxとして，TANH(x)をFPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LEXP, LHSIN, LHCOS, LHTAN, LRCPN, FTOL, LTOF

(3) 消費スタック・サイズ

16（LHTANからの戻り番地２バイトを含む）

(4) 使用レジスタ

FPR1, FPR2, FPR3, FPR4, FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：6.26 ms

最大：10.53 ms（tanh(86.6433868)）

(6) アルゴリズム

次の式により求めます。

$$TANH(x)=\frac{SINH(x)}{COSH(x)}$$

(7) 処理手順

(a) COSH(x)をLHCOS関数により求めます。

(b) オーバフローの場合

- xのもとの符号が正なら　1を解として返します。
- xのもとの符号が負なら－1を解として返します。

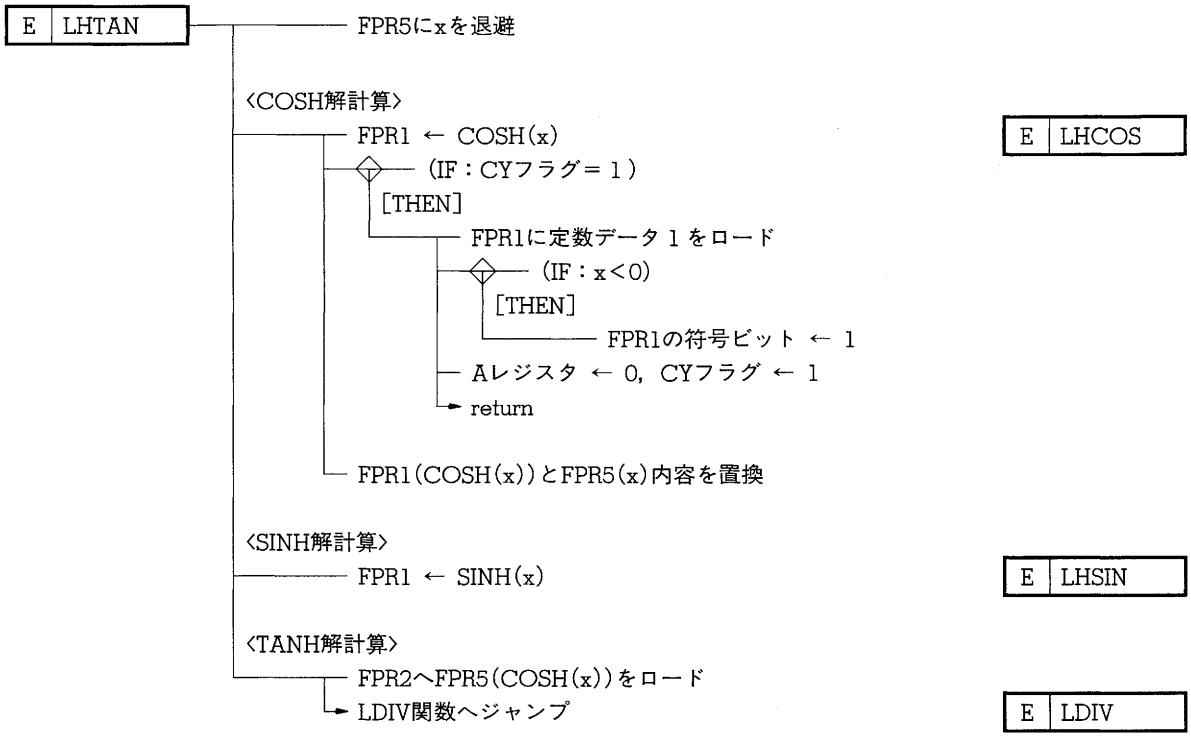
(c) SINH(x)をLHSIN関数により求めます。

(d) SINH(x)/COSH(x)を解とします。

(8) 浮動小数点定数データ

オーバフロー時の解として，定数データ１を使用しています。

(9) 処 理 図


E LHTAN
FPR5にxを退避
〈COSH解計算〉
FPR1 ← COSH(x)
E LHCOS
(IF：CYフラグ＝1)
[THEN]
FPR1に定数データ1をロード
(IF：x<0)
[THEN]
FPR1の符号ビット ← 1
Aレジスタ ← 0, CYフラグ ← 1
return
FPR1(COSH(x))とFPR5(x)内容を置換
〈SINH解計算〉
FPR1 ← SINH(x)
E LHSIN
〈TANH解計算〉
FPR2へFPR5(COSH(x))をロード
LDIV関数へジャンプ
E LDIV

4.17　絶対値関数（LABS）

(1) 処理内容

FPR1の値の絶対値をとり，FPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LABS

(3) 消費スタック・サイズ

2（LABSからの戻り番地2バイトのみ）

(4) 使用レジスタ

FPR1

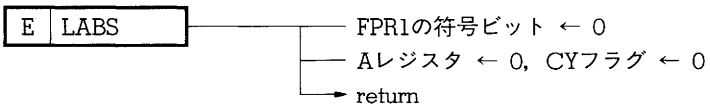
(5) 処理時間（内部システム・クロック：6 MHz）

7.1 μs

(6) 処理手順

FPR1のサイン・ビットを0にします。

(7) 処 理 図



4.18　逆数関数（LRCPN）

(1) 処理内容

FPR1の値の逆数をとり，FPR1に返します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT,, LFLT1, LLD, LRCPN

(3) 消費スタック・サイズ

4（LRCPNからの戻り番地２バイトを含む）

(4) 使用レジスタ

FPR1, FPR2

(5) 処理時間（内部システム・クロック：6 MHz）

平均：534 μs

最大：581 μs（1/6.62484413e+26）

(6) 処理手順

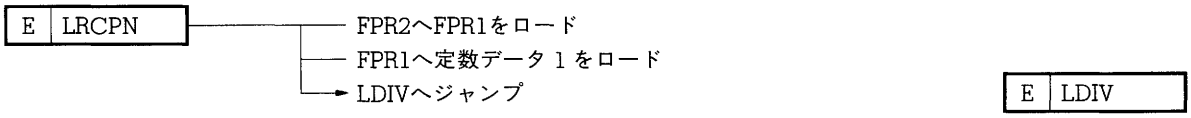
(a) FPR1の値をFPR2へ転送し，FPR1に定数１をセットします。

(b) LDIV関数へジャンプします。

(7) 浮動小数点定数データ

定数データ１を使用しています。

(8) 処 理 図



第５章　座標変換関数

座標変換関数には，次のものを用意しています。

（1）極座標→直交座標への変換関数（POTORA）

極座標値（r，θ）を直交座標値（x，y）へ変換します。

値の受け渡しは，r，xをFPR1により，θ，yをFPR2により行います。

（2）直交座標→極座標への変換関数（RATOPO）

直交座標値（x，y）を極座標値（r，θ）へ変換します。

値の受け渡しは，x，rをFPR1により，y，θをFPR2により行います。

5.1　極座標→直交座標への変換関数（POTORA）

(1) 処理内容

FPR1の値をr，FPR2の値をθとして，極座標（r，θ）を直交座標（x，y）へ変換し，xをFPR1に，yをFPR2に返します。

●θの単位：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSIN，LCOS，POTORA，FTOL，LTOF

(3) 消費スタック・サイズ

16（POTORAからの戻り番地2バイトを含む）

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：　4.74 ms

最大：10.12 ms（r=0.5，θ=1.70141173e+38）

(6) アルゴリズム

次の式により変換します。

$$x=r\times\cos(\theta),\quad y=r\times\sin(\theta)$$

(7) 処理手順

(a) r=0の場合，座標（0，0）を返します。

(b) r<0の場合，エラーを返します。

(c) θをLMOD90関数により，$\theta=\theta'+n\times\pi/2$（nは整数，$0\leqq\theta'<\pi/2$）に変換します。

(d) $\sin(\theta)$をLSIN90関数により，$\cos(\theta)$をLCOS90関数により求めます。

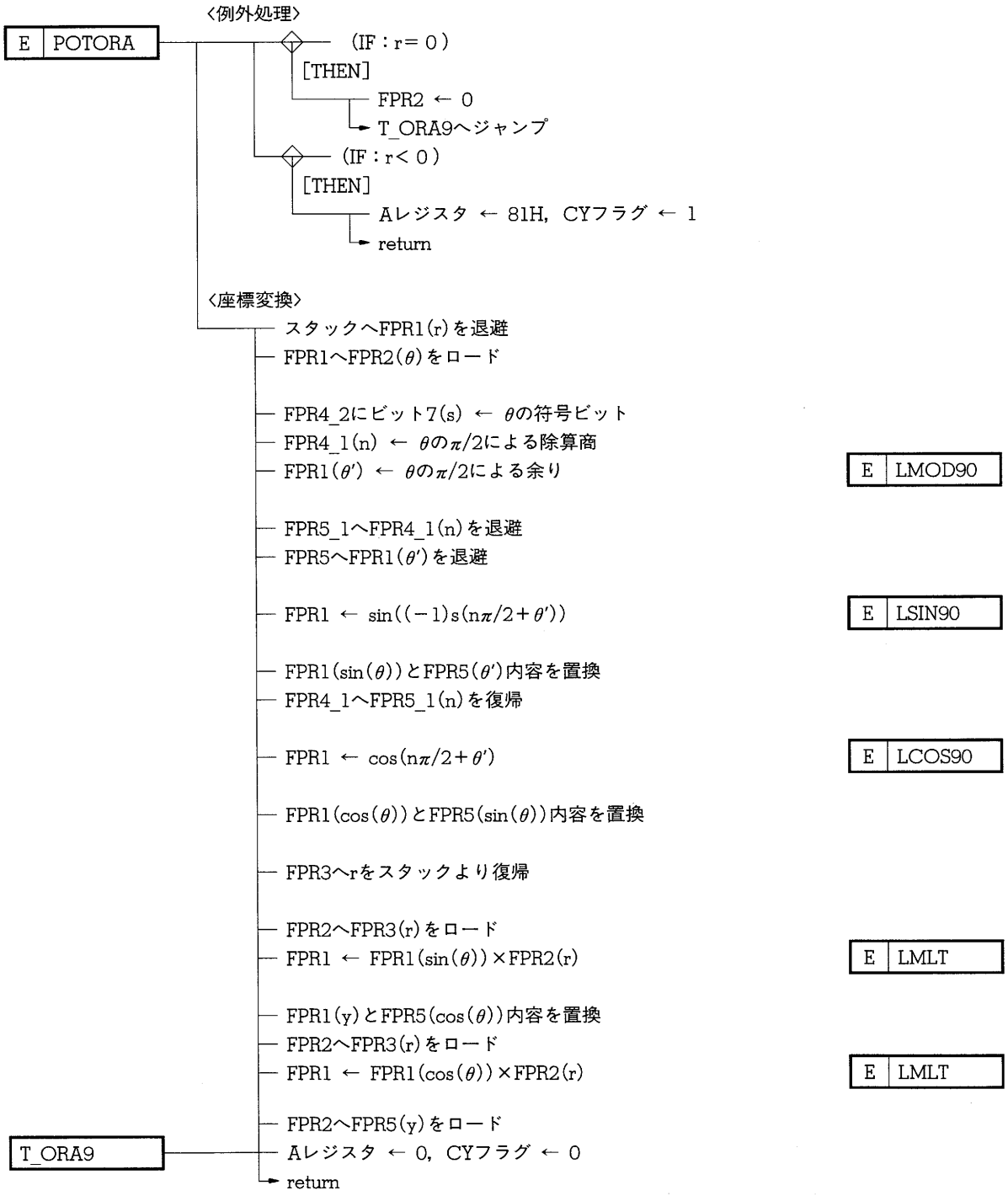
(e) $r\times\cos(\theta)$をx，$r\times\sin(\theta)$をyとします。

(8) 作業エリア

(a) FPR5_1をθの$\pi/2$による除算の商nの退避エリアとして使用します。

(b) rの退避エリアにスタックを使用します。

(9) 処 理 図


〈例外処理〉
E POTORA
(IF：r＝0)
[THEN]
FPR2 ← 0
T_ORA9へジャンプ
(IF：r＜0)
[THEN]
Aレジスタ ← 81H，CYフラグ ← 1
return
〈座標変換〉
スタックへFPR1(r)を退避
FPR1へFPR2(θ)をロード
FPR4_2にビット7(s) ← θの符号ビット
FPR4_1(n) ← θのπ/2による除算商
FPR1(θ') ← θのπ/2による余り
E LMOD90
FPR5_1へFPR4_1(n)を退避
FPR5へFPR1(θ')を退避
FPR1 ← sin((−1)s(nπ/2＋θ'))
E LSIN90
FPR1(sin(θ))とFPR5(θ')内容を置換
FPR4_1へFPR5_1(n)を復帰
FPR1 ← cos(nπ/2＋θ')
E LCOS90
FPR1(cos(θ))とFPR5(sin(θ))内容を置換
FPR3へrをスタックより復帰
FPR2へFPR3(r)をロード
FPR1 ← FPR1(sin(θ))×FPR2(r)
E LMLT
FPR1(y)とFPR5(cos(θ))内容を置換
FPR2へFPR3(r)をロード
FPR1 ← FPR1(cos(θ))×FPR2(r)
E LMLT
FPR2へFPR5(y)をロード
T_ORA9
Aレジスタ ← 0，CYフラグ ← 0
return

5.2　直交座標→極座標への変換関数（RATOPO）

(1) 処理内容

FPR1の値をx，FPR2の値をyとして，直交座標（x，y）を極座標（r，θ）へ変換し，rをFPR1に，θをFPR2に返します。

- 返値θの範囲：$-\pi$～π
- 単位　　：ラジアン

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LSQRT，LATAN，RATOPO

(3) 消費スタック・サイズ

13（RATOPOからの戻り番地2バイトを含む）

(4) 使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：5.33 ms

最大：7.90 ms（x＝−0.5，y＝1）

(6) アルゴリズム

次の2式により求めます。

$$r=\sqrt{(x^2+y^2)}$$
$$\theta=\arctan(y/x)$$

(7) 処理手順

(a) x＝y＝0の場合，そのままリターンします。

(b) x^2+y^2を求めます。

(c) x^2+y^2がオーバフローの場合，エラーを返します。

(d) y/xを求めます。

(e) y/xがオーバフローの場合，次のようになります。

- y＞0なら，$\theta=\pi/2$
- y＜0なら，$\theta=-\pi/2$

(f) y/xが正常終了なら，ARCTAN(y/x)をLATAN関数により求め，θとします。

(g) (f)で，x<0の場合，次のようになります。

- y≧0なら，θにπを加算
- y<0なら，θからπを減算

(h) $\sqrt{(x^2+y^2)}$ を求め，rとします。

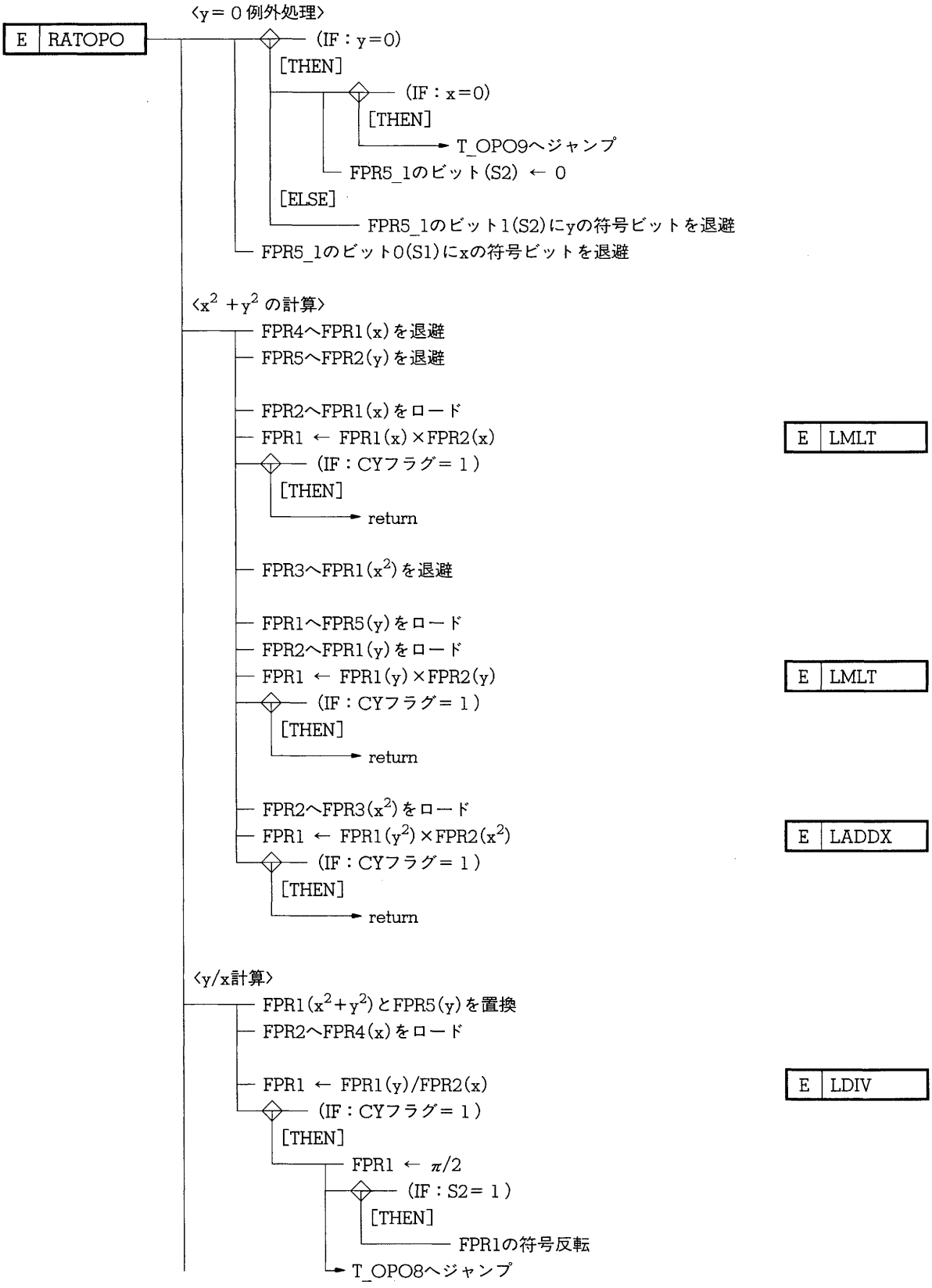
(8) 作業エリア

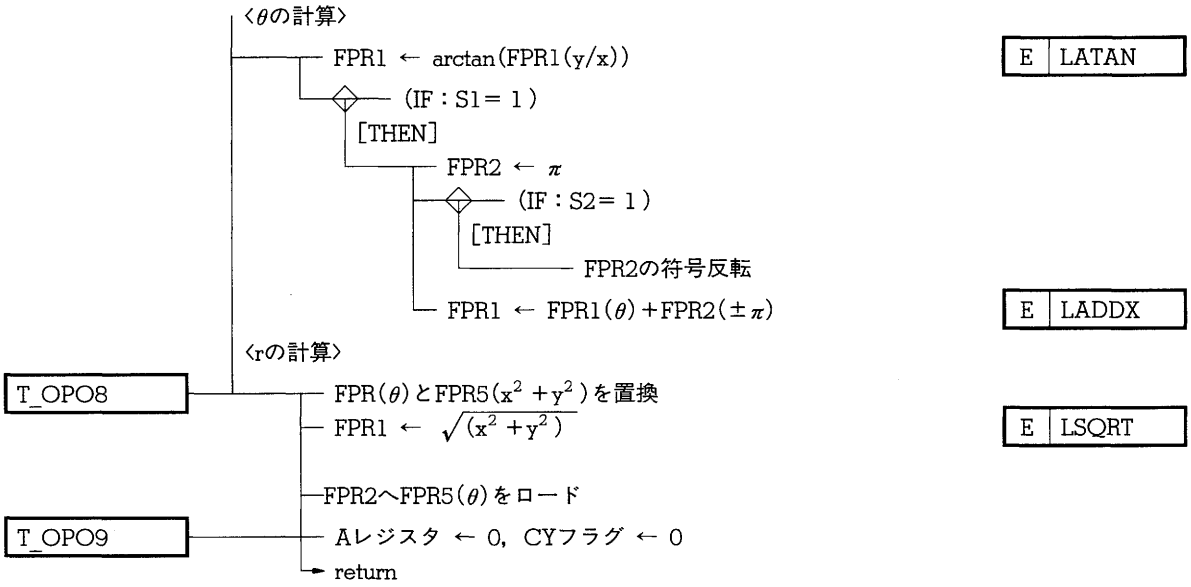
FPR5_1のビット0をx，ビット1をyの符号ビットの退避エリアとして使用します（処理図では，変数名S1，S2で表しています)。

(9) 浮動小数点定数データ

$\pi/2$，およびπ（πは拡張形式）を定数データとして使用しています。

(10) 処 理 図


E RATOPO
〈y＝0 例外処理〉
(IF：y＝0)
[THEN]
(IF：x＝0)
[THEN]
T_OPO9へジャンプ
FPR5_1のビット(S2) ← 0
[ELSE]
FPR5_1のビット1(S2)にyの符号ビットを退避
FPR5_1のビット0(S1)にxの符号ビットを退避
〈x2 ＋y2 の計算〉
FPR4へFPR1(x)を退避
FPR5へFPR2(y)を退避
FPR2へFPR1(x)をロード
FPR1 ← FPR1(x)×FPR2(x)
E LMLT
(IF：CYフラグ＝1)
[THEN]
return
FPR3へFPR1(x2)を退避
FPR1へFPR5(y)をロード
FPR2へFPR1(y)をロード
FPR1 ← FPR1(y)×FPR2(y)
E LMLT
(IF：CYフラグ＝1)
[THEN]
return
FPR2へFPR3(x2)をロード
FPR1 ← FPR1(y2)×FPR2(x2)
E LADDX
(IF：CYフラグ＝1)
[THEN]
return
〈y/x計算〉
FPR1(x2＋y2)とFPR5(y)を置換
FPR2へFPR4(x)をロード
FPR1 ← FPR1(y)/FPR2(x)
E LDIV
(IF：CYフラグ＝1)
[THEN]
FPR1 ← π/2
(IF：S2＝1)
[THEN]
FPR1の符号反転
T_OPO8へジャンプ


〈θの計算〉
FPR1 ← arctan(FPR1(y/x))
(IF : S1 = 1)
[THEN]
FPR2 ← π
(IF : S2 = 1)
[THEN]
FPR2の符号反転
FPR1 ← FPR1(θ) + FPR2(±π)
〈rの計算〉
T_OPO8
FPR(θ)とFPR5(x² + y²)を置換
FPR1 ← √(x² + y²)
FPR2へFPR5(θ)をロード
T_OPO9
Aレジスタ ← 0, CYフラグ ← 0
return
E LATAN
E LADDX
E LSQRT

第６章　型変換関数

型変換関数には，次のものを用意しています。

(1) 文字列→浮動小数点形式への変換関数（ATOL）

先頭アドレスがHLレジスタで示される文字列を浮動小数点形式に変換し，FPR1に格納します。

(2) 浮動小数点形式→文字列への変換関数（LTOA）

FPR1の値を文字列に変換し，HLレジスタが示すアドレスより格納します。

(3) ２バイト整数型→浮動小数点形式への変換関数（FTOL）

DEレジスタ内容を符号付き２バイト整数型として，浮動小数点形式に変換し，FPR1に格納します。

(4) 浮動小数点形式→２バイト整数型への変換関数（LTOF）

FPR1の値を符号付き２バイト整数型へ変換し，DEレジスタに格納します。

(5) IEEE-754形式→78K/II形式への浮動小数点形式変換関数（LTO78）

FPR1の内容をIEEE-754形式から78K/II形式へ変換します。

(6) 78K/II形式→IEEE-754形式への浮動小数点形式変換関数（LTOIE）

FPR1の内容を78K/II形式からIEEE-754形式へ変換します。

６.１　文字列→浮動小数点形式への変換関数（ATOL）

（１）処理内容

先頭アドレスがHLレジスタで示される文字列を浮動小数点形式に変換し，FPR1に返します。

（２）リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LFLT1，LFLT2，LLD，LEXP，ATOL，FTOL，LTOF

（３）消費スタック・サイズ

14（ATOLからの戻り番地２バイトを含む）

（４）使用レジスタ

FPR1，FPR2，FPR3，FPR4，FPR5

（５）処理時間（内部システム・クロック：６MHz）

平均：5.57 ms

最大：9.34 ms（“1.00000000000000000000000000e－37”）

（６）文字列の構成因子

文字列は次の17種の文字によって構成されます。

文字	アスキー・コード
0－9	30H－39H
＋	2BH
－	2DH
.	2EH
E	45H
e	65H
△（スペース）	20H
NUL	00H

(7) 文字列形式

文字列の形式は，次のようになります。

```
[(+/−)] 99……9 [(E/e) [(+/−)] 99]
          |                  |
        仮数部          指数部（底は10）
```

備考　[ ]：省略可

9：0−9

(/)：どちらかを選択

例　“−99.8”　＝−99.8

“. 007e0”　＝. 007

“0998E−03”　＝998×10^{-3}

(8) 文字列の規定

次の規定以外の文字列は，この関数ではエラーとなります。

(a) 文字列の終わりは△（スペース）またはNULで判断する。

(b) 文字列構成因子以外の文字が文字列中に含まれてはならない。

(c) 仮数部文字列の最大長は27で，1つ以下の'.'を含んでよい。
ただし，1つ以上の数字が含まれていなければならない。

(d) 指数部文字列の長さは2または1でなければならない。

(e) 値が2^{127}以上または-2^{127}以下となった場合はエラー

(9) 処理手順

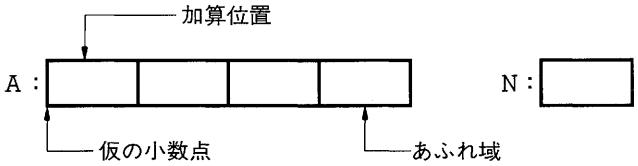
(a) 先頭文字が'－'なら符号を負，他なら正として記憶します。

(b) 仮数部文字列から小数部の桁数を求め，Fとします。

(c) 小数点を除去した仮数部文字列A_1，A_2……A_nより，Aの値を４バイト整数型で求めます。

$$A=(\cdots((A_1\cdot 10+A_2)\cdot 10+A_3)\cdot 10\cdots A_{n-1})\cdot 10+A_n$$

次の方法で計算します。



(i) 初期値をA＝0，N＝0とします。

(ii) A_1を加算位置に加算します。

A_1が存在しなければ，エラーを返します。

(iii) あふれ域＝0の場合

Aに10をかけ，A_kを加算します。

(iv) あふれ域≠0の場合

Nに１を加算しA_kを無視します。

(v) (iii)－(iv) をk＝2－nまで繰り返します。

(vi) n＞27ならエラーを返します。

(d) (c) で求めた仮数部値Aを浮動小数点形式に正規化し，指数部をY_1仮数部（指数部を80Hにしたもの）をX_1とします。

(e) 指数部文字列'(＋/－) B_1B_2'より，指数部値Bを求めます。

(f) 指数部値Bに，小数部の桁数と無視した仮数部の桁数を加えた実質指数値B'(＝B－F＋N)を求めます。

(g) 以上で求めた，X_1，Y_1，B'により，解を次の式で計算します。

$$\begin{aligned}&(-1)^{(符号)}\times X_1\times 2^{Y_1}\times 10^{B'}\\=&(-1)^{(符号)}\times X_1\times 2^{\log_2 10\times B'+Y_1}\\=&(-1)^{(符号)}\times X_1\times 2^{\mathrm{dec}\,(\log_2 10\times B')}\times 2^{\mathrm{int}\,(\log_2 10\times B')+Y_1}\\=&(-1)^{(符号)}\times X_1\times e^{\log 2\times \mathrm{dec}\,(\log_2 10\times B')}\times 2^{\mathrm{int}\,(\log_2 10\times B')+Y_1}\end{aligned}$$

備考　dec(x) はxの小数部，int (x) はxの整数部を表します。

(10) 作業エリア

(a) FPR5_1のビット７を符号の退避エリアとして使用します(処理図では,sという記号で表しています)。

(b) 仮数部値の計算部において，次のように使用します。

- FPR4_1を，小数部の桁数Fのカウンタとして使用
- FPR4_2を，無視した桁数Nカウンタとして使用
- Xレジスタを，仮数部桁数のカウンタとして使用

（処理図では，記号DC1，DC2，DC3で表しています)。

DEHLレジスタを仮数部文字列の２進化変換用ワークAとして使用します。

(c) 仮数部の正規化部以降では，次のように使用します。

- FPR4_1を，F_N（負数は２の補数形式）の退避エリアとして使用
- FPR4_2を正規化した10進仮数部値の指数部Y_1の退避エリアとして使用

(d) 文字判定のためのINDEXストリングは次のとおりです。

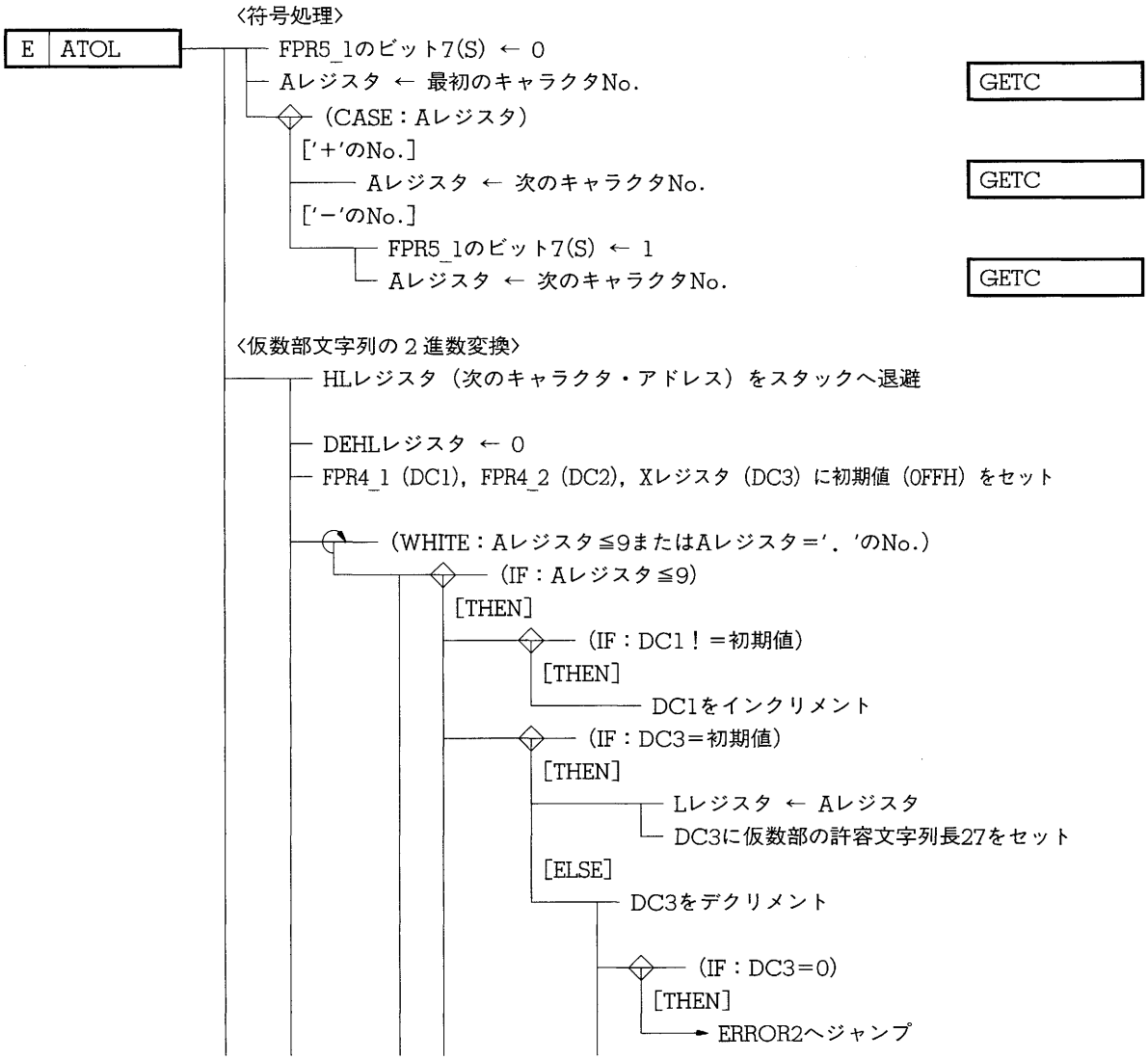
```
'0123456789eE△NUL. －＋'
```

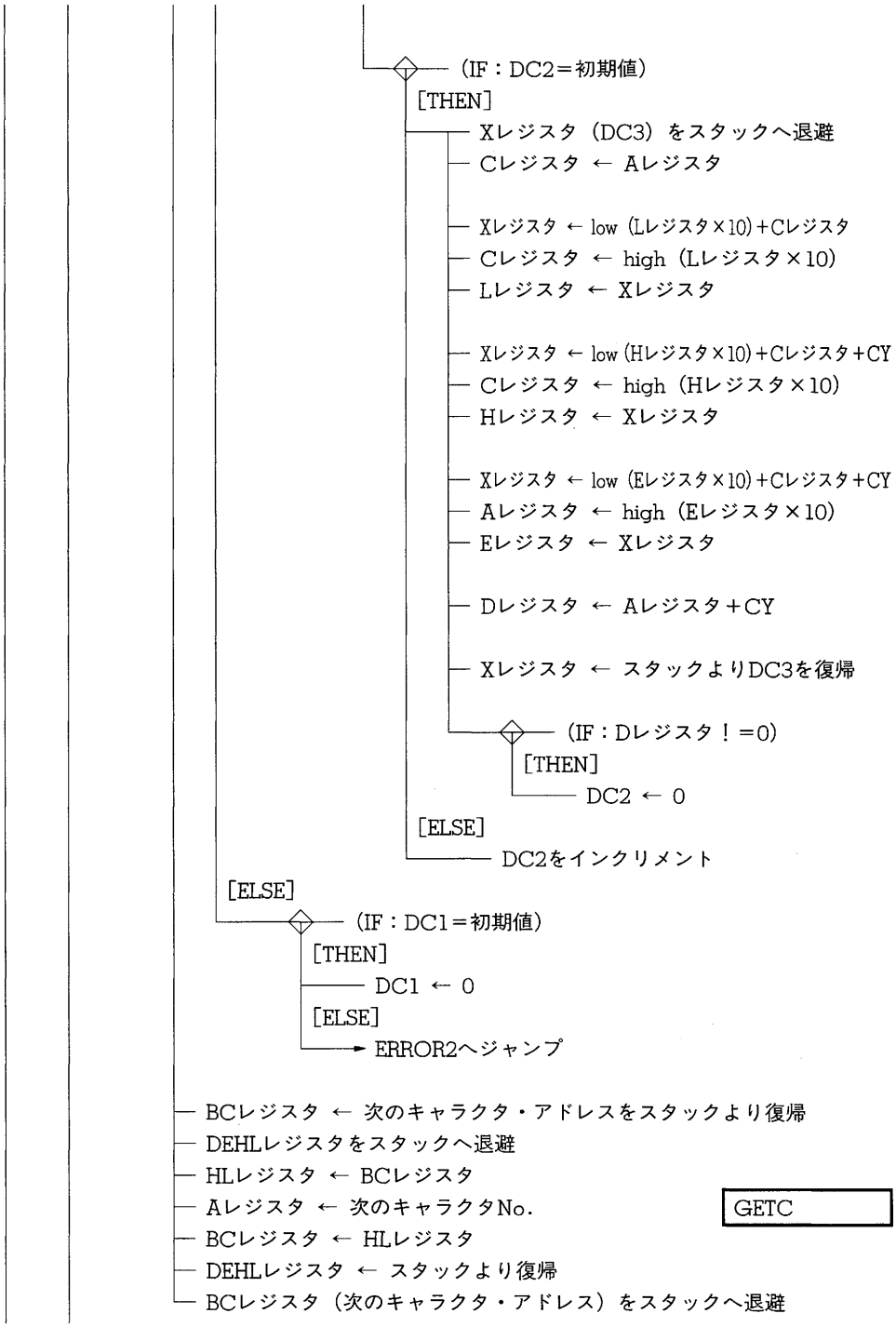
内部サブルーチンGETCは入力文字のINDEXストリング中の位置を得るための関数で,INDEXストリングの左端からの位置No.0－16とINDEXエラーの場合0FFHを返します。

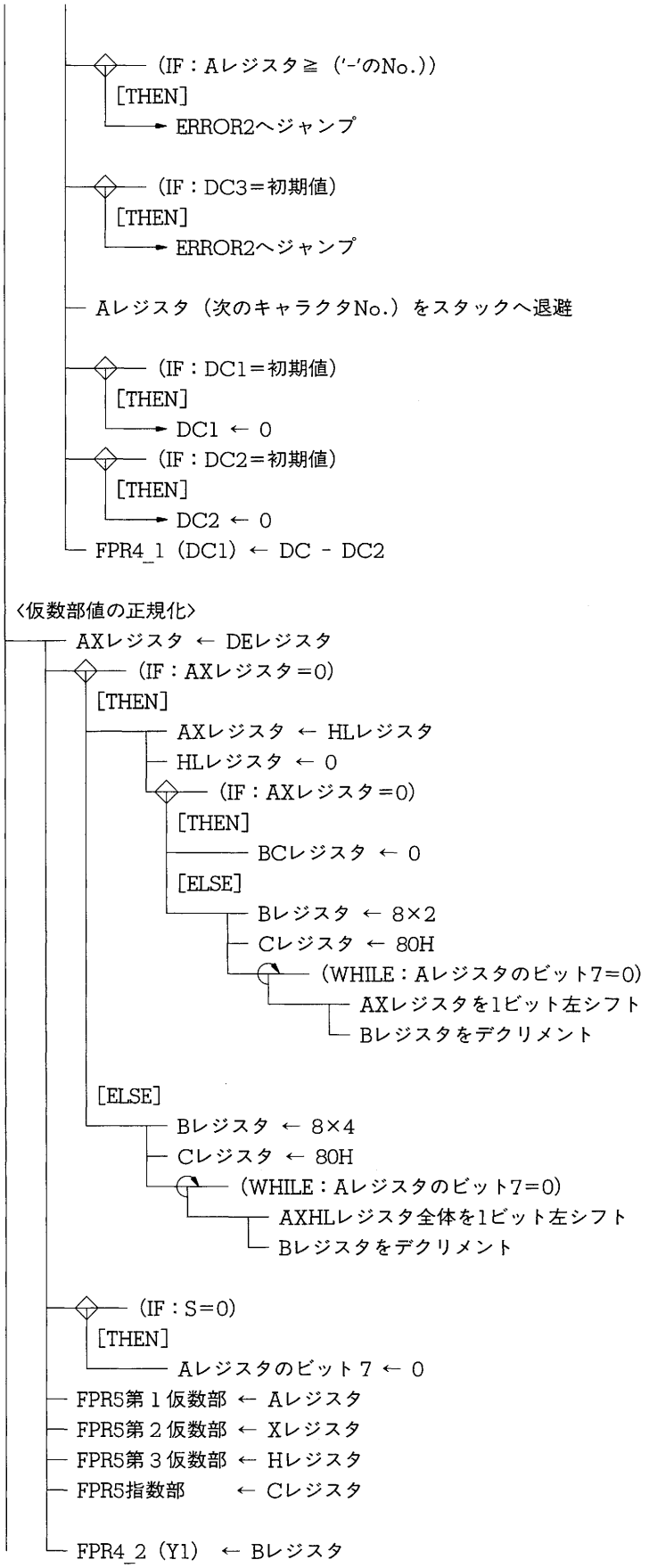
(11) 浮動小数点定数データ

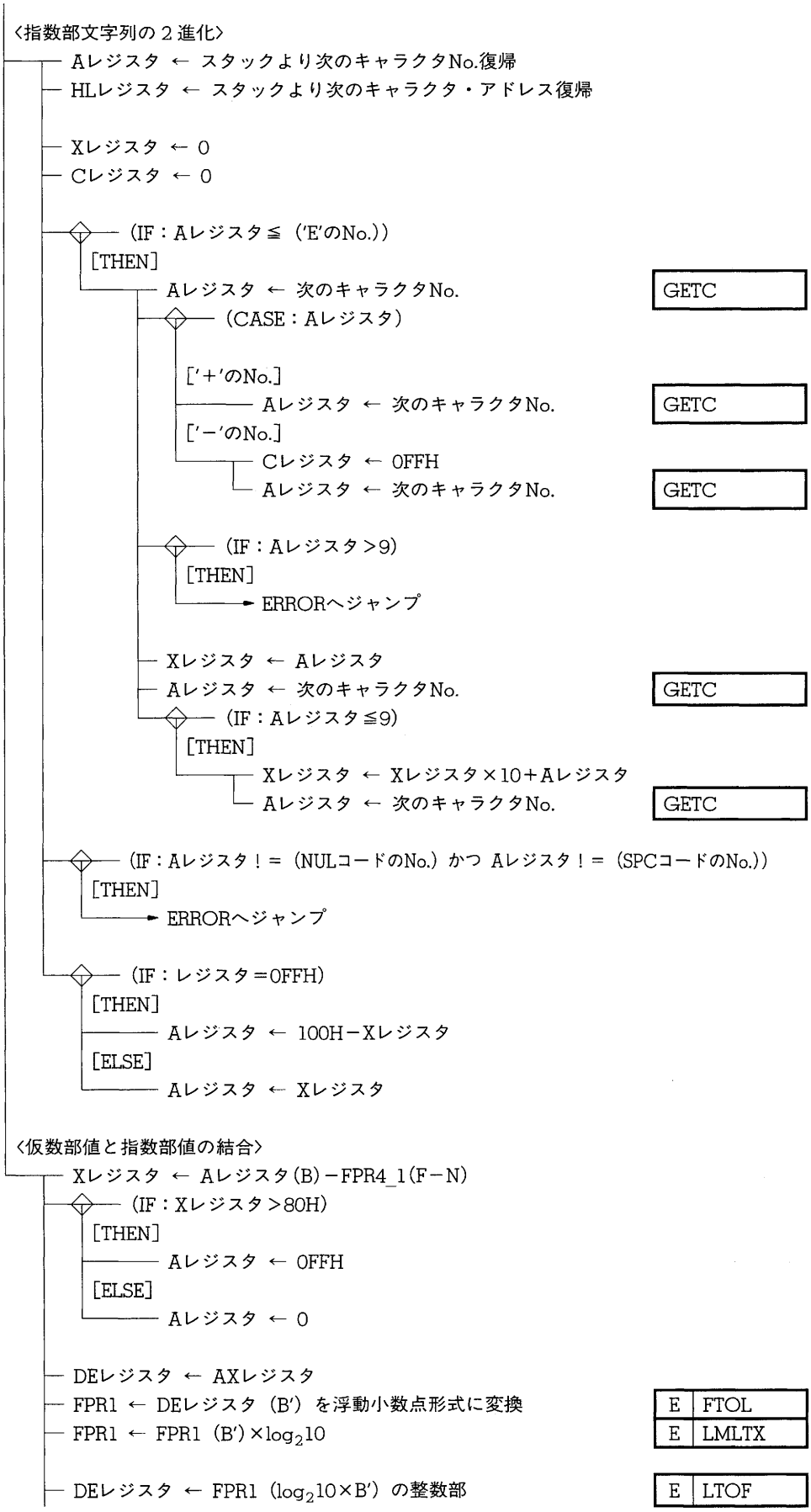
拡張仮数部を持つ定数データ$\log_2 10$と$\log 2$を使用しています。

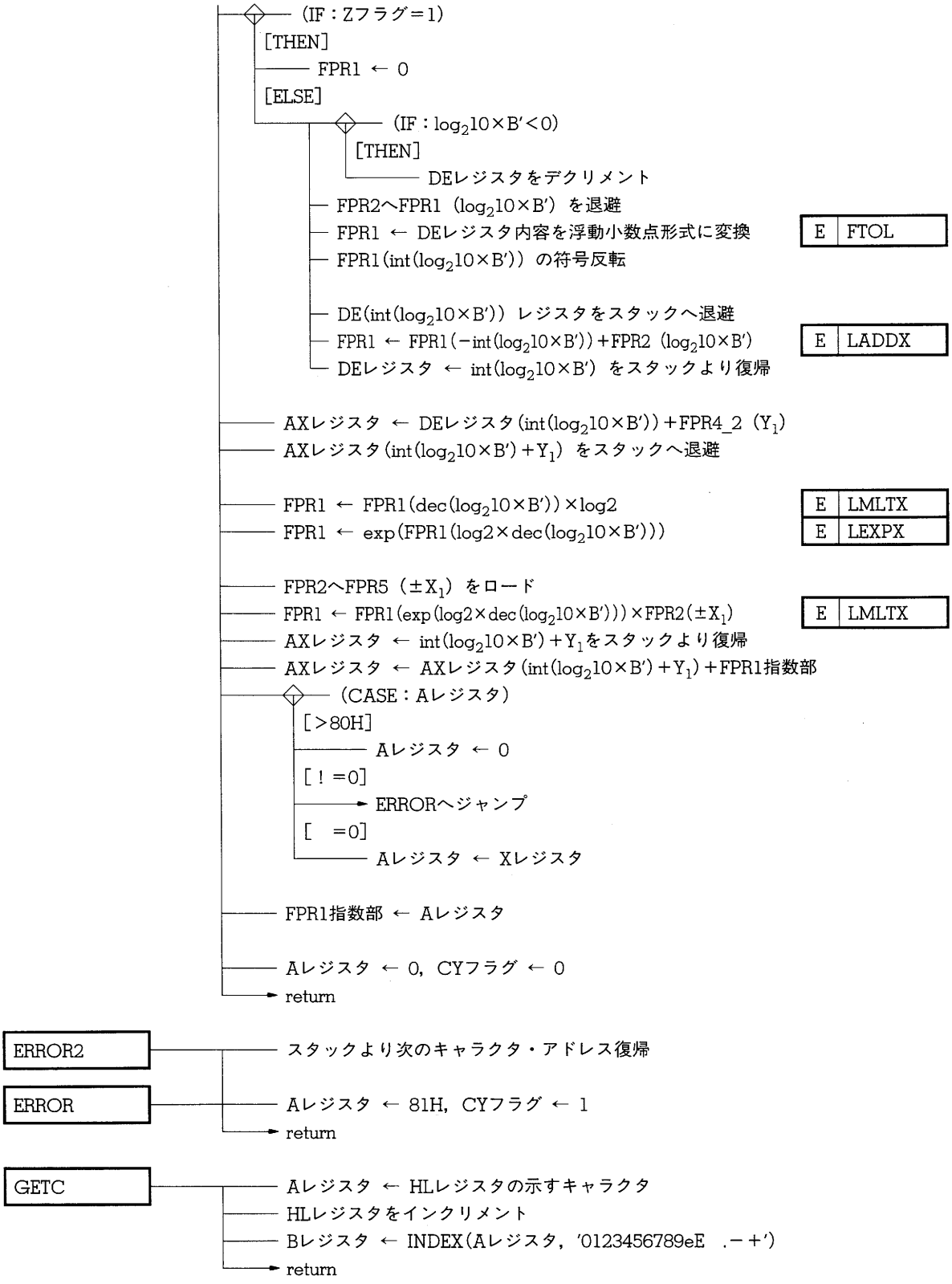
(12) 処 理 図


E ATOL
〈符号処理〉
FPR5_1のビット7(S) ← 0
Aレジスタ ← 最初のキャラクタNo.
GETC
(CASE：Aレジスタ)
['+'のNo.]
Aレジスタ ← 次のキャラクタNo.
GETC
['−'のNo.]
FPR5_1のビット7(S) ← 1
Aレジスタ ← 次のキャラクタNo.
GETC
〈仮数部文字列の２進数変換〉
HLレジスタ（次のキャラクタ・アドレス）をスタックへ退避
DEHLレジスタ ← 0
FPR4_1 (DC1), FPR4_2 (DC2), Xレジスタ (DC3) に初期値 (0FFH) をセット
(WHITE：Aレジスタ≦9またはAレジスタ='．'のNo.)
(IF：Aレジスタ≦9)
[THEN]
(IF：DC1！＝初期値)
[THEN]
DC1をインクリメント
(IF：DC3＝初期値)
[THEN]
Lレジスタ ← Aレジスタ
DC3に仮数部の許容文字列長27をセット
[ELSE]
DC3をデクリメント
(IF：DC3＝0)
[THEN]
ERROR2へジャンプ


(IF：DC2＝初期値)
[THEN]
Xレジスタ（DC3）をスタックへ退避
Cレジスタ ← Aレジスタ
Xレジスタ ← low（Lレジスタ×10)+Cレジスタ
Cレジスタ ← high（Lレジスタ×10)
Lレジスタ ← Xレジスタ
Xレジスタ ← low（Hレジスタ×10)+Cレジスタ+CY
Cレジスタ ← high（Hレジスタ×10)
Hレジスタ ← Xレジスタ
Xレジスタ ← low（Eレジスタ×10)+Cレジスタ+CY
Aレジスタ ← high（Eレジスタ×10)
Eレジスタ ← Xレジスタ
Dレジスタ ← Aレジスタ+CY
Xレジスタ ← スタックよりDC3を復帰
(IF：Dレジスタ！=0)
[THEN]
DC2 ← 0
[ELSE]
DC2をインクリメント
[ELSE]
(IF：DC1＝初期値)
[THEN]
DC1 ← 0
[ELSE]
ERROR2へジャンプ
BCレジスタ ← 次のキャラクタ・アドレスをスタックより復帰
DEHLレジスタをスタックへ退避
HLレジスタ ← BCレジスタ
Aレジスタ ← 次のキャラクタNo.
BCレジスタ ← HLレジスタ
DEHLレジスタ ← スタックより復帰
BCレジスタ（次のキャラクタ・アドレス）をスタックへ退避
GETC


(IF：Aレジスタ≧('-'のNo.))
[THEN]
ERROR2へジャンプ
(IF：DC3＝初期値)
[THEN]
ERROR2へジャンプ
Aレジスタ（次のキャラクタNo.）をスタックへ退避
(IF：DC1＝初期値)
[THEN]
DC1 ← 0
(IF：DC2＝初期値)
[THEN]
DC2 ← 0
FPR4_1 (DC1) ← DC - DC2
〈仮数部値の正規化〉
AXレジスタ ← DEレジスタ
(IF：AXレジスタ＝0)
[THEN]
AXレジスタ ← HLレジスタ
HLレジスタ ← 0
(IF：AXレジスタ＝0)
[THEN]
BCレジスタ ← 0
[ELSE]
Bレジスタ ← 8×2
Cレジスタ ← 80H
(WHILE：Aレジスタのビット7＝0)
AXレジスタを1ビット左シフト
Bレジスタをデクリメント
[ELSE]
Bレジスタ ← 8×4
Cレジスタ ← 80H
(WHILE：Aレジスタのビット7＝0)
AXHLレジスタ全体を1ビット左シフト
Bレジスタをデクリメント
(IF：S＝0)
[THEN]
Aレジスタのビット7 ← 0
FPR5第1仮数部 ← Aレジスタ
FPR5第2仮数部 ← Xレジスタ
FPR5第3仮数部 ← Hレジスタ
FPR5指数部 ← Cレジスタ
FPR4_2 (Y1) ← Bレジスタ


〈指数部文字列の２進化〉
Aレジスタ ← スタックより次のキャラクタNo.復帰
HLレジスタ ← スタックより次のキャラクタ・アドレス復帰
Xレジスタ ← 0
Cレジスタ ← 0
(IF：Aレジスタ≦('E'のNo.))
[THEN]
Aレジスタ ← 次のキャラクタNo.
GETC
(CASE：Aレジスタ)
['+'のNo.]
Aレジスタ ← 次のキャラクタNo.
GETC
['−'のNo.]
Cレジスタ ← 0FFH
Aレジスタ ← 次のキャラクタNo.
GETC
(IF：Aレジスタ>9)
[THEN]
ERRORへジャンプ
Xレジスタ ← Aレジスタ
Aレジスタ ← 次のキャラクタNo.
GETC
(IF：Aレジスタ≦9)
[THEN]
Xレジスタ ← Xレジスタ×10+Aレジスタ
Aレジスタ ← 次のキャラクタNo.
GETC
(IF：Aレジスタ！=(NULコードのNo.) かつ Aレジスタ！=(SPCコードのNo.))
[THEN]
ERRORへジャンプ
(IF：レジスタ=0FFH)
[THEN]
Aレジスタ ← 100H−Xレジスタ
[ELSE]
Aレジスタ ← Xレジスタ
〈仮数部値と指数部値の結合〉
Xレジスタ ← Aレジスタ(B)−FPR4_1(F−N)
(IF：Xレジスタ>80H)
[THEN]
Aレジスタ ← 0FFH
[ELSE]
Aレジスタ ← 0
DEレジスタ ← AXレジスタ
FPR1 ← DEレジスタ (B') を浮動小数点形式に変換
E FTOL
FPR1 ← FPR1 (B')×log2 10
E LMLTX
DEレジスタ ← FPR1 (log2 10×B') の整数部
E LTOF



備考1. high（ ）は，乗算結果の上位バイトを示します。

2. low（ ）は，乗算結果の下位バイトを示します。

6.2　浮動小数点形式→文字列への変換関数（LTOA）

(1) 処理内容

FPR1の値を文字列に変換し，HLレジスタが示すアドレスより格納します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LFLT1, LFLT2, LLD, LLOG, LLOG10, LEXP, LEXP10, LTOA, FTOL, LTOF

(3) 消費スタック・サイズ

16（LTOAからの戻り番地２バイトを含む）

(4) 使用レジスタ

FPR1, FPR2, FPR3, FPR4, FPR5

(5) 処理時間（内部システム・クロック：6 MHz）

平均：9.87 ms

最大：11.97 ms（6E120377）

(6) 出力文字列の形式

```
形式1：［−］9.999999E［−］99
                  仮数部    指数部
形式2：0
```

規定　(a) 0の場合にのみ形式２を出力し，他はすべて形式１を用います。

(b) 文字列の終わりに'NUL'コードが付加されます。

(c) 形式１の仮数部，および指数部の文字列長は，それぞれ８，２で固定となります。

(d) 形式１の仮数部，および指数部のサインは，負の場合にのみ付加されます。

(e) 'NUL'コードを含めて，最大文字列長は14となります。

(7) 処理手順

(a) 浮動小数点値X＝0なら，文字列'0NUL'を出力し終了します。

(b) 次式により，Xを$a\times10^{b}$に変換します。

$$b=\mathrm{floor}(\log_{10}(|X|)),\ b>-39 \rightarrow a=X\times10^{-b}$$

$$b=-39 \text{ or } -40 \rightarrow a=(X\times10^{2})\times10^{-b-2}$$

ここで，floor（X）はXから負の方向へ向かって一番近い整数を示します。

bが−39以下のとき，10^{-b}を直接計算しないのは，10^{39}が78K/IIの浮動小数点系ではオーバフローになってしまうからです。

(c) a<0なら'−'を出力し，a←|a|とします。

(d) aは数学的には1≦a<10ですが，実際には計算誤差により，a<1またはa≧10となり得ます。この場合，次の補正を行います。

a<1なら，a←a×10,　b←b−1
a≧10なら，a←a/10,　b←b+1

(e) 1≦a<10のため，小数点位置は固定となります。

下に示す計算結果より，文字列'A_1，A_2……A_7'を出力します。

A_1=int(a),　a=dec(a)×10
A_2=int(a),　a=dec(a)×10
……
A_7=int(a)

ここで，int(a) はaの整数部を，dec(a) はaの小数部を示します。

(f) 'E'を出力します。

(g) b<0なら'−'を出力し，b←|b|とします。

(h) bの文字列'B_1B_2'（B_1=b/10，B_2=b−B_1×10）を出力します。

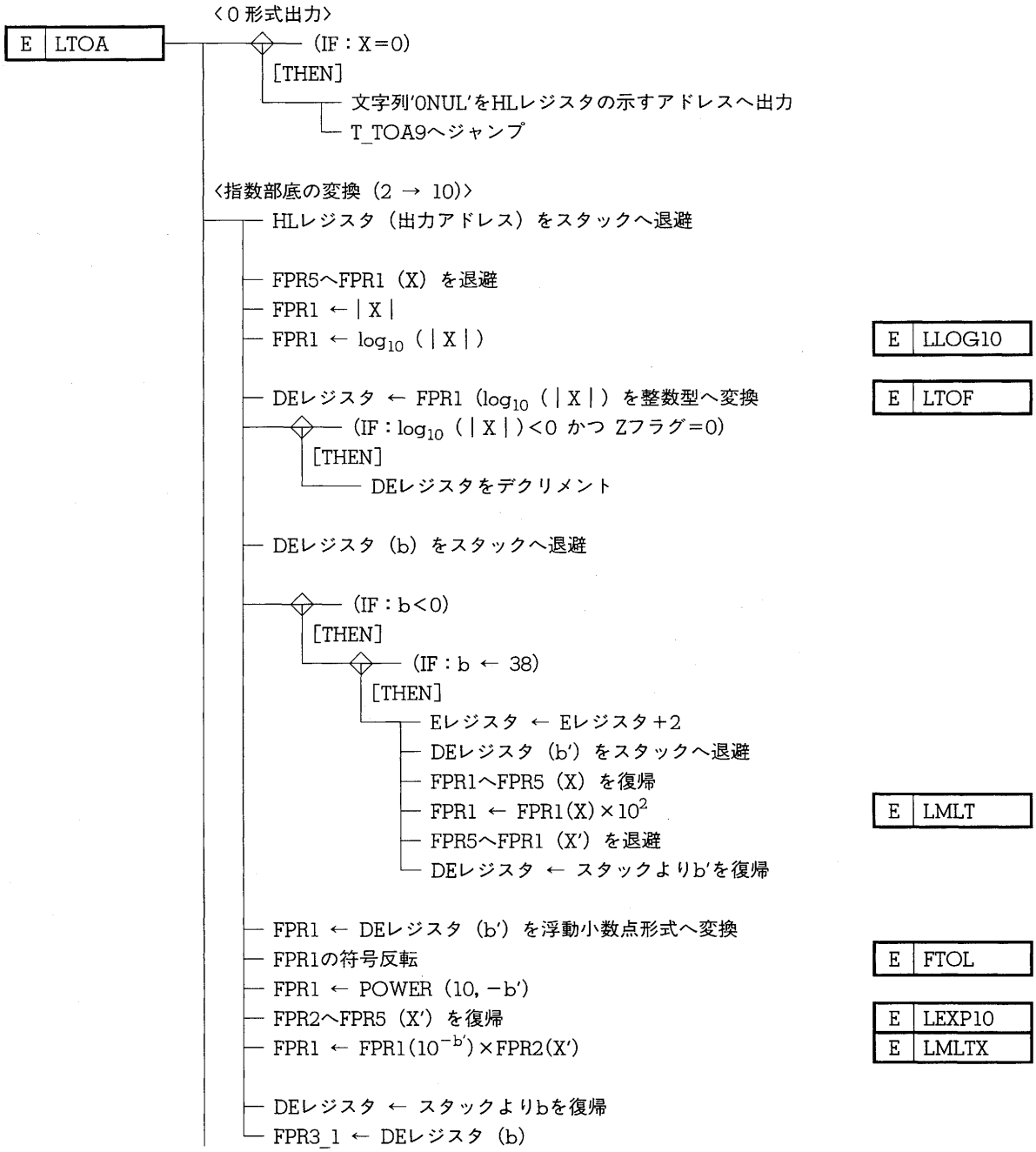
(i) 'NUL'コードを出力します。

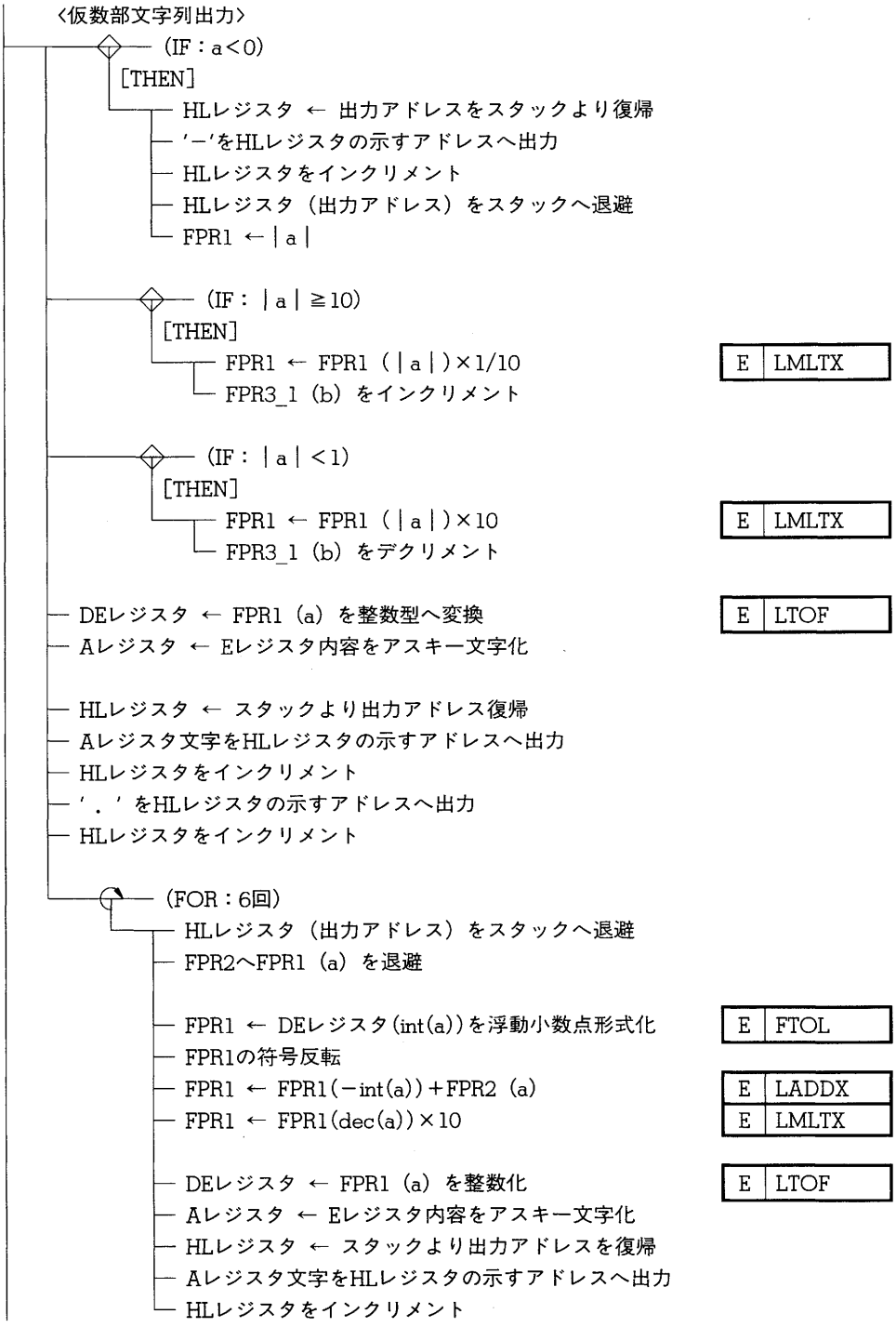
(8) 浮動小数点定数データ

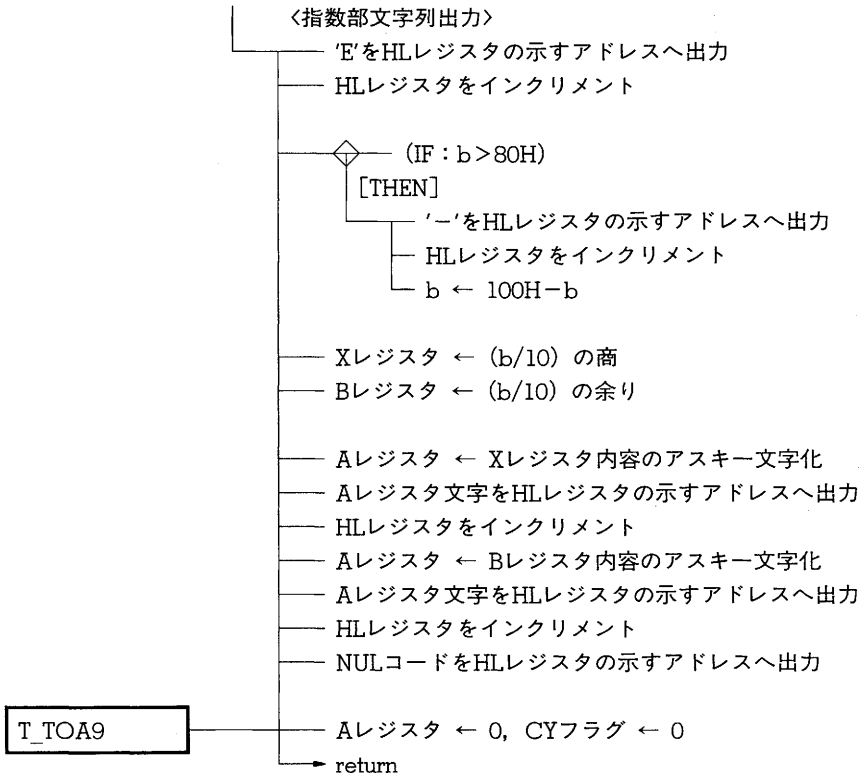
(a) 定数データ100を使用しています。

(b) 拡張仮数部を持つ定数データ10，1/10を使用しています。

(9) 処 理 図


E LTOA
〈0 形式出力〉
(IF : X=0)
[THEN]
文字列'0NUL'をHLレジスタの示すアドレスへ出力
T_TOA9へジャンプ
〈指数部底の変換 (2 → 10)〉
HLレジスタ (出力アドレス) をスタックへ退避
FPR5へFPR1 (X) を退避
FPR1 ← | X |
FPR1 ← log10 (| X |)
E LLOG10
DEレジスタ ← FPR1 (log10 (| X |) を整数型へ変換
E LTOF
(IF : log10 (| X |)<0 かつ Zフラグ=0)
[THEN]
DEレジスタをデクリメント
DEレジスタ (b) をスタックへ退避
(IF : b<0)
[THEN]
(IF : b ← 38)
[THEN]
Eレジスタ ← Eレジスタ+2
DEレジスタ (b') をスタックへ退避
FPR1へFPR5 (X) を復帰
FPR1 ← FPR1(X)×10^2
E LMLT
FPR5へFPR1 (X') を退避
DEレジスタ ← スタックよりb'を復帰
FPR1 ← DEレジスタ (b') を浮動小数点形式へ変換
FPR1の符号反転
E FTOL
FPR1 ← POWER (10, −b')
FPR2へFPR5 (X') を復帰
E LEXP10
FPR1 ← FPR1(10^−b')×FPR2(X')
E LMLTX
DEレジスタ ← スタックよりbを復帰
FPR3_1 ← DEレジスタ (b)


〈仮数部文字列出力〉
(IF：a<0)
[THEN]
HLレジスタ ← 出力アドレスをスタックより復帰
'−'をHLレジスタの示すアドレスへ出力
HLレジスタをインクリメント
HLレジスタ（出力アドレス）をスタックへ退避
FPR1 ←|a|
(IF：|a|≧10)
[THEN]
FPR1 ← FPR1（|a|）×1/10
FPR3_1（b）をインクリメント
E LMLTX
(IF：|a|<1)
[THEN]
FPR1 ← FPR1（|a|）×10
FPR3_1（b）をデクリメント
E LMLTX
DEレジスタ ← FPR1（a）を整数型へ変換
E LTOF
Aレジスタ ← Eレジスタ内容をアスキー文字化
HLレジスタ ← スタックより出力アドレス復帰
Aレジスタ文字をHLレジスタの示すアドレスへ出力
HLレジスタをインクリメント
'．'をHLレジスタの示すアドレスへ出力
HLレジスタをインクリメント
(FOR：6回)
HLレジスタ（出力アドレス）をスタックへ退避
FPR2へFPR1（a）を退避
FPR1 ← DEレジスタ(int(a))を浮動小数点形式化
E FTOL
FPR1の符号反転
FPR1 ← FPR1(−int(a))+FPR2（a）
E LADDX
FPR1 ← FPR1(dec(a))×10
E LMLTX
DEレジスタ ← FPR1（a）を整数化
E LTOF
Aレジスタ ← Eレジスタ内容をアスキー文字化
HLレジスタ ← スタックより出力アドレスを復帰
Aレジスタ文字をHLレジスタの示すアドレスへ出力
HLレジスタをインクリメント


〈指数部文字列出力〉
'E'をHLレジスタの示すアドレスへ出力
HLレジスタをインクリメント
(IF：b>80H)
[THEN]
'−'をHLレジスタの示すアドレスへ出力
HLレジスタをインクリメント
b ← 100H−b
Xレジスタ ← (b/10) の商
Bレジスタ ← (b/10) の余り
Aレジスタ ← Xレジスタ内容のアスキー文字化
Aレジスタ文字をHLレジスタの示すアドレスへ出力
HLレジスタをインクリメント
Aレジスタ ← Bレジスタ内容のアスキー文字化
Aレジスタ文字をHLレジスタの示すアドレスへ出力
HLレジスタをインクリメント
NULコードをHLレジスタの示すアドレスへ出力
T_TOA9
Aレジスタ ← 0, CYフラグ ← 0
return

6.3　2バイト整数型→浮動小数点形式への変換関数（FTOL）

(1) 処理内容

DEレジスタ・ペア内容を符号付き2バイト整数型として浮動小数点形式に変換し，FPR1に返します（DEレジスタ内容は保存されます）。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，FTOL

(3) 消費スタック・サイズ

2（FTOLからの戻り番地2バイトのみ）

(4) 使用レジスタ

FPR1

(5) 処理時間（内部システム・クロック：6 MHz）

平均：41 μs

最大：72 μs（−1）

(6) 2バイト整数型形式

最上位ビットが符号ビットで，負数は2の補数表現とします。

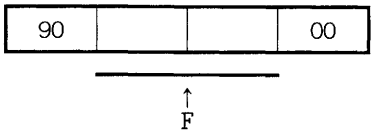
2バイト整数型Fは，−8000H〜+7FFFHの範囲の整数値をとります。

(7) 処理手順

(a) 2バイト整数値F=0なら，0を返します。

(b) −7FFFH≦F<0なら，Fの2の補数をとります。

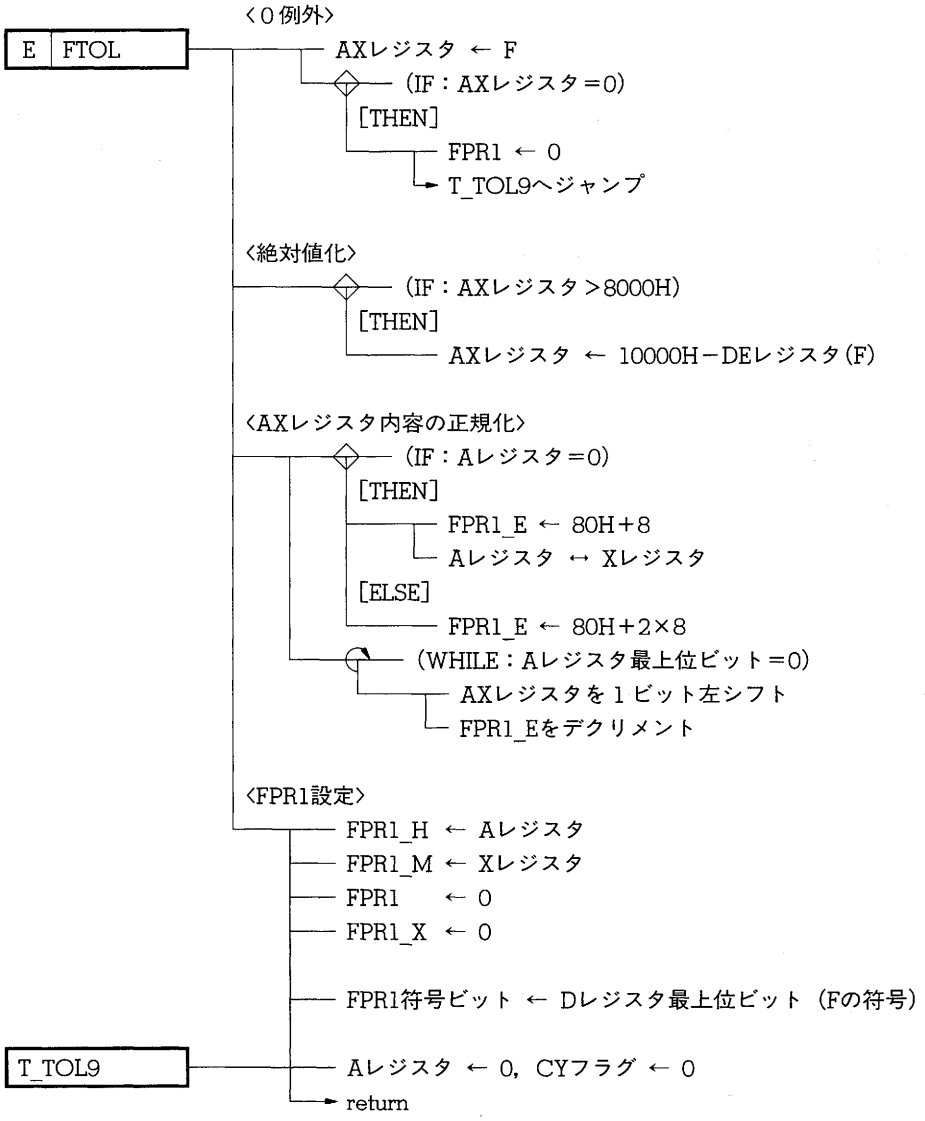
(c) 下図の方法でFを浮動小数点形式に変換します。



→ 正規化
（サイン・ビットの代わりにMSBを持つ形で行います）。

(d) もとのFの符合を浮動小数点形式に埋め込みます。

(8) 処 理 図


〈0例外〉
E FTOL
AXレジスタ ← F
(IF：AXレジスタ=0)
[THEN]
FPR1 ← 0
T_TOL9へジャンプ
〈絶対値化〉
(IF：AXレジスタ>8000H)
[THEN]
AXレジスタ ← 10000H−DEレジスタ(F)
〈AXレジスタ内容の正規化〉
(IF：Aレジスタ=0)
[THEN]
FPR1_E ← 80H+8
Aレジスタ ↔ Xレジスタ
[ELSE]
FPR1_E ← 80H+2×8
(WHILE：Aレジスタ最上位ビット=0)
AXレジスタを1ビット左シフト
FPR1_Eをデクリメント
〈FPR1設定〉
FPR1_H ← Aレジスタ
FPR1_M ← Xレジスタ
FPR1 ← 0
FPR1_X ← 0
FPR1符号ビット ← Dレジスタ最上位ビット（Fの符号）
T_TOL9
Aレジスタ ← 0, CYフラグ ← 0
return

6.4　浮動小数点形式→２バイト整数型への変換関数（LTOF）

(1) 処理内容

FPR1の値を符号付き２バイト整数型へ変換し，DEレジスタに返します。

また，FPR1に小数部が含まれなかったならZフラグ＝1を，変換過程で小数部の切り捨てを行った場合，Zフラグ＝0を返します（FPR1の内容は保存されます）。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LTOF

(3) 消費スタック・サイズ

２（LTOFからの戻り番地２バイトのみ）

(4) 使用レジスタ

FPR1

(5) 処理時間（内部システム・クロック：６MHz）

平均：61 μs

最大：77 μs（−1）

(6) 処理手順

(a) 指数部＝0なら，整数値０とZフラグ１を返します。

指数部≦80Hなら，整数値０とZフラグ０を返します。

指数部＞90Hなら，エラーを返します。

(b) 整数部をunsigned Integer形式で取り出します。

MSBをセットしておきます。

指数部≦88Hなら第１仮数部を（88H−指数部）ビット右シフトし，整数値を得ます。

指数部＞88Hなら第１−第２仮数部を（90H−指数部）ビット右シフトし，整数値を得ます。

(c) unsigned Integer→Integerへ変換します。

(b) で求めた整数値について，

8000Hより大きい　→エラー

8000Hかつ正　　→エラー

8000Hかつ負　　→そのまま

8000H未満かつ正　→そのまま

8000H未満かつ負　→２の補数をとり，解とします。

(d) (b) において，

○Zフラグ１を返す場合

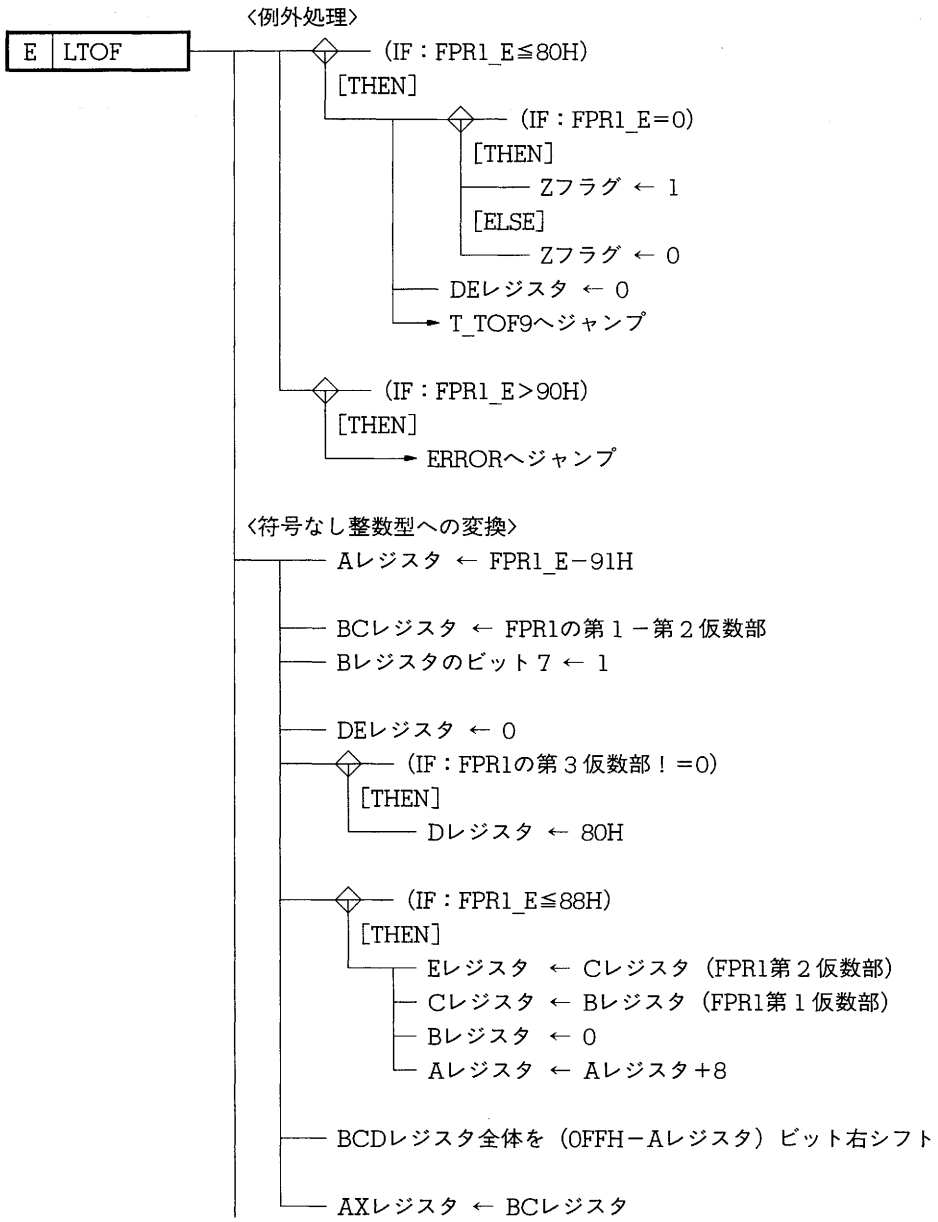
●指数部≦88Hのケースで，第２－第３仮数部の全ビットが０かつ第１仮数部の右シフト過程でキャリーがでなかったとき

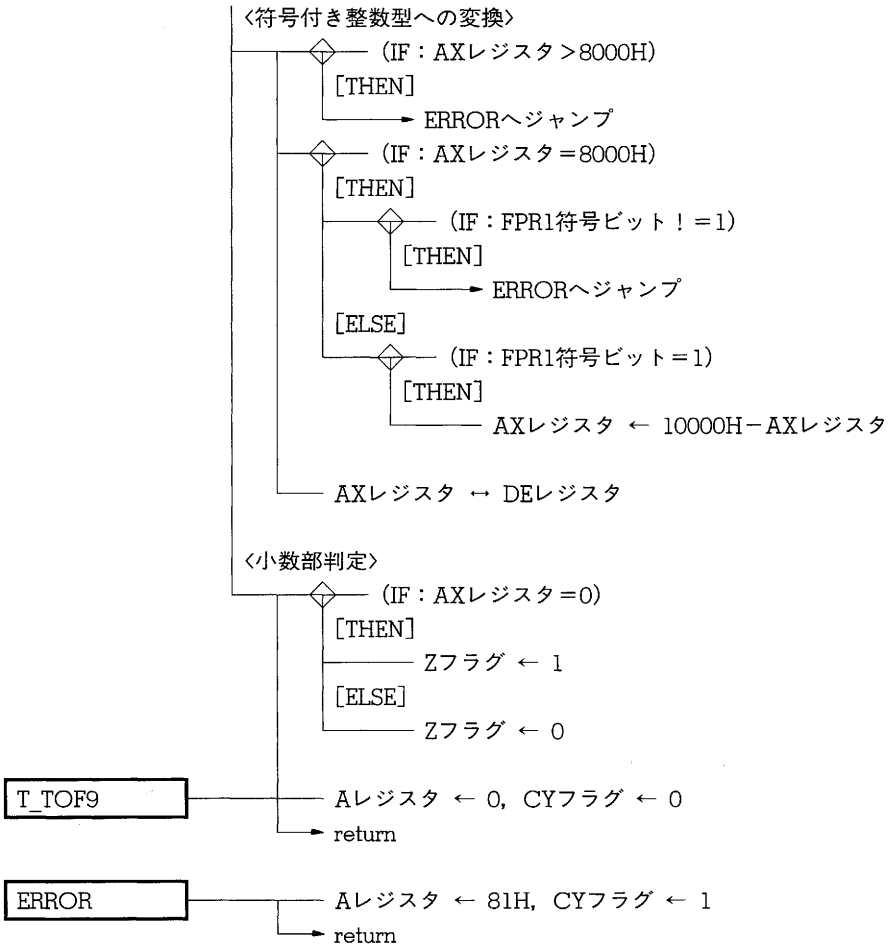
●指数部＞88Hのケースで，第３仮数部の全ビットが０かつ第１－第２仮数部の右シフト過程でキャリーがでなかったとき

○Zフラグ０を返す場合

上記以外

(7) 処 理 図


E LTOF
〈例外処理〉
(IF：FPR1_E≦80H)
[THEN]
(IF：FPR1_E=0)
[THEN]
Zフラグ ← 1
[ELSE]
Zフラグ ← 0
DEレジスタ ← 0
T_TOF9へジャンプ
(IF：FPR1_E>90H)
[THEN]
ERRORへジャンプ
〈符号なし整数型への変換〉
Aレジスタ ← FPR1_E−91H
BCレジスタ ← FPR1の第1−第2仮数部
Bレジスタのビット7 ← 1
DEレジスタ ← 0
(IF：FPR1の第3仮数部！=0)
[THEN]
Dレジスタ ← 80H
(IF：FPR1_E≦88H)
[THEN]
Eレジスタ ← Cレジスタ（FPR1第2仮数部）
Cレジスタ ← Bレジスタ（FPR1第1仮数部）
Bレジスタ ← 0
Aレジスタ ← Aレジスタ+8
BCDレジスタ全体を（0FFH−Aレジスタ）ビット右シフト
AXレジスタ ← BCレジスタ


〈符号付き整数型への変換〉
(IF：AXレジスタ＞8000H)
[THEN]
ERRORへジャンプ
(IF：AXレジスタ＝8000H)
[THEN]
(IF：FPR1符号ビット！＝1)
[THEN]
ERRORへジャンプ
[ELSE]
(IF：FPR1符号ビット＝1)
[THEN]
AXレジスタ ← 10000H－AXレジスタ
AXレジスタ ↔ DEレジスタ
〈小数部判定〉
(IF：AXレジスタ＝0)
[THEN]
Zフラグ ← 1
[ELSE]
Zフラグ ← 0
T_TOF9
Aレジスタ ← 0，CYフラグ ← 0
return
ERROR
Aレジスタ ← 81H，CYフラグ ← 1
return

6.5　IEEE-754形式→78K/II形式への浮動小数点形式変換関数（LTO78）

(1) 処理内容

FPR1の内容をIEEE-754形式から78K/II形式へ変換します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT，LTO78

(3) 消費スタック・サイズ

2（LTO78からの戻り番地2バイトのみ）

(4) 使用レジスタ

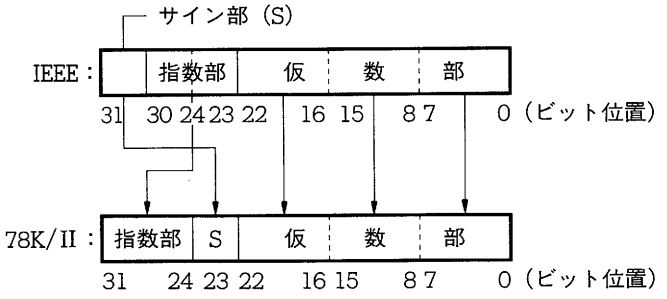
FPR1

(5) 処理時間（内部システム・クロック：6 MHz）

17.8 μs

(6) 処理手順

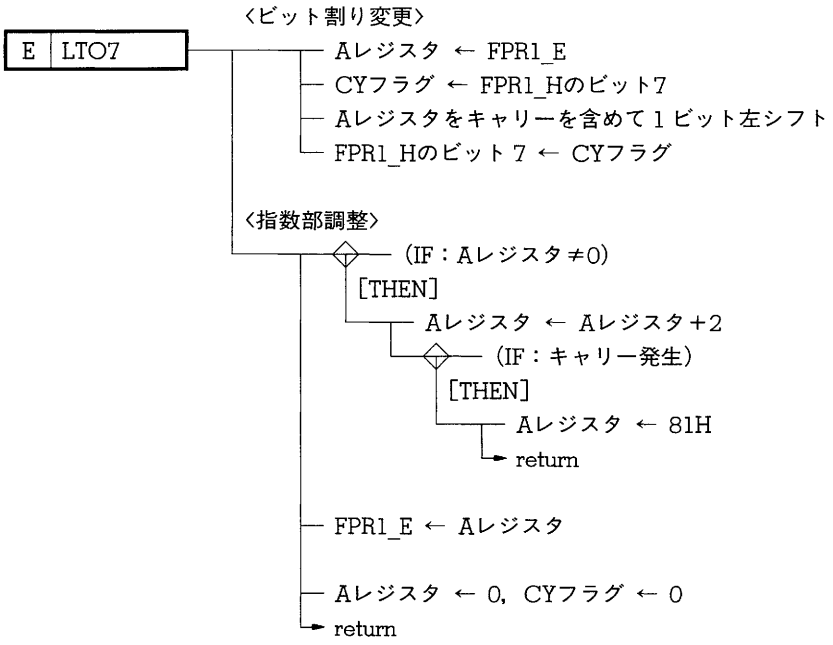
(a) IEEE形式のサイン部，指数部，仮数部のビット割りを78K/IIのビット割りに変換します。



(b) 指数部≧0FEHなら，エラーを返します。

(c) 指数部≠0なら，指数部に2を加算します。

(7) 処 理 図


E
LTO7
〈ビット割り変更〉
Aレジスタ ← FPR1_E
CYフラグ ← FPR1_Hのビット7
Aレジスタをキャリーを含めて1ビット左シフト
FPR1_Hのビット7 ← CYフラグ
〈指数部調整〉
(IF：Aレジスタ≠0)
[THEN]
Aレジスタ ← Aレジスタ+2
(IF：キャリー発生)
[THEN]
Aレジスタ ← 81H
return
FPR1_E ← Aレジスタ
Aレジスタ ← 0, CYフラグ ← 0
return

6.6　78K/II形式→IEEE-754形式への浮動小数点形式変換関数（LTOIE）

(1) 処理内容

FPR1の内容を78K/II形式からIEEE-754形式へ変換します。

(2) リンケージ対象オブジェクト・モジュール・ファイル

DFLT, LTOIE

(3) 消費スタック・サイズ

2（LTOIEからの戻り番地2バイトのみ）

(4) 使用レジスタ

FPR1

(5) 処理時間（内部システム・クロック：6 MHz）

17.8 μs

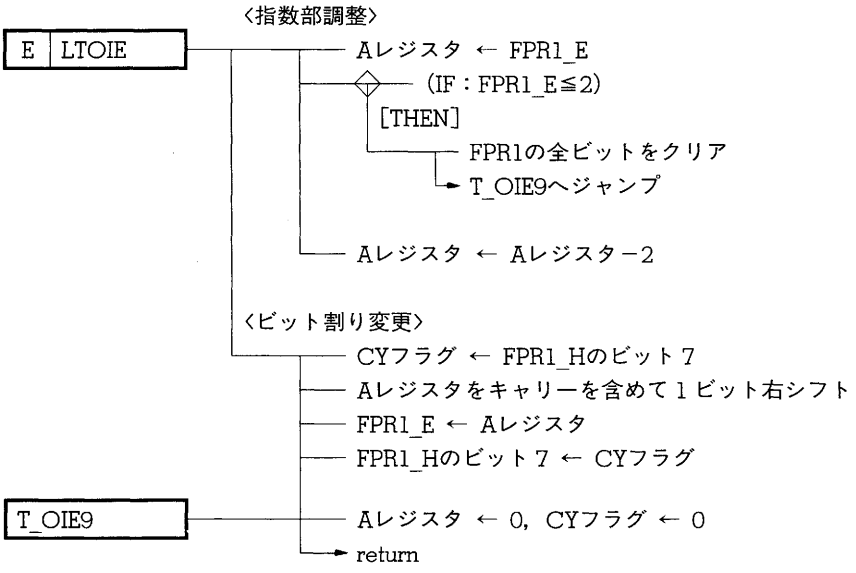
(6) 処理手順

(a) 指数部≦2なら，FPR1の全ビットを0にして終了します。

(b) 指数部から2を引きます。

(c) LTO78関数と逆の形でビット割りを変換します。

(7) 処 理 図



第7章　プログラム・リスト

(1) EQU. INC

```
SHORT   EQU     4       ;size of real type
INTEGR  EQU     2       ;size of integer type
BYTE    EQU     8       ;bit figures of byte
ZEROEX  EQU     080H    ;expornent bias for 0
R_OK    EQU     0       ;normal return code
R_ERR   EQU     81H     ;abnormal return code
```

(2) REF1. INC

```
EXTRN FPR1,FPR2

EXTRN FPR1_X,FPR1_M,FPR1_H,FPR1_E
EXTRN FPR2_X,FPR2_M,FPR2_H,FPR2_E

EXTRN FPR3,FPR3_1
EXTRN FPR4,FPR4_1,FPR4_2

EXTRN FPR5,FPR5_1,FPR5_2
```

(3) REF2. INC

```
EXTRN    LLD21,LLD21X
EXTRN    LLD31,LLD31X
EXTRN    LLD41,LLD41X
EXTRN    LLD51,LLD51X
EXTRN    LLD52
EXTRN    LLD13
EXTRN    LLD23,LLD23X
EXTRN    LLD14,LLD14X
EXTRN    LLD24,LLD24X
EXTRN    LLD15
EXTRN    LLD25,LLD25X
EXTRN    LLD1C,LLD1CX
EXTRN    LLD2C,LLD2CX

EXTRN    LXC13,LXC13X
EXTRN         LXC14X
EXTRN    LXC15,LXC15X
```

(4) ASCII. INC

```
A_PL    EQU     02BH        ;'+'
A_MN    EQU     02DH        ;'-'
A_PD    EQU     02EH        ;'.'
A_NL    EQU     000H        ;nul
A_SP    EQU     020H        ;space
A_E     EQU     045H        ;'E'
A_E2    EQU     065H        ;'e'
A_0     EQU     030H        ;'0'
A_9     EQU     039H        ;'9'

N_PL    EQU     16
N_MN    EQU     15
N_PD    EQU     14
N_NL    EQU     13
N_SP    EQU     12
N_E     EQU     11
N_9     EQU     9

S_INDX  EQU     17

@_INDX  MACRO
        DB A_PL, A_MN, A_PD, A_NL
        DB A_SP, A_E,  A_E2, A_9
        DB A_9-1,A_9-2,A_9-3,A_9-4
        DB A_9-5,A_9-6,A_9-7,A_9-8
        DB A_9-9
        ENDM
```

(5) DFLT. SRC

```
$       TITLE   ('FLOATING POINT REGISTERS')
        NAME    M_DFLT
;*****************************************************
;*
;*  FLOATING POINT REGISTERS
;*
;*              (ADDRESSING IN SADDR)
;*
;*****************************************************

        PUBLIC FPR1,FPR2

        PUBLIC FPR1_X,FPR1_M,FPR1_H,FPR1_E
        PUBLIC FPR2_X,FPR2_M,FPR2_H,FPR2_E

        PUBLIC FPR3,FPR3_1
        PUBLIC FPR4,FPR4_1,FPR4_2
        PUBLIC FPR5,FPR5_1,FPR5_2

        DSEG   SADDR
;******** FLOATING POINT REGISTER 1 **
FPR1_X:
        DS     1
FPR1:
        DS     1
FPR1_M:
        DS     1
FPR1_H:
        DS     1
FPR1_E:
        DS     1

;******** FLOATING POINT REGISTER 2 **
FPR2_X:
        DS     1
FPR2:
        DS     1
FPR2_M:
        DS     1
FPR2_H:
        DS     1
FPR2_E:
        DS     1
```

```
;******** FLOATING POINT REGISTER 3 **
FPR3:
FPR3_1:
        DS      1
        DS      1
        DS      1
        DS      1
        DS      1

;******** FLOATING POINT REGISTER 4 **
FPR4:
FPR4_1:
        DS      1
FPR4_2:
        DS      1
        DS      1
        DS      1
        DS      1

;******** FLOATING POINT REGISTER 5 **
FPR5:
FPR5_1:
        DS      1
FPR5_2:
        DS      1
        DS      1
        DS      1
        DS      1

        END
```

(6) LFLT1. SRC

```
$       TITLE   ('THE 4 RULES FUNCTIONS')
        NAME    M_LFLT1

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)

        PUBLIC  LADD,LSUB,LMLT,LDIV
        PUBLIC  LADDX,LMLTX

        CSEG

;*************************************************
;*
;*  FLOATING POINT ADDITION FUNCTION
;*
;*    DESTINATION REGISTER: FPR1
;*    SOURCE REGISTER     : FPR2
;*
;*    RESULT : FPR1 += FPR2
;*             ERROR then set CY
;*
;*************************************************
LADD:
        FPR1_X = #0
        FPR2_X = #0

;******** CHECK DIFFERENCE OF EXPORNENT **

LADDX:

        A = FPR1_E
        A -= FPR2_E
        if_bit (!CY)
          if (A >= #SHORT*BYTE || FPR2_E == #0)
            goto T_ADD9
          endif
        else
```

```
        if (A <= #LOW(-(SHORT*BYTE)) || FPR1_E == #0)
          FPR1_X = FPR2_X
          FPR1   = FPR2
          FPR1_M = FPR2_M
          FPR1_H = FPR2_H
          FPR1_E = FPR2_E
          goto T_ADD9
        endif
      endif


;******** LOAD MANTISSA TO GENERAL reg. **

      ;* LHED reg. <- FPR1 *
      ;* FPR1_X    <- A (DIFFERENCE of EXP) *
      A <-> FPR1_X
      A <-> L

      A = FPR1
      A <-> H

      A = FPR1_M
      A <-> E

      A = FPR1_H
      A |= #80H       ;set MSB
      A <-> D

      ;* CBXA reg. <- FPR2 *
      A = FPR2_X
      A <-> C
      A = FPR2
      A <-> B
      A = FPR2_M
      A <-> X
      A = FPR2_H
      A |= #80H         ;set MSB
```

```
;******** BE AGREED MANTISSA POTENTIAL **

        if (FPR1_X != #0)      ;exp. agree?
          if_bit (!FPR1_X.7)    ;destination is higher
            repeat
              SHRW AX,1
              RORC B,1
              RORC C,1
              FPR1_X--
            until_bit (Z)
          else                  ;sourse is higher
            repeat
              SHRW DE,1
              RORC H,1
              RORC L,1
              FPR1_X++
            until_bit (Z)
            FPR1_E = FPR2_E
          endif
        endif

;******** CALC. MANTISSA **

        FPR2_H ^= FPR1_H
        if_bit (!FPR2_H.7)      ;sign agree?
          ADD  L,C
          ADDC H,B
          ADDC E,X
          ADDC D,A          ;CY : mantissa overflow bit
        else
          if (AX == DE)
            if (B == H)
              if (C == L)
                goto ZERO
              endif
            endif
          endif
```

```
          if_bit (!CY)          ; source > destination
            NOT1 FPR1_H.7
            A <-> D
            X <-> E
            B <-> H
            C <-> L
          endif

          SUB  L,C
          SUBC H,B
          SUBC E,X
          SUBC D,A
        endif

;*********************
;*        NORMALIZE **
;*********************
T_NOR:
        A = D

        if_bit (CY)              ; mantissa overflow
          FPR1_E++
          if_bit (Z)
            goto ERROR
          endif
          RORC A,1
          RORC E,1
          RORC H,1
          RORC L,1
        else
          if (FPR1_E == #0)
            goto ZERO
          endif

T_NOR2:
          while_bit (!A.7)      ;catastrophic cancellation
            FPR1_E--
            if_bit (Z)
              goto T_ADD9
            endif
            SHLW HL,1
            ROLC E,1
            ROLC A,1
          endw
```

```
          if (FPR1_E == #0)
            goto ERROR              ; exp=100H
          endif
        endif

        ;* return mantissa *
        if_bit (!FPR1_H.7)
          A &= #7FH
        endif
        FPR1_H = A
        FPR1_M = E (A)
        FPR1   = H (A)
        FPR1_X = L (A)
T_ADD9:
        A = #R_OK
        CLR1 CY
        RET
ZERO:
        FPR1_E = #0
        goto T_ADD9
ERROR:
        A = #R_ERR
        SET1 CY
        RET

;*****************************************************
;*
;*  FLOATING POINT SUBTRACTION FUNCTION
;*
;*    DESTINATION REGISTER: FPR1
;*    SOURCE REGISTER     : FPR2
;*
;*    RESULT : FPR1 -= FPR2
;*             ERROR then set CY
;*
;*****************************************************
LSUB:
        NOT1 FPR2_H.7
        goto LADD
```

```
;*************************************************
;*
;*  FLOATING POINT MULTIPLICATION FUNCTION
;*
;*    DESTINATION REGISTER: FPR1
;*    SOURCE REGISTER     : FPR2
;*
;*    RESULT : FPR1 *= FPR2
;*             ERROR then set CY
;*
;*************************************************
LMLT:
        FPR1_X = #0
        FPR2_X = #0

;******** ZERO EXCEPTION **
LMLTX:
        if (FPR1_E == #0 || FPR2_E == #0)
          goto ZERO
        endif

;******** MULTIPLE EXPORNENT **

        A = FPR1_E
        A += FPR2_E

        if_bit (CY)
          if (A > #ZEROEX)      ;exp. > 100H
            goto ERROR
          endif
          A += #ZEROEX
        else
          A -= #ZEROEX
          if_bit (Z || CY)      ;exp. <= 0
            goto ZERO
          endif
        endif

        FPR1_E = A              ;set expornent

        CY = FPR1_H.7
        CY ^= FPR2_H.7
        FPR1_H.7 = CY           ;set sign
```

```
;******** CALC. MANTISSA (result set to AEHLreg.)**

        SET1 FPR2_H.7           ;set MSB

        ;* FPR1_X *
        A = FPR1_X      ;FPR1_X * FPR2_H
        A <-> B
        A = FPR2_H
        MULU B
        A <-> L


        ;* FPR1   *
        A = FPR1        ;FPR1   * FPR2_M
        A <-> B
        A = FPR2_M
        MULU B
        L += A           ;->CY

        A = FPR2_H      ;FPR1   * FPR2_H
        MULU B
        ADDC A,#0        ;<-CY
        L += X           ;->CY
        ADDC A,#0        ;<-CY
        A <-> H

        ;* FPR1_M *
        A = FPR1_M      ;FPR1_M * FPR2
        A <-> B
        A = FPR2
        MULU B
        L += A           ;->CY

        A = FPR2_M      ;FPR1_M * FPR2_M
        MULU B
        ADDC A,#0        ;<-CY
        L += X           ;->CY
        ADDC H,A         ;<-CY,->CY
```

```
        A = FPR2_H      ;FPR1_M * FPR2_H
        MULU B
        ADDC A,#0        ;<-CY
        H += X           ;->CY
        ADDC A,#0        ;<-CY
        A <-> E

        ;* FPR1_H *
        A = FPR1_H      ;FPR1_H * FPR2_X
        A |= #80H       ;set MSB
        A <-> B
        A = FPR2_X
        MULU B
        L += A           ;->CY

        A = FPR2        ;FPR1_H * FPR2
        MULU B
        ADDC A,#0        ;<-CY
        L += X           ;->CY
        ADDC H,A         ;<-CY,->CY

        A = FPR2_M      ;FPR1_H * FPR2_M
        MULU B
        ADDC A,#0        ;<-CY
        H += X           ;->CY
        ADDC E,A         ;<-CY,->CY


        A = FPR2_H      ;FPR1_H * FPR2_H
        MULU B
        ADDC A,#0        ;<-CY
        E += X           ;->CY
        ADDC A,#0        ;<-CY

;******** NORMALIZE **

        goto T_NOR2
```

```
;*************************************************
;*
;*  FLOATING POINT DIVISION FUNCTION
;*
;*    DESTINATION REGISTER: FPR1
;*    SOURCE REGISTER     : FPR2
;*
;*    RESULT : FPR1 /=FPR2
;*             ERROR then set CY
;*
;*************************************************
LDIV:

;******** ZERO EXCEPTION **

        if (FPR2_E == #0)
          goto ERROR
        endif
        if (FPR1_E == #0)
          goto ZERO
        endif

;******** DIVIDE EXPORNENT **

        A = FPR1_E
        A -= FPR2_E

        if_bit (CY)
          if_bit (!A.7)          ;exp. < 0
            goto ZERO
          endif
        else
          if_bit (A.7)           ;exp. >= 100H
            goto ERROR
          endif
        endif
        A += #ZEROEX

        FPR1_E = A               ;set expornent
```

```
        CY = FPR1_H.7               ;make sign
        CY ^= FPR2_H.7
        FPR1_H.7 = CY               ;set sign

;******** LOAD MANTISSA **

        ;* XBCreg. <- FPR1(1st-3rd mantissa) *
        A = #(SHORT-1)*BYTE+1       ;loop counter
        A <-> FPR1
        A <-> C

        A = FPR1_M
        A <-> B

        A = FPR1_H
        A |= #80H           ;set MSB
        A <-> X

        CLR1 CY

        ;* Lreg. <- FPR2(1st mantissa) *
        A = FPR2_H
        A |= #80H           ;set MSB
        A <-> L

;******** DIVIDE MANTISSA **

        H = #0

        goto T_DIV1

        ;* DEHreg. <- quotient *
        repeat
          SHLW BC,1     ;b*2
          ROLC X,1
```

```
T_DIV1:
          NOT1 CY
          if_bit (CY)
            if (X == L)
              A = B
              if (A == FPR2_M)
                A = C
                CMP A,FPR2
              endif
            endif
          endif


          if_bit (!CY)      ;if d > S
            A = FPR2        ;  then d -= S
            SUB C,A
            A = FPR2_M
            SUBC B,A
            SUBC X,L
            SET1 CY
          else
            CLR1 CY
          endif

          ROLC H,1          ;set quotient
          ROLC E,1
          ROLC D,1          ;CY then mantissa overflow

          FPR1--
        until_bit (Z)

;******** NORMALIZE **

        L = #0          ;clr 4th mantissa

        goto T_NOR

        END
```

(7) LFLT2. SRC

```
$       TITLE     ('FLOATING POINT COMMON FUNCTIONS 1')
        NAME      M_LFLT2

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN     LADDX,LMLTX

        PUBLIC    LPLY,LPLY2

        CSEG

;*****************************************************************
;*
;*   FLOATING POINT FUNCTION THAT
;*
;*     CALC. A POLYNOMIAL EXPRESSION by EXTENDED
;*
;*                                  2                   n
;*  polynomial.1: x + k1xy + k1k2xy  + ... + k1k2..knxy
;*
;*     input  conditions:
;*        FPR1 <- X , FPR4 <- y
;*        DE <- head address of coefficient array (k1,,kn)
;*        C <- n
;*
;*                                  2                   n
;*  polynomial.2: z + k1xy + k1k2xy  + ... + k1k2..knxy
;*
;*     input  conditions:
;*        FPR1 <- x , FPR4 <- y, FPR3 <- z
;*        DE <- head address of coefficient array (k1,,kn)
;*        C <- n
;*
;*     output conditions(common to both):
;*        FPR1 <- result of polynomial expression
;*        FPR4 : keep
;*
;*****************************************************************
```

```
LPLY:
        CALL !LLD31X
        goto LPLY2

        repeat

          CALL !LXC13X

LPLY2:
          CALL !LLD24X

          PUSH BC
          PUSH DE

          CALL !LMLTX

          POP DE
          CALL !LLD2CX
          PUSH DE

          CALL !LMLTX

          CALL !LLD21X
          CALL !LXC13X

          CALL !LADDX

          POP DE
          POP BC

          if (FPR3+SHORT == #0)
            RET
          endif

          A = FPR1_E
          A -= FPR3+SHORT
          if_bit (!CY)
            if (A >= #(SHORT-1)*BYTE)
              RET
            endif
          endif
```

```
  C--
until_bit (Z)
RET

END
```

(8) LLD. SRC

```
$       TITLE   ('FPR LOAD FUNCTIONS')
        NAME    M_LLD

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)

;*****************************************************
;*
;*  FLOATING POINT REGISTER LOAD FUNCTIONS
;*
;*****************************************************

        PUBLIC  LLD21,LLD21X
        PUBLIC  LLD31,LLD31X
        PUBLIC  LLD41,LLD41X
        PUBLIC  LLD51,LLD51X
        PUBLIC  LLD52
        PUBLIC  LLD13
        PUBLIC  LLD23,LLD23X
        PUBLIC  LLD14,LLD14X
        PUBLIC  LLD24,LLD24X
        PUBLIC  LLD15
        PUBLIC  LLD25,LLD25X
        PUBLIC  LLD1C,LLD1CX
        PUBLIC  LLD2C,LLD2CX

        PUBLIC  LXC13,LXC13X
        PUBLIC        LXC14X
        PUBLIC  LXC15,LXC15X

        CSEG

;******* LOAD FPR2,FPR1

LLD21X:
        FPR2_X = FPR1_X
LLD21:
        FPR2   = FPR1
        FPR2_M = FPR1_M
        FPR2_H = FPR1_H
        FPR2_E = FPR1_E
        RET
```

```
;******* LOAD FPR3,FPR1

LLD31X:
        FPR3   = FPR1_X
LLD31:
        FPR3+1 = FPR1
        FPR3+2 = FPR1_M
        FPR3+3 = FPR1_H
        FPR3+4 = FPR1_E
        RET

;******* LOAD FPR4,FPR1

LLD41X:
        FPR4   = FPR1_X
LLD41:
        FPR4+1 = FPR1
        FPR4+2 = FPR1_M
        FPR4+3 = FPR1_H
        FPR4+4 = FPR1_E
        RET

;******* LOAD FPR5,FPR1

LLD51X:
        FPR5   = FPR1_X
LLD51:
        FPR5+1 = FPR1
        FPR5+2 = FPR1_M
        FPR5+3 = FPR1_H
        FPR5+4 = FPR1_E
        RET

;******* LOAD FPR5,FPR2

LLD52:
        FPR5+1 = FPR2
        FPR5+2 = FPR2_M
        FPR5+3 = FPR2_H
        FPR5+4 = FPR2_E
        RET
```

```
;******* LOAD FPR1,FPR3

LLD13:
        FPR1   = FPR3+1
        FPR1_M = FPR3+2
        FPR1_H = FPR3+3
        FPR1_E = FPR3+4
        RET

;******* LOAD FPR2,FPR3

LLD23X:
        FPR2_X = FPR3
LLD23:
        FPR2   = FPR3+1
        FPR2_M = FPR3+2
        FPR2_H = FPR3+3
        FPR2_E = FPR3+4
        RET

;******* LOAD FPR1,FPR4

LLD14X:
        FPR1_X = FPR4
LLD14:
        FPR1   = FPR4+1
        FPR1_M = FPR4+2
        FPR1_H = FPR4+3
        FPR1_E = FPR4+4
        RET

;******* LOAD FPR2,FPR4

LLD24X:
        FPR2_X = FPR4
LLD24:
        FPR2   = FPR4+1
        FPR2_M = FPR4+2
        FPR2_H = FPR4+3
        FPR2_E = FPR4+4
        RET
```

```
;******* LOAD FPR1,FPR5

LLD15:
        FPR1   = FPR5+1
        FPR1_M = FPR5+2
        FPR1_H = FPR5+3
        FPR1_E = FPR5+4
        RET

;******* LOAD FPR2,FPR5

LLD25X:
        FPR2_X = FPR5
LLD25:
        FPR2   = FPR5+1
        FPR2_M = FPR5+2
        FPR2_H = FPR5+3
        FPR2_E = FPR5+4
        RET
        RET

;******* LOAD FPR1,constant

LLD1CX:
        FPR1_X = [DE+] (A)
LLD1C:
        FPR1   = [DE+] (A)
        FPR1_M = [DE+] (A)
        FPR1_H = [DE+] (A)
        FPR1_E = [DE+] (A)
        RET

;******* LOAD FPR2,constant

LLD2CX:
        FPR2_X = [DE+] (A)
LLD2C:
        FPR2   = [DE+] (A)
        FPR2_M = [DE+] (A)
        FPR2_H = [DE+] (A)
        FPR2_E = [DE+] (A)
        RET
```

```
;******* XCHANGE FPR1,FPR3

LXC13X:
        FPR1_X <-> FPR3
LXC13:
        FPR1   <-> FPR3+1
        FPR1_M <-> FPR3+2
        FPR1_H <-> FPR3+3
        FPR1_E <-> FPR3+4
        RET

;******* XCHANGE FPR1,FPR4

LXC14X:
        FPR1_X <-> FPR4
        FPR1   <-> FPR4+1
        FPR1_M <-> FPR4+2
        FPR1_H <-> FPR4+3
        FPR1_E <-> FPR4+4
        RET

;******* XCHANGE FPR1,FPR5

LXC15X:
        FPR1_X <-> FPR5
LXC15:
        FPR1   <-> FPR5+1
        FPR1_M <-> FPR5+2
        FPR1_H <-> FPR5+3
        FPR1_E <-> FPR5+4
        RET

        END
```

(9) LSIN. SRC

```
$       TITLE       ('SINE FUNCTION')
        NAME        M_LSIN

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LPLY
        EXTRN       LADDX,LMLTX

        EXTRN       FTOL,LTOF

        PUBLIC      LSIN
        PUBLIC      LMOD90,LSIN90

S_PLY   EQU         5

C_1X    EQU         0A2H
C_1L    EQU         0DAH
C_1M    EQU         00FH
C_1H    EQU         049H
C_1E    EQU         081H

        CSEG

;*********************************************************
;*
;*   FLOATING POINT SINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- sin(x)
;*
;*********************************************************
LSIN:

;********* SCALING of X **

          CALL !LMOD90
```

```
;********* GET SIN(x' + n * pai/2)  , 0 <= x' < pai/2 **

LSIN90:
          if_bit (FPR4_1.0)   ;mod(n/2)=1?
            SET1 FPR1_H.7
            DE = #C_1
            CALL !LLD2CX
            CALL !LADDX       ; x' <- pai/2 - x'
          endif

          CY = FPR4_1.1       ; mod(n/4)=2 or 3?
          CY ^= FPR4_2.7      ;  then turn sign bit

          FPR1_H.7 = CY       ;set SIGN

;********* CALC. POLYNOMIAL EXPRESSION **

          CALL !LLD41X        ;set x'' to FPR4

          CALL !LLD21X
          CALL !LMLTX         ;x''*x''

          CALL !LXC14X        ;set x''*x'' to FPR4
                              ;set x'' to FPR1
          DE = #C_K
          C = #S_PLY
          CALL !LPLY

          A = #R_OK
          CLR1 CY
          RET
```

```
;*************************************
;**        GET MOD by pai/2 (->FPR1) **
;*************************************

LMOD90:
          FPR1_X = #0
          FPR4_1 = #0              ;quotient keeper
          FPR4_2 = FPR1_H          ;sign keep
          CLR1 FPR1_H.7

          while (forever)
            ;* CMP FPR1,pai/2
               if (FPR1_E == #C_1E)
                 if (FPR1_H == #C_1H)
                   if (FPR1_M == #C_1M)
                     if (FPR1 == #C_1L)
                       CMP FPR1_X,#C_1X
                     endif
                   endif
                 endif
               endif

            if_bit (CY)
              break
            endif

            CALL !LLD31X         ;esc. x' to FPR3

            ;* *2/pai *
            DE = #C_2
            CALL !LLD2CX
            CALL !LMLTX          ;x' / pai/2
            FPR1_X  = #0
            FPR1    = #0
            FPR1_M &= #0FCH      ;valid digit → 14

            ;* FPR1 -> integer *
            CALL !LTOF
            if_bit (!CY)
              if_bit (!Z)
                CALL !FTOL
              endif
```

```
        A = E
        A += FPR4_1     ;add last 2bit of quotient
        FPR4_1 = A
      endif

      DE = #C_1
      CALL !LLD2CX
      CALL !LMLTX          ; int(x' / pai/2) * pai/2

      SET1 FPR1_H.7
      CALL !LLD23X
      CALL !LADDX          ; x' - int(x' / pai/2) * pai/2
    endw

    RET

C_1:
    DB C_1X,C_1L,C_1M,C_1H,C_1E ; const pai/2
C_2:
    DB 06EH,083H,0F9H,022H,080H ;       2/pai

C_K:                             ;coefficient array for LPLY
    DB 0AAH,0AAH,0AAH,0AAH,07EH ;const -1/6
    DB 0CCH,0CCH,0CCH,0CCH,07CH ;      -1/20
    DB 0C3H,030H,00CH,0C3H,07BH ;      -1/42
    DB 0E3H,038H,08EH,0E3H,07AH ;      -1/72
    DB 04FH,009H,0F2H,094H,07AH ;      -1/110

    END
```

(10) LCOS. SRC

```
$       TITLE       ('COSINE FUNCTION')
        NAME        M_LCOS

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)

        EXTRN       LMOD90,LSIN90

        PUBLIC      LCOS,LCOS90

        CSEG
;*********************************************************
;*
;*   FLOATING POINT COSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- cos(x)
;*
;*********************************************************
LCOS:

;********* SCALING of X **

          CALL !LMOD90

;********* GET COS(X' + n * pai/2)  , -pai/2 < x' < pai/2 **

LCOS90:
          CLR1 FPR4_2.7    ; x' <- |x'|
          FPR4_1++         ; n <- n+1
          goto LSIN90

          END
```

(11) LTAN. SRC

```
$       TITLE       ('TANGENT FUNCTION')
        NAME        M_LTAN

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LDIV
        EXTRN       LMOD90,LSIN90,LCOS90

        PUBLIC      LTAN

        CSEG

;*********************************************************
;*
;*   FLOATING POINT TANGENT FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- tan(x)
;*                        ERROR then set CY
;*
;*********************************************************
LTAN:

          CALL !LMOD90          ;get mod by pai/2

          A = FPR4_2
          A <-> X               ;X: esc. sign bit
          A = FPR4_1            ;A: esc. quotient by pai/2
          PUSH AX

          CALL !LLD51X          ;esc. mod by pai/2 to FPR5

          CALL !LCOS90          ;cos(x)
          POP AX

          FPR4_1 = A
          FPR4_2 = X (A)

          CALL !LXC15X          ;esc. cos(x) to FPR5
          CALL !LSIN90          ;sin(x)

          CALL !LLD25           ;ret. cos(x) to FPR2
          goto LDIV             ;sin(x)/cos(x)

          END
```

(12) LLOG. SRC

```
$       TITLE       ('LOGARITHMIC FUNCTION')
        NAME        M_LLOG

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LADD,LADDX,LMLT,LMLTX,LDIV
        EXTRN       LPLY2
        EXTRN       FTOL

        PUBLIC      LLOG

S_PLY   EQU         6

C_2L    EQU         000H
C_2M    EQU         000H
C_2H    EQU         000H
C_2E    EQU         081H

        CSEG

;*********************************************************
;*
;*   FLOATING POINT LOGARITHMIC FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- log(x)
;*                        ERROR then set CY
;*
;*********************************************************
LLOG:
          FPR1_X = #0
```

```
;********* EXCEPTION **

          if (FPR1_E == #0)
            goto ERROR
          endif
          if_bit (FPR1_H.7)
            goto ERROR
          endif

          ;* FPR1 = 1? *
          if (FPR1_E == #C_2E)
            if (FPR1_H == #C_2H)
              if (FPR1_M == #C_2M)
                if (FPR1 == #C_2L)
                  FPR1_E = #0
                  goto T_LOG9
                endif
              endif
            endif
          endif

;********* CALC. EXP.PART LOG **

          A = #ZEROEX
          A <-> FPR1_E
          CALL !LLD31           ;esc. mantissa(Xf) to FPR3

          A -= #ZEROEX
          if_bit (!CY)
            D = #0
          else
            D = #0FFH
          endif
          A <->E                ;DE: integer value of exp.part
          CALL !FTOL            ;real value of exp.part

          DE = #C_1
          CALL !LLD2CX
          CALL !LMLTX           ;Xe * log(2)
          CALL !LLD41X          ;esc. exp.part LOG to FPR4
```

```
;********* TRANS. MANTISSA FOR TAYLOR APPROXIMATE **

          CALL !LLD13
          DE = #C_2
          CALL !LLD2C
          CALL !LADD                ; Xf+1

          CALL !LXC13
          DE = #C_2
          CALL !LLD2C
          SET1 FPR2_H.7
          CALL !LADD                ;Xf-1

          CALL !LLD23
          CALL !LDIV                ;(Xf-1)/(Xf+1)  :x'

;********* CALC. MANTISSA LOG **

          CALL !LLD31X          ;esc X' to FOR3
          FPR3+4++              ;set 2X' to FPR3

          CALL !LLD21X
          CALL !LMLT            ; X'*X'
          CALL !LXC14X          ;set X'*X' to FPR4


          CALL !LLD23X
          CALL !LADDX           ;set Xe*log(2)+2X' to FPR1
          CALL !LXC13X

          DE = #C_K
          C = #S_PLY
          CALL !LPLY2

T_LOG9:
          A = #R_OK
          CLR1 CY
          RET
ERROR:
          A = #R_ERR
          SET1 CY
          RET
```

```
C_1:
        DB 0F7H,017H,072H,031H,080H ; const log(2)
C_2:
        DB      C_2L,C_2M,C_2H,C_2E ;       1

C_K:                                ; coefficient array of LPLY
        DB 0AAH,0AAH,0AAH,02AH,07FH ; const 1/3
        DB 099H,099H,099H,019H,080H ;       3/5
        DB 0B6H,06DH,0DBH,036H,080H ;       5/7
        DB 0C7H,071H,01CH,047H,080H ;       7/9
        DB 017H,05DH,074H,051H,080H ;       9/11
        DB 0D8H,089H,09DH,058H,080H ;      11/13

        END
```

(13) **LLOG10. SRC**

```
$       TITLE       ('LOGARITHMIC FUNCTION 2')
        NAME        M_LLOG10

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LMLTX
        EXTRN       LLOG

        PUBLIC      LLOG10

        CSEG

;*********************************************************
;*
;*   FLOATING POINT LOGARITHMIC FUNCTION 2
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- log10(x)
;*                        ERROR then set CY
;*
;*********************************************************
LLOG10:

;********* log(x) / log(10) **

          CALL !LLOG
          if_bit (CY)
            RET
          endif

          DE = #C_1
          CALL !LLD2CX
          goto LMLTX

C_1:
          DB 0A9H,0D8H,05BH,05EH,07FH ; const 1/log(10)

          END
```

(14) LEXP. SRC

```
$       TITLE       ('EXPONENTIAL FUNCTION')
        NAME        M_LEXP

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LADDX,LMLTX
        EXTRN       LPLY2
        EXTRN       LTOF,FTOL

        PUBLIC      LEXP,LEXPX

S_PLY   EQU         8

        CSEG

;*********************************************************
;*
;*   FLOATING POINT EXPONENTIAL FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- exp(x)
;*                        ERROR then set CY
;*
;*********************************************************
LEXP:
          FPR1_X = #0

;********* TRANSLATE to EXP2(X) **
LEXPX:
          FPR3_1 = FPR1_H       ;esc sign bit

          DE = #C_1
          CALL !LLD2CX
          CALL !LMLTX           ;1/log2*X
          if_bit (!CY)
            CALL !LTOF
          endif
          if_bit (CY)
            goto T_FLOW
          endif
```

```
;********* CALC. EXP **

          if_bit (!Z && FPR3_1.7)
            DE--                ;DE <- floor integer(X/log2)
          endif

          AX = DE
          AX += #ZEROEX+1       ;AX <- upper integer(X/log2)

          if (A != #0)
            goto T_FLOW
          endif

          FPR5_1 = X (A)        ;esc exp

;********* CALC. MANTISSA **

          CALL !LLD21X
          CALL !FTOL            ; floor integer(X/log2)
          NOT1 FPR1_H.7
          CALL !LADDX           ; X/log2 -(floor integer(X/log2))

          CALL !LLD41X          ;set X'

          DE = #C_2
          CALL !LLD2CX
          CALL !LMLTX           ;log2·X'

          CALL !LLD31X          ;set log2·X'

          DE = #C_3
          CALL !LLD2CX
          CALL !LADDX           ;1+log2·X'

          CALL !LXC13X

          DE = #C_K
          C = #S_PLY
          CALL !LPLY2
```

```
          if (FPR1_E == #80H)   ;if cancellation happen then
            if (FPR5_1 != #0)   ;
              FPR5_1--          ;  adjust expornent
            endif
          endif

;********* RETURN EXP.PART **

          FPR1_E = FPR5_1
T_EXP9:
          A = #R_OK
          CLR1 CY
          RET
T_FLOW:
          if_bit (FPR3_1.7)
            FPR1_E = #0
            goto T_EXP9
          endif

          A = #R_ERR
          SET1 CY
          RET

C_1:
          DB 029H,03BH,0AAH,038H,081H  ; const  1/log2
C_2:
          DB 0F7H,017H,072H,031H,080H  ;        log2
C_3:
          DB 000H,000H,000H,000H,081H  ;        1

C_K:                                   ; coefficient array of LPLY2
          DB 0F7H,017H,072H,031H,07FH  ; const log2/2
          DB 0F5H,01FH,098H,06CH,07EH  ;       log2/3
          DB 0F7H,017H,072H,031H,07EH  ;       log2/4
          DB 0F9H,0DFH,0F4H,00DH,07EH  ;       log2/5
          DB 0F5H,01FH,098H,06CH,07DH  ;       log2/6
          DB 01BH,089H,0CBH,04AH,07DH  ;       log2/7
          DB 0F7H,017H,072H,031H,07DH  ;       log2/8
          DB 0F8H,0BFH,0BAH,01DH,07DH  ;       log2/9

          END
```

(15) LEXP10. SRC

```
$       TITLE       ('EXPONENTIAL FUNCTION 2')
        NAME        M_LEXP10

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LMLTX
        EXTRN       LEXPX

        PUBLIC      LEXP10

        CSEG

;*********************************************************
;*
;*   FLOATING POINT EXPONENTIAL FUNCTION 2
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- exp10(x)
;*                        ERROR then set CY
;*
;*********************************************************
LEXP10:
          FPR1_X = #0

;********* TRANSLATE EXP10 TO EXP **

          FPR3_1 = FPR1_H

          DE = #C_1
          CALL !LLD2CX
          CALL !LMLTX
          if_bit (CY)
            if_bit (FPR3_1.7)
              FPR1_E = #0
              A = #R_OK
              CLR1 CY
            endif
            RET
          endif
```

```
;********* CALC. exp(X*log(10)) **

          goto LEXPX

C_1:
          DB 0DDH,08DH,05DH,013H,082H ;const log(10)

          END
```

(16) **LPOW. SRC**

```
$       TITLE       ('POWER FUNCTION')
        NAME        M_LPOW

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LMLTX
        EXTRN       LLOG,LEXPX

        PUBLIC      LPOW

        CSEG

;*********************************************************
;*
;*   FLOATING POINT POWER FUNCTION
;*
;*     input  condition : FPR1 <- a , FPR2 <- b
;*
;*     output conditions: FPR1 <- a^b
;*                        ERROR then set CY
;*
;*********************************************************
LPOW:

;********* a=0 EXCEPTION **

          if (FPR1_E == #0)
            CMP FPR2_E,#0
            if_bit (Z || FPR2_H.7)
              goto ERROR            ;0^0,0^(negative)=overflow
            else
              FPR1_E = #0
              goto T_POW9           ;0^(positive)=0
            endif
          endif
```

```
;********* a<0 EXCEPTION **

          X = #0                ;X.0 :sign of result
          if_bit (FPR1_H.7)
            A = FPR2_E
            if (A == #0)
              DE = #C_1
              CALL !LLD1C
              goto T_POW9           ;x^0=1
            endif

;**     ** b: DECIMAL PART = 0 ? **

            if (A <= #ZEROEX)
              goto ERROR            ;(negative)^(decimal)=error
            endif

            A <-> X

            A = FPR2_H
            A |= #80H
            A <-> B
            A = FPR2_M
            A <-> C
            A = FPR2
            A <-> X

            A -= #ZEROEX+BYTE*(SHORT-1)
            if_bit (CY)
              repeat
                SHRW BC,1
                RORC X,1
                if_bit (CY)
                  goto ERROR    ;include decimal digit
                endif
                A++
              until_bit (Z)
            endif
```

```
;**     ** b: ODD or EVEN ? **

          if_bit (!Z)          ;not include UNIT1
            X = #0
          endif
        endif

        PUSH AX                ;esc UNIT1(X.0)

;********* CALC. exp(b * log(|a|) **

        CALL !LLD52            ;esc b to FPR5
        CLR1 FPR1_H.7
        CALL !LLOG             ;log(|a|)
        CALL !LLD25            ;ret. b to FPR2
        FPR2_X = #0
        CALL !LMLTX            ;b * log(|a|)

        if_bit (CY)            ;overflow
          POP BC
          if_bit (FPR5+(SHORT-1).7)
            FPR1_E = #0
            goto T_POW9
          endif
          RET
        endif

        CALL !LEXPX

;********* RETURN SIGN BIT **

        POP BC
        X = C

        if_bit (X.0)           ;if b = odd integer & a<0
          SET1 FPR1_H.7        ; then set sign bit
        endif

        RET
```

```
T_POW9:
        A = #R_OK
        CLR1 CY
        RET
ERROR:
        A = #R_ERR
        SET1 CY
        RET

C_1:
        DB 000H,000H,000H,081H ;const 1

        END
```

(17) **LSQRT. SRC**

```
$       TITLE       ('SQUARE ROOT FUNCTION')
        NAME        M_LSQRT

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LADD,LDIV

        PUBLIC      LSQRT

C_LIM   EQU         5       ;limiter of approximate

        CSEG

;*********************************************************
;*
;*  FLOATING POINT SQUARE ROOT FUNCTION
;*
;*   input  condition : FPR1 <- x
;*
;*   output conditions: FPR1 <- root(x)
;*                      ERROR then set CY
;*
;*********************************************************
LSQRT:

;********* EXCEPTION **

        if (FPR1_E == #0)
          goto T_QRT9
        endif
        if_bit (FPR1_H.7)
          goto ERROR
        endif
```

```
;********* GET EXP.PART ROOT **

          A = FPR1_E
          A++
          CY = PSW.6  ; CY <- Z
          RORC A,1
          A += #ZEROEX/2

          FPR4_1 = A              ;escape exp.part root

;*********  a -> r/2 (.125 - .5) **

          if_bit (FPR1_E.0)
            FPR1_E = #ZEROEX-2
          else
            FPR1_E = #ZEROEX-1
          endif

;********* CALC. VIRT.PART ROOT **

          CALL !LLD41         ;esc. r/2 to FPR4

          DE = #C_1
          CALL !LLD2C
          CALL !LADD        ; r/2 + .5  : 2ndary approximate

          FPR3_1 = #C_LIM-1
          repeat
            CALL !LLD31     ;esc.previous approximate(Rp) to FPR3
            CALL !LLD14
            CALL !LLD23
            CALL !LDIV      ; r/2 / Rp

            CALL !LLD23
            FPR2_E--        ; Rp/2
            CALL !LADD      ; Rp/2 + R/(2Rp) : next approximate

            if (FPR1_E == FPR3+4)
              if (FPR1_H == FPR3+3)
                if (FPR1_M == FPR3+2)
                  CMP FPR1,FPR3+1
                endif
              endif
            endif
```

```
            if_bit (!CY)
              break
            endif

            FPR3_1--
          until_bit (Z)

          FPR1_E = FPR4_1        ;ret. exp part ROOT
T_QRT9:
          A = #R_OK
          CLR1 CY
          RET
ERROR:
          A = #R_ERR
          SET1 CY
          RET

C_1:
          DB 000H,000H,000H,080H ;const .5

          END
```

(18) **LASIN. SRC**

```
$       TITLE       ('ARCSINE FUNCTION')
        NAME        M_LASIN

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LMLT,LDIV
        EXTRN       LADDX
        EXTRN       LSQRT,LATAN

        PUBLIC      LASIN

C_1X    EQU         000H
C_1L    EQU         000H
C_1M    EQU         000H
C_1H    EQU         000H
C_1E    EQU         081H

        CSEG

;************************************************
;*
;*   FLOATING POINT ARCSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- arcsin(x)
;*                        ERROR then set CY
;*
;************************************************
LASIN:

;******** EXCEPTION **

         if (FPR1_E == #0)
           goto T_SIN9
         endif

         CALL !LLD51          ; esc X to FPR5

         CLR1 FPR1_H.7        ; x <- |x|
```

```
        ;* |x| = 1? *
        if (FPR1_E == #C_1E)
          if (FPR1_H == #C_1H)
            if (FPR1_M == #C_1M)
              if (FPR1 == #C_1L)
                DE = #C_2
                CALL !LLD1CX
                if_bit (FPR5+1+2.7) ; X < 0 ?
                  SET1 FPR1_H.7
                endif
                goto T_SIN9
              endif
            endif
          endif
        endif

        if_bit (!CY)          ; |x| > 1
          A = #R_ERR          ; then error
          SET1 CY
          RET
        endif

;******** TRANS. to ARCTAN  **

        CALL !LLD21
        CALL !LMLT          ;X*X
        SET1 FPR1_H.7       ;-X*X

        DE = #C_1
        CALL !LLD2CX
        CALL !LADDX         ;1-X*X
        CALL !LSQRT         ;root(1-X*X)

        CALL !LLD21
        CALL !LLD15
        CALL !LDIV          ;X/root(1-X*X)

        goto LATAN

T_SIN9:
        A = #R_OK
        CLR1 CY
        RET
```

```
C_1:
        DB C_1X,C_1L,C_1M,C_1H,C_1E ;const 1
C_2:
        DB 0A2H,0DAH,00FH,049H,081H ;       pai/2

        END
```

(19) **LACOS. SRC**

```
$       TITLE       ('ARCCOSINE FUNCTION')
        NAME        M_LACOS

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LADDX
        EXTRN       LASIN

        PUBLIC      LACOS

        CSEG

;*******************************************************
;*
;*   FLOATING POINT ARCCOSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- arccos(x)
;*                        ERROR then set CY
;*
;*******************************************************
LACOS:

         CALL !LASIN
         if_bit (CY)
           RET
         endif

         NOT1 FPR1_H.7

         DE = #C_1
         CALL !LLD2CX
         goto LADDX      ;pai/2 -sin(x)

C_1:
         DB 0A2H,0DAH,00FH,049H,081H ;const pai/2

         END
```

(20) **LATAN. SRC**

```
$       TITLE       ('ARCTANGENT FUNCTION')
        NAME        M_LATAN

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LADD,LDIV
        EXTRN       LADDX,LMLTX
        EXTRN       LPLY2

        PUBLIC      LATAN

S_PLY   EQU         7

        CSEG

;*********************************************************
;*
;*   FLOATING POINT ARCTANGENT FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- arctan(x)
;*
;*********************************************************
LATAN:
         FPR1_X = #0

;******** ZERO EXCEPTION **

         if (FPR1_E == #0)
           goto T_TAN9
         endif
```

```
;******** ESCAPE SIGN & EXPORNENT **

        CMP FPR1_E,#ZEROEX+1    ;CY(if |X|<1)

        A = FPR1_H
        A &= #80H               ;Z (rev. sign bit)

        PUSH PSW

        CLR1 FPR1_H.7

;******** TRANS. X TO (|X|-1)/(|X|+1) case if |X| >= 1 **

        if_bit (!CY)
          CALL !LLD31

          DE = #C_1
          CALL !LLD2C
          CALL !LADD
          CALL !LXC13       ;esc. |X|+1 to FPR3

          DE = #C_1
          CALL !LLD2C
          SET1 FPR2_H.7
          CALL !LADD

          CALL !LLD23       ;ret. |X|+1
          CALL !LDIV        ;(|X|-1)/(|X|+1) : X'
        endif

;******** CALC POLINOMIAL FUNCTION **

        CALL !LLD41X      ;esc. x' to FPR4
        CALL !LLD21X

        CALL !LMLTX       ; x'*x'
        CALL !LXC14X      ;set x'*x' to FPR4
```

```
        CALL !LLD31X      ;esc. x' to FPR3
        DE = #C_K
        CALL !LLD2CX
        PUSH DE
        CALL !LMLTX       ; AO*x'
        POP DE

        CALL !LXC13X      ;set AO*x' to FPR3
                          ;set x' to FPR1
        C = #S_PLY
        CALL !LPLY2

;******** CALC. APPROXIMATE **

        POP PSW
        PUSH PSW

        if_bit (!CY)      ; X >= 1 ?
          DE = #C_2
          CALL !LLD2CX
          CALL !LADDX     ; pai/4 + arctan(x')
        endif

;******** RETURN SIGN BIT **

        POP PSW
        if_bit (!Z)       ; X < 0 ?
          SET1 FPR1_H.7
        endif
T_TAN9:
        A = #R_OK
        CLR1 CY
        RET

C_1:
        DB      000H,000H,000H,081H ;const 1
C_2:
        DB 0A2H,0DAH,00FH,049H,080H ;       pai/4
```

```
C_K:                                ;coefficient array(An)
                                    ;     of Hastings approximate
        DB 0CEH,0F4H,0FFH,07FH,080H ;cof. A0
        DB 0E2H,01BH,0A6H,0AAH,07FH ;     A1
        DB 0B0H,093H,034H,099H,080H ;     A2/A1
        DB 03DH,0A4H,081H,0B2H,080H ;     A3/A2
        DB 0B7H,06CH,078H,0B1H,080H ;     A4/A3
        DB 097H,08AH,071H,094H,080H ;     A5/A4
        DB 0FBH,03CH,032H,0C8H,07FH ;     A6/A5
        DB 0BAH,052H,0E5H,0BDH,07EH ;     A7/A6

        END
```

(21) **LHSIN. SRC**

```
$       TITLE     ('HYPERBOLICSINE FUNCTION')
        NAME      M_LHSIN

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN     LADD,LMLT
        EXTRN     LPLY
        EXTRN     LEXP,LRCPN

        PUBLIC    LHSIN

S_PLY   EQU       3

        CSEG

;*********************************************************
;*
;*   FLOATING POINT HYPERBOLICSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- sinh(x)
;*                        ERROR then set CY
;*
;*********************************************************
LHSIN:

;******** CALC. (exp(|X|)-exp(-|X|))/2  case if |X| >= 0.5 **

         if_bit (FPR1_E.7)  ; |X| >= 0.5

           A = FPR1_H
           PUSH AX          ;esc sign bit

           CLR1 FPR1_H.7    ; x <- |x|
```

```
          CALL !LEXP        ; exp(|x|)
          if_bit (CY)
            POP BC
            RET             ;overflow
          endif

          CALL !LLD31       ;esc. exp(|x|) to FPR3
          CALL !LRCPN       ; exp(-|x|)

          SET1 FPR1_H.7
          CALL !LLD23
          CALL !LADD        ; exp(|x|)-exp(-|x|)
          FPR1_E--          ; (exp(|x|)-exp(-|x|))/2

          POP AX
          if_bit (A.7)
            SET1 FPR1_H.7
          endif

;******** CALC. DIRECT APPROXIMATE      case if |X| <  0.5 **

        else                ; |X| < 0.5

          CALL !LLD41

          CALL !LLD21
          CALL !LMLT        ;X*X

          CALL !LXC14X
          FPR1_X = #0

          DE = #C_K
          C  = #S_PLY
          CALL !LPLY
        endif

        A = #R_OK
        CLR1 CY
        RET
```

```
C_K:                                    ; coefficient array of LPLY
        DB 0AAH,0AAH,0AAH,02AH,07EH ; const 1/6
        DB 0CCH,0CCH,0CCH,04CH,07CH ;       1/20
        DB 0C3H,030H,00CH,043H,07BH ;       1/42

        END
```

(22) LHCOS. SRC

```
$       TITLE       ('HYPERBOLICCOSINE FUNCTION')
        NAME        M_LHCOS

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LADD
        EXTRN       LRCPN
        EXTRN       LEXP

        PUBLIC      LHCOS

        CSEG

;*********************************************************
;*
;*   FLOATING POINT HYPERBOLICCOSINE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- cosh(x)
;*                        ERROR then set CY
;*
;*********************************************************
LHCOS:

         CLR1 FPR1_H.7

;******** CALC. (exp(|X|)+exp(-|X|))/2 **

         CALL !LEXP         ; exp(|x|)
         if_bit (CY)
           RET              ;overflow
         endif

         CALL !LLD31
         CALL !LRCPN        ; exp(-|x|)

         CALL !LLD23
         CALL !LADD         ; exp(|x|)+exp(-|x|)
         FPR1_E--           ; (exp(|x|)+exp(-|x|))/2
         RET

         END
```

(23) LHTAN. SRC

```
$       TITLE     ('HYPERBOLICTANGENT FUNCTION')
        NAME      M_LHTAN

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN     LDIV
        EXTRN     LHSIN,LHCOS

        PUBLIC    LHTAN

        CSEG

;*********************************************************
;*
;*   FLOATING POINT HYPERBOLICTANGENT FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- tanh(x)
;*
;*********************************************************
LHTAN:

        CALL !LLD51         ; esc X to FPR5

;******** CALC. LHCOS **

        CALL !LHCOS
        if_bit (CY)
          DE = #C_1
          CALL !LLD1C       ; tanh(positive infinity)=1
          if_bit (FPR5+1+2.7)
            SET1 FPR1_H.7   ; tanh(negative infinity)=-1
          endif
          A = #R_OK
          CLR1 CY
          RET
        endif

        CALL !LXC15         ; esc cosh(X) to FPR5
```

```
;******** CALC. LHSIN **

        CALL !LHSIN         ; sinh(X)

;******** CALC. LHTAN **

        CALL !LLD25         ; ret cosh(X)
        goto  LDIV

C_1:
        DB 000H,000H,000H,081H ; const 1

        END
```

(24) LABS. SRC

```
$       TITLE       ('ABSOLUTE FUNCTION')
        NAME        M_LABS

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)

        PUBLIC      LABS

        CSEG

;*********************************************************
;*
;*   FLOATING POINT ABSOLUTE FUNCTION
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <-  |x|
;*
;*********************************************************
LABS:
        CLR1 FPR1_H.7

        A = #R_OK
        CLR1 CY
        RET

        END
```

(25) LRCPN. SRC

```
$       TITLE       ('RECIPROCAL NUMBER FUNCTION')
        NAME        M_LRCPN

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LDIV

        PUBLIC      LRCPN

        CSEG

;*******************************************************
;*
;*   FLOATING POINT FUNCTION THAT
;*                  GET RECIPROCAL NUMBER
;*
;*     input  condition : FPR1 <- x
;*
;*     output conditions: FPR1 <- 1/x
;*                        ERROR then set CY
;*
;*******************************************************
LRCPN:
        CALL !LLD21
        DE = #C_1
        CALL !LLD1C

        goto LDIV

C_1:
        DB 000H,000H,000H,081H ;const 1

        END
```

(26) POTORA. SRC

```
$       TITLE       ('TRANS. TO RIGHT ANGLE COORDINATES')
        NAME        M_POTORA

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LMLT
        EXTRN       LMOD90,LSIN90,LCOS90

        PUBLIC      POTORA

        CSEG

;*********************************************************
;*
;*   FLOATING POINT FUNCTION THAT
;*       TRANS. COORDINATES FROM POLE TO RIGHT ANGLE
;*
;*     input  condition : FPR1 <- r, FPR2 <- sheeta
;*
;*     output conditions: FPR1 <- X, FPR2 <- Y
;*                        ERROR then set CY
;*
;*********************************************************
POTORA:

;********* EXCEPTION **

          if (FPR1_E == #0)     ; r=0
            FPR2_E = #0         ;  then (x,y)=(0,0)
            goto T_ORA9
          endif

          if_bit (FPR1_H.7)     ; r<0 then error
            A = #R_ERR
            SET1 CY
            RET
          endif
```

```
;********* TRANSLATE **

          A = FPR2            ;load FPR1,sheeta
          A <-> FPR1
          A <-> C

          A = FPR2_M
          A <-> FPR1_M
          A <-> B

          A = FPR2_H
          A <-> FPR1_H
          A <-> X

          A = FPR2_E
          A <-> FPR1_E

          PUSH AX             ;esc r to stack
          PUSH BC

          CALL !LMOD90

          FPR5_1 = FPR4_1     ;esc n
          CALL !LLD51         ;esc sheeta'

          CALL !LSIN90

          CALL !LXC15         ;esc sin(sheeta)
          FPR4_1 = FPR5_1     ;ret n

          CALL !LCOS90

          CALL !LXC15         ;esc cos(sheeta)

          POP BC              ;ret r
          POP AX

          FPR3+4 = A          ;esc r to FPR3
          FPR3+3 = X (A)
          FPR3+2 = B (A)
          FPR3+1 = C (A)
```

```
            CALL !LLD23
            CALL !LMLT          ;Y

            CALL !LXC15         ;esc Y & set cos(sheeta) to FPR1
            CALL !LLD23         ;        set r           to FPR2
            CALL !LMLT          ;X

            CALL !LLD25         ;ret Y
T_ORA9:
            A = #R_OK
            CLR1 CY
            RET

            END
```

(27) RATOPO. SRC

```
$       TITLE       ('TRANS. TO POLE COORDINATES')
        NAME        M_RATOPO

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

        EXTRN       LMLT,LDIV
        EXTRN       LADDX
        EXTRN       LSQRT,LATAN

        PUBLIC      RATOPO

        CSEG

;*********************************************************
;*
;*   FLOATING POINT FUNCTION THAT
;*       TRANS. COORDINATES FROM RIGHT ANGLE TO POLE
;*
;*     input  condition : FPR1 <- X, FPR2 <- Y
;*
;*     output conditions: FPR1 <- r, FPR2 <- sheeta
;*                        ERROR then set CY
;*
;*********************************************************
RATOPO:

;********* Y=0 EXCEPTION & KEEP SIGN BIT**

          if (FPR2_E == #0)
            if (FPR1_E == #0)
              goto T_OPO9
            endif
            CLR1 FPR5_1.1
          else
            CY = FPR2_H.7
            FPR5_1.1 = CY   ;esc sign of Y to FPR5_1.1
          endif

          CY = FPR1_H.7
          FPR5_1.0 = CY     ;esc sign of X to FPR5_1.0
```

```
;********* CALC. X*X+Y*Y **

          CALL !LLD41       ;esc. x to FPR4
          CALL !LLD52       ;esc. y to FPR5

          CALL !LLD21
          CALL !LMLT        ; x*x
          if_bit (CY)
            RET
          endif

          CALL !LLD31X      ;esc. x*x to FPR3

          CALL !LLD15
          CALL !LLD21
          CALL !LMLT        ; y*y
          if_bit (CY)
            RET
          endif

          CALL !LLD23X
          CALL !LADDX       ; x*x + y*y
          if_bit (CY)
            RET
          endif

;********* CALC. Y/X **

          CALL !LXC15       ;esc. X*X+Y*Y to FPR5
          CALL !LLD24

          CALL !LDIV                ; y/x
          if_bit (CY)
            DE = #C_1
            CALL !LLD1C             ; sheeta=pai/2
            if_bit (FPR5_1.1)
              SET1 FPR1_H.7         ; sheeta=-pai/2
            endif
            goto T_OPO8
          endif
```

```
;********* CALC. sheeta **

          CALL !LATAN

          if_bit (FPR5_1.0)       ; x<0 ?
            DE = #C_2
            CALL !LLD2CX          ; sheeta = arctan(y/x) + pai
            if_bit (FPR5_1.1)
              SET1 FPR2_H.7       ; sheeta = arctan(y/x) - pai
            endif
            CALL !LADDX
          endif

;********* CALC. r **

T_OP08:
          CALL !LXC15
          CALL !LSQRT

          CALL !LLD25           ;ret sheeta
T_OP09:
          A = #R_OK
          CLR1 CY
          RET

C_1:


          DB      0DAH,00FH,049H,081H ;const pai/2
C_2:
          DB 0A2H,0DAH,00FH,049H,082H ;       pai

          END
```

(28) ATOL. SRC

```
$         TITLE     ('TRANSLATE ASCII STRING')
          NAME      M_ATOL

$ INCLUDE(EQU.INC)
$ INCLUDE(ASCII.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

          EXTRN     FTOL,LTOF
          EXTRN     LADDX,LMLTX
          EXTRN     LEXPX

          PUBLIC    ATOL

S_VIRT    EQU       27          ; maximum length of mantissa

          CSEG

;*********************************************************
;*
;*   FLOATING POINT FUNCTION THAT
;*            TRANSLATE ASCII STRING
;*
;*     input  condition : HL <- HEAD ADDRESS of STRING
;*
;*     output conditions: FPR1 <- (REAL VALUE MEANING STRING)
;*                        ERROR then set CY
;*
;*********************************************************
ATOL:

;********* TRANS. SIGN **

          FPR5_1 = #0               ;sign keeper

          CALL !GETC
          if     (A == #N_PL)
            CALL !GETC
          elseif (A == #N_MN)
            SET1 FPR5_1.7
            CALL !GETC
          endif
```

```
;********* TRANS. MANTISSA TO BINARY **

          PUSH HL

          DE = #0
          HL = DE

          SET1 FPR4_1.7      ;work: decimal digit of mantissa
          SET1 FPR4_2.7      ;    : negrect digit of ""

          SET1 X.7           ;    : mantissa digit counter

          while (A <= #N_9 || A == #N_PD) ;'0'-'9','.'
            if (A <= #N_9)
              if_bit (!FPR4_1.7)
                FPR4_1++               ;decimal digit count up
              endif

              if_bit (X.7)
                A <-> L                ;initial digit
                X = #S_VIRT-1+1

              else

                X--
                if_bit (Z)
                  goto ERROR2          ;mantissa length over
                endif
                if_bit (FPR4_2.7)
                  PUSH AX
                  C = A

                  ; *10 +A
                  B = #10
                  A = L
                  MULU B
                  X += C       ;->CY
                  L = X
                  A <-> C

                  A = H
                  MULU B
                  ADDC X,C     ;<-CY,->CY
                  H = X
                  A <-> C
```

```
        A = E
        MULU B
        ADDC X,C      ;<-CY,->CY
        E = X

        ADDC A,#0     ;<-CY
        A <->D

        POP AX

        if_bit (!Z)           ;work area fill
          FPR4_2 = #0         ; then negrect forword digit
        endif
      else
        FPR4_2++              ;negrect digit count up
      endif

    endif

  else       ; '.'

    if_bit (FPR4_1.7)
      FPR4_1 = #0             ; begin count of decimal digit
    else
      goto ERROR2             ; duplicate
    endif
  endif

  POP BC
  PUSH DE
  PUSH HL
  HL = BC
  CALL !GETC
  BC = HL
  POP HL
  POP DE
  PUSH BC
endw

if (A >= #N_MN)
  goto ERROR2
endif

if_bit (X.7)          ;no mantissa digit
  goto ERROR2
endif
```

```
          PUSH AX

          A = FPR4_2
          if_bit (A.7)
            A = #0
          endif

          A <-> X
          A = FPR4_1
          if_bit (A.7)
            A = #0
          endif

          A -= X                ;decimal digit - negrect digit
          FPR4_1 = A            ;esc. decimal digit (:F-N)

;********* NORMALIZE MANTISSA val. **

          AX = DE
          if (AX == #0)
            A <-> H
            X <-> L
            if (AX == #0)
              BC = AX
            else
              BC = #BYTE*(SHORT-2)*100H+ZEROEX
              while_bit (!A.7)
                SHLW AX,1
                B--
              endw
            endif
          else
            BC = #BYTE*SHORT*100H+ZEROEX
            while_bit (!A.7)
              SHLW HL,1
              ROLC X,1
              ROLC A,1
              B--
            endw
          endif

          if_bit (!FPR5_1.7)
            A &= #7FH           ;ret sign bit
          endif
          FPR5+1+2 = A          ;esc mantissa val(x1: .5<=x1<1)
          FPR5+1+1 = X (A)
```

```
          FPR5+1   = H (A)
          FPR5+1+3 = C (A)
          FPR4_2   = B (A)    ;esc mantissa exp of 2 (y1)

;********** TRANS. EXP_PART **

          POP AX
          POP HL
          X = #0              ;work exp val
          C = #0              ;     sign of exp

          if (A <= #N_E)      ;'E' or 'e'
            CALL !GETC
            if     (A == #N_PL)
              CALL !GETC
            elseif (A == #N_MN)
              C = #0FFH
              CALL !GETC
            endif
            if (A > #N_9)
              goto ERROR
            endif

            A <-> X             ;1st. digit
            CALL !GETC
            if (A <= #N_9)
              A <-> D
              A = X
              E = #10
              MULU E
              X += D            ;1st.digit * 10 + 2nd.digit
              CALL !GETC
            endif
          endif

          if (A != #N_NL && A != #N_SP)
            goto ERROR
          endif

          A = C
          A <-> X             ;A:exp-part value (:B)
          if_bit (X.7)
            A ^= #0FFH
            A++
          endif
```

```
;********* UNITE MANTISSA.val & EXP.val **

        A -= FPR4_1         ; B - (F-N) (:B')
        X = #0
        if_bit (A.7)
          X = #0FFH
        endif

        A <-> X
        DE = AX
        CALL !FTOL          ; B' → real
        DE = #C_1
        CALL !LLD2CX
        CALL !LMLTX         ;log2(10) * B'

        CALL !LTOF
        if_bit (Z)
          FPR1_E = #0         ; dec(log2(10)*B') ← 0
        else
          if_bit (FPR1_H.7)
            DE--
          endif
          CALL !LLD21X
          CALL !FTOL
          NOT1 FPR1_H.7
          PUSH DE
          CALL !LADDX       ;dec(log2(10)*B')
          POP DE
        endif

        A = FPR4_2
        X = #0
        A <-> X
        AX += DE
        PUSH AX             ;int(log2(10)*B')+Y1

        DE = #C_2
        CALL !LLD2CX
        CALL !LMLTX         ;dec(log2(10)*B')*log(2)
        CALL !LEXPX

        CALL !LLD25         ;ret. ±X1
        FPR2_X = #0
        CALL !LMLTX         ;±X1 * exp(dec(log2(10)*B')*log(2))
```

```
          A = FPR1_E
          X = #0
          A <-> X

          POP BC
          AX += BC              ;exp.part result
          if_bit (A.7)
            A = #0
          elseif (A != #0)
            goto ERROR
          else
            A = X
          endif

          FPR1_E = A

          A = #R_OK
          CLR1 CY

          RET
ERROR2:
          POP HL
ERROR:
          A = #R_ERR
          SET1 CY
          RET

GETC:
          A = [HL+]
          DE = #INDEX
          B = #S_INDX
          repeat
            if (A == [DE+])
              break
            endif
            B--
          until_bit (Z)

          B--
          A = B
          RET

INDEX:
          @_INDX
```

```
C_1:
        DB 04BH,078H,09AH,054H,082H ;const. log2(10)
C_2:
        DB 0F7H,017H,072H,031H,080H ;       log(2)

        END
```

(29) LTOA. SRC

```
$         TITLE     ('TRANSLATE TO ASCII STRING')
          NAME      M_LTOA

$ INCLUDE(EQU.INC)
$ INCLUDE(ASCII.INC)
$ INCLUDE(REF1.INC)
$ INCLUDE(REF2.INC)

          EXTRN     LADD,LMLT
          EXTRN     LADDX,LMLTX
          EXTRN     LTOF,FTOL
          EXTRN     LLOG10,LEXP10

          PUBLIC    LTOA

S_VIRT    EQU       7           ; string length of mantissa

C_LIM     EQU       38          ; max expression of exp10

C_2H      EQU       20H
C_2E      EQU       84H

          CSEG

;*********************************************************
;*
;*   FLOATING POINT FUNCTION THAT
;*            TRANSLATE TO ASCII STRING
;*
;*     input  condition : FPR1 <- x
;*                       HL <- STORE ADDRESS of STRING
;*
;*     output conditions: STRING HEAD IS APPOINTED TO HL
;*
;*********************************************************
LTOA:
```

```
;********* ZERO FORMAT **

          if (FPR1_E == #0)
            [HL+] = #A_0 (A)
            [HL]  = #A_NL (A)
            goto T_TOA9
          endif

;********* TRANS. to a * exp10(b) (1<= a <10) **

          PUSH HL

          CALL !LLD51               ;esc x to FPR5
          CLR1 FPR1_H.7             ; x <- |x|
          CALL !LLOG10              ; log10(|x|)

          CALL !LTOF                ; trunc integer(log10(|x|)
          if_bit (FPR1_H.7 && !Z)   ; include decimal digit ?
            DE--                    ; floor integer(log10(|x|)  : b
          endif

          PUSH DE                   ;esc. b to STACK

          A = E
          if_bit (A.7)              ; b<0 ?
            A += #C_LIM
            if_bit (!CY)            ; b<(-C_LIM) ?
              E++
              E++                   ;  then coordinate b to caluculatable range
              PUSH DE               ; b'
              CALL !LLD15
              DE = #C_1
              CALL !LLD2C
              CALL !LMLT            ; x*100 :X'
              CALL !LLD51
              POP DE
            endif
          endif
```

```
          CALL !FTOL              ; real(b')
          NOT1 FPR1_H.7
          CALL !LEXP10            ; exp10(-b')
          CALL !LLD25
          FPR2_X = #0
          CALL !LMLTX             ; X' * exp10(-b')   : a

          POP DE
          FPR3_1 = E (A)          ;esc b to FPR3_1

;********* OUTPUT MANTISSA OF EXP10 **

          if_bit (FPR1_H.7)       ; a<0 ?
            POP HL
            [HL+] = #A_MN (A)
            PUSH HL
            CLR1 FPR1_H.7         ; a <- |a|
          endif

          if (FPR1_E == #C_2E)
            CMP FPR1_H,#C_2H
          endif

          if_bit (!CY)            ;limit |a|<10 over then
            DE = #C_3             ;  normalize
            CALL !LLD2CX
            CALL !LMLTX
            FPR3_1++
          endif

          if (FPR1_E < #ZEROEX+1) ;limit |a|>=1 over then
            DE = #C_2             ;  normalize
            CALL !LLD2CX
            CALL !LMLTX
            FPR3_1--
          endif

          CALL !LTOF              ; integer(a)
          A = E
          A += #A_0
          POP HL
```

```
          [HL+] = A
          [HL+] = #A_PD (A)

          B = #S_VIRT-1
          repeat
            PUSH BC
            PUSH HL

            CALL !LLD21X
            CALL !FTOL
            SET1 FPR1_H.7
            CALL !LADDX           ; a - integer(a)
            DE = #C_2
            CALL !LLD2CX
            CALL !LMLTX           ; (a - integer(a))*10

            CALL !LTOF            ; integer(a)
            A = E
            A += #A_0

            POP HL
            [HL+] = A

            POP BC
            B--
          until_bit (Z)

;********* OUTPUT EXP.PART OF EXP10 **

          [HL+] = #A_E (A)

          if_bit (FPR3_1.7)
            [HL+] = #A_MN (A)
            A = #0
            A -= FPR3_1
          else
            A = FPR3_1
          endif
```

```
            X = #0
            A <-> X

            B = #10
            DIVUW B
            A = X

            A += #A_0
            [HL+] = A
            A = B
            A += #A_0
            [HL+] = A
            [HL] =#A_NL (A)
T_TOA9:
            A = #R_OK
            CLR1 CY
            RET

C_1:
            DB      000H,000H,048H,087H ;const 100
C_2:
            DB 000H,000H,000H,C_2H,C_2E ;const 10
C_3:
            DB 0CDH,0CCH,0CCH,04CH,07DH ;const 1/10

            END
```

(30) FTOL. SRC

```
$       TITLE       ('TRANS. FIXED TO REAL')
        NAME        M_FTOL

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)

        PUBLIC      FTOL

        CSEG

;*********************************************************
;*
;*  FUNCTION THAT TRANSLATE FIXED TO REAL
;*
;*    input  condition : DE <- (integer with sign bit)
;*
;*    output condition : FPR1 <- (real value meaning DE)
;*                       DE keep
;*
;*********************************************************
FTOL:

;********* ZERO EXCEPTION **

          AX = DE
          if (AX == #0)
            FPR1_E = #0
            goto T_TOL9
          endif

;********* GET ABSOLUTE VALUE **

          if (AX >= #8000H+1)
            AX = #0
            AX -= DE
          endif
```

```
;********* TRANSLATE **

          if (A == #0)
            FPR1_E = #ZEROEX+BYTE
            A <-> X
          else
            FPR1_E = #ZEROEX+BYTE*INTEGR
          endif

          while_bit (!A.7)
            SHLW AX,1
            FPR1_E--
          endw

;********* SET VIRT.PART & SIGN BIT **

          FPR1_H = A
          FPR1_M = X (A)
          FPR1   = #0
          FPR1_X = #0

          A = D
          if_bit (!A.7)
            CLR1 FPR1_H.7
          endif

T_TOL9:
          A = #R_OK
          CLR1 CY
          RET

          END
```

(31) LTOF. SRC

```
$       TITLE       ('TRANS. REAL TO FIXED')
        NAME        M_LTOF

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)

        PUBLIC      LTOF

        CSEG

;*********************************************************
;*
;*  FUNCTION THAT TRANSLATE
;*                REAL TO FIXED
;*
;*    input  condition : FPR1 <- (real value)
;*
;*    output condition : DE <- (integer value meaning FPR1)
;*                       ERROR then set CY
;*                       not INCLUDE DECIMAL PART then set Z
;*                keep : FPR1
;*
;*********************************************************
LTOF:

          A = FPR1_E
          A -= #ZEROEX+1

;********* EXCEPTION **

          if_bit (CY)                ; integer(FPR1)=0 ?
            CMP A,#LOW(-(ZEROEX+1))
            DE = #0
            goto T_TOF9
          endif

          A -= #BYTE*INTEGR
          if_bit (!CY)
            goto ERROR               ;overflow
          endif
```

```
;********* GET UNSIGNED INTEGER **

          A <-> X
          A = FPR1_H
          A != #80H
          A <-> B
          A = FPR1_M
          A <-> C
          A <-> X

          DE = #0
          if (FPR1 != #0)
            D = #80H
          endif

          if_bit (!A.3)
            C <-> E
            B <-> C
            A += #BYTE
          endif

          A++
          if_bit (!Z)
            repeat
              SHRW BC,1
              RORC D,1
              A++
            until_bit (Z)
          endif

          X = C
          A = B

;********* UNSIGNED --> SIGNED INTEGER **

          if (AX >= #8001H)
            goto ERROR
          endif
          if_bit (A.7)          ;(AX == #8000H)
            if_bit (!FPR1_H.7)
              goto ERROR
            endif
          else
```

```
            if_bit (FPR1_H.7)
              A ^= #0FFH
              A <-> X
              A ^= #0FFH
              A <-> X
              AX++
            endif
          endif

          A <-> D
          X <-> E

;********* DECIMAL PART JUDGE **

          CMPW AX,#0

T_TOF9:
          A = #R_OK
          CLR1 CY
          RET
ERROR:
          A = #R_ERR
          SET1 CY
          RET

          END
```

(32) LTO78. SRC

```
$       TITLE       ('TRANS. FORMAT TO 78K/2')
        NAME        M_LTO78

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)

        PUBLIC      LTO78

        CSEG

;*********************************************************
;*
;*  FUNCTION THAT TRANSLATE FORMAT
;*                IEEE-754 TO 78K/2
;*
;*    input  condition : FPR1 <- (real:format by iEEE-754)
;*
;*    output condition : FPR1 <- (real:format by 78k/2)
;*                       ERROR then set CY
;*
;*********************************************************
LTO78:

;********* TRANS. BIT FORMAT **

          A = FPR1_E
          CY = FPR1_H.7
          ROLC A,1
          FPR1_H.7 = CY

;********* ADJUST EXP.PART **

          if (A != #0)
            A += #2
            if_bit (CY)
              A = #R_ERR
              RET
            endif
          endif

          FPR1_E = A

          A = #R_OK
          RET

          END
```

(33) LTOIE. SRC

```
$       TITLE       ('TRANS. FORMAT TO iEEE-754')
        NAME        M_LTOIE

$ INCLUDE(EQU.INC)
$ INCLUDE(REF1.INC)

        PUBLIC      LTOIE

        CSEG

;*********************************************************
;*
;*  FUNCTION THAT TRANSLATE FORMAT
;*                78K/2 TO IEEE-754
;*
;*    input  condition : FPR1 <- (real:format by 78K/2)
;*
;*    output condition : FPR1 <- (real:format by iEEE-754)
;*
;*********************************************************
LTOIE:

;********* ADJUST EXP.PART **

          A = FPR1_E
          A -= #3
          if_bit (CY)
            FPR1   = #0
            FPR1_M = #0
            FPR1_H = #0
            FPR1_E = #0
            goto T_OIE9
          endif

          A++

;********* TRANS. BIT FORMAT **

          CY = FPR1_H.7
          RORC A,1
          FPR1_E = A
          FPR1_H.7 = CY
T_OIE9:
          A = #R_OK
          CLR1 CY
          RET

          END
```

お問い合わせは，最寄りのNECへ

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-01　東京都港区芝五丁目7番1号（NEC本社ビル）	東　京	(03)3454-1111　(大代表)
中部支社 半導体第一販売部 半導体第二販売部	〒460　名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋 名古屋	(052)222-2170 (052)222-2190
関西支社 半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540　大阪市中央区城見一丁目4番24号（NEC関西ビル）	大　阪 大　阪 大　阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社 札幌	(011)231-0161	太田支店 太田	(0276)46-4011	富山支店 富山	(0764)31-8461
東北支社 仙台	(022)267-8740	宇都宮支店 宇都宮	(028)621-2281	三重支店 津	(0592)25-7341
岩手支店 盛岡	(019)651-4344	小山支店 小山	(0285)24-5011	京都支社 京都	(075)344-7824
山形支店 山形	(0236)23-5511	長野支社 松本	(0263)35-1662	神戸支社 神戸	(078)333-3854
郡山支店 郡山	(0249)23-5511	甲府支店 甲府	(0552)24-4141	中国支社 広島	(082)242-5504
いわき支店 いわき	(0246)21-5511	埼玉支社 大宮	(048)641-1411	鳥取支店 鳥取	(0857)27-5311
長岡支店 長岡	(0258)36-2155	立川支社 立川	(0425)26-5981	岡山支店 岡山	(086)225-4455
土浦支店 土浦	(0298)23-6161	千葉支社 千葉	(043)238-8116	四国支社 高松	(0878)36-1200
水戸支店 水戸	(029)226-1717	静岡支社 静岡	(054)255-2211	新居浜支店 新居浜	(0897)32-5001
神奈川支社 横浜	(045)324-5524	北陸支社 金沢	(0762)23-1621	松山支店 松山	(089)945-4149
群馬支店 高崎	(0273)26-1255	福井支店 福井	(0776)22-1866	九州支社 福岡	(092)271-7700

【本資料に関する技術お問い合わせ先】

半導体ソリューション技術本部 マイクロコンピュータ技術部	〒210　川崎市幸区塚越三丁目484番地	川　崎	(044)548-7924
半導体販売技術本部 東日本販売技術部	〒108-01　東京都港区芝五丁目7番1号（NEC本社ビル）	東　京	(03)3798-9619
半導体販売技術本部 中部販売技術部	〒460　名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋	(052)222-2125
半導体販売技術本部 西日本販売技術部	〒540　大阪市中央区城見一丁目4番24号（NEC関西ビル）	大　阪	(06) 945-3383

半導体
インフォメーションセンター
FAX(044)548-7900
(FAXにてお願い致します)

C97.1

キリトリ

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］78K/IIシリーズ　アプリケーション・ノート　浮動小数点演算プログラム編
（IEA-686D（第5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，NEC半導体ソリューション技術本部員，
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。

NEC半導体インフォメーションセンター
FAX：（044）548-7900