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

保守／廃止

78K/ IIIシリーズ

16/8 ビット・シングルチップ・マイクロコンピュータ

ソフトウエア基礎編

μPD78322 サブシリーズ
μPD78328 サブシリーズ
μPD78334 サブシリーズ
μPD78352A サブシリーズ
μPD78356 サブシリーズ
μPD78366A サブシリーズ
μPD78372 サブシリーズ

資料番号　U12118JJ4V1AN00（第４版）
（旧資料番号　IEA-684）
発行年月　September 1997　N

保守／廃止

〔メ　モ〕

保守／廃止

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。

保守／廃止

本版で改訂された主な箇所

箇　所	内　容
全　般	次の製品を追加 μPD78356(A), 78P356(A), 78361A, 78362A, 78P364A, 78363A, 78365A, 78366A, 78368A, 78P368A, 78372(A), 78372(A1), 78372(A2), 78P372(A), 78P372(A1), 78P372(A2)
	次の製品を削除 μPD78355A, 78356A, 78P356A, 78362, 78P364, 78365, 78366, 78P368, 78370, 78372, 78P372
	次の製品を開発中→開発済みに変更 μPD78P324, 78P324(A), 78P324(A1), 78P324(A2), 78350A, 78355, 78356, 78P356
p. 143	**付録B　改版履歴**を追加

本文欄外の★印は，本版で改訂された主な箇所を示しています。

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

保守／廃止

はじめに

78K/IIIシリーズは，78Kシリーズに属する16／8ビット・シングルチップ・マイクロコンピュータです。特に制御用途に適した強力な命令セットを備え，品種ごとの各種アプリケーションに適した周辺ハードウエアを内蔵した製品を展開しています。

このアプリケーション・ノートで扱う，μPD78322，78328，78334，78352A，78356，78366A，78372の7つのサブシリーズは，78K/IIIシリーズの中で，μPD78312Aの各種機能の強化拡大とともに命令セットの強化補充をしたものです。

このアプリケーション・ノートは，これら7つのサブシリーズの基本的なプログラム例と，μPD78312Aにはなかった命令の説明とをまとめました。

★ **78K/IIIシリーズ製品一覧**

サブシリーズ名	品名
μPD78312A	μPD78310A, 78312A, 78P312A
μPD78322	μPD78320, 78322, 78P322, 78323, 78324, 78P324, 78320(A), 78320(A1), 78320(A2), 78322(A), 78322(A1), 78322(A2), 78323(A), 78323(A1), 78323(A2), 78324(A), 78324(A1), 78324(A2), 78P324(A), 78P324(A1), 78P324(A2)
μPD78328	μPD78327, 78328, 78P328, 78327(A), 78328(A)
μPD78334	μPD78330, 78334, 78P334, 78330(A), 78330(A1), 78330(A2), 78334(A), 78334(A1), 78334(A2), 78P334(A), 78P334(A1), 78P334(A2)
μPD78352A	μPD78350, 78350A,, 78352A, 78P352
μPD78356	μPD78355, 78356, 78P356, 78356(A), 78P356(A)
μPD78366A	μPD78361A[注], 78362A, 78P364A, 78363A, 78365A, 78366A, 78368A[注], 78P368A
μPD78372	μPD78372(A), 78372(A1), 78372(A2), 78P372(A), 78P372(A1), 78P372(A2)

注　開発中

μPD78312Aと共通の命令につきましては，**μPD78312Aアプリケーション・ノート（I）（IEM-964），同（II）（IEA-628）**をご参照ください。

このアプリケーション・ノートでは，μPD78320を代表品種として扱っています。

μPD78320以外の製品をお使いになる場合は，この資料のμPD78320をご使用製品名に読み替えてください。

本資料に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。

78K/IIIシリーズの命令セットは，オペランド欄のレジスタ名称の記述方法として絶対名称と機能名称とを使い分けることができます。

絶対名称は２種類の機能名称セット(RSS＝0，1)に対応しています。どちらのセットを各機能レジスタに割り当てるかは，アセンブラの疑似命令“RSS”により指定することができます。

詳細は，各製品のユーザーズ・マニュアルまたはアセンブラのユーザーズ・マニュアルを参照してください。

レジスタの絶対名称-機能名称対応表

絶対名称	機能名称	
	RSS＝0	RSS＝1
R0	X	
R1	A	
R2	C	
R3	B	
R4		X
R5		A
R6		C
R7		B
R8	VP_L	VP_L
R9	VP_H	VP_H
R10	UP_L	UP_L
R11	UP_H	UP_H
R12	E	E
R13	D	D
R14	L	L
R15	H	H
RP0	AX	
RP1	BC	
RP2		AX
RP3		BC
RP4	VP	VP
RP5	UP	UP
RP6	DE	DE
RP7	HL	HL

保守／廃止

★ 品質水準

- 標準品：μPD78320, 78322, 78P322, 78323, 78324, 78P324, 78327, 78328, 78P328, 78330, 78334, 78P334, 78350, 78350A, 78352A, 78P352, 78355, 78356, 78P356, 78361A, 78362A, 78P364A, 78363A, 78365A, 78366A, 78368A, 78P368A
- 特別品：μPD78320(A), 78320(A1), 78320(A2), 78322(A), 78322(A1), 78322(A2), 78323(A), 78323(A1), 78323(A2), 78324(A), 78324(A1), 78324(A2), 78P324(A), 78P324(A1), 78P324(A2), 78327(A), 78328(A), 78330(A), 78330(A1), 78330(A2), 78334(A), 78334(A1), 78334(A2), 78P334(A), 78P334(A1), 78P334(A2), 78356(A), 78P356(A), 78372(A), 78372(A1), 78372(A2), 78P372(A), 78P372(A1), 78P372(A2)

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」（資料番号 C11531J）をご覧ください。

応用分野

〔標準品〕

- モータ制御機器を扱う分野（μPD78322サブシリーズ，78334サブシリーズ，78352Aサブシリーズ，78356サブシリーズ）
- PWMインバータ制御分野，インバータ・エアコン（μPD78328サブシリーズ，78366Aサブシリーズ）

〔特別品〕

- 自動車電装分野（μPD78322サブシリーズ，78328サブシリーズ，78334サブシリーズ，78356サブシリーズ，78372サブシリーズ）

保守／廃止

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

μPD78322サブシリーズ

資料名	資料番号	
	和文	英文
μPD78322　プロダクト・レター	IF-6293	—
μPD78322(A)　プロダクト・レター	IF-6318	—
μPD78320，78322　データ・シート	U10455J	U10455E
μPD78P322　データ・シート	U10435J	U10435E
μPD78323，78324　データ・シート	U10456J	U10456E
μPD78P324，78P324(A)　データ・シート	IC-8315	IC-2857
μPD78320(A)，78322(A)　データ・シート	IC-8327	IC-2879
μPD78323(A)，78324(A)　データ・シート	IC-8712	IC-3211
μPD78322　ユーザーズ・マニュアル	IEU-619	IEU-1248
78K/IIIシリーズ　アプリケーション・ノート　ソフトウエア基礎編	このマニュアル	IEA-1272
78K/III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	U12119J	IEA-1291
μPD78322　特殊機能レジスタ活用表	IEM-5501	—
μPD78322　インストラクション・セット	IEM-601	—
μPD78322　インストラクション活用表	IEM-602	—

μPD78328サブシリーズ

資料名	資料番号	
	和文	英文
μPD78327，78328　データ・シート	U10208J	U10208E
μPD78P328　データ・シート	U10209J	U10209E
μPD78327(A)，78328(A)　データ・シート	IC-8291	IC-2858
μPD78328　ユーザーズ・マニュアル	IEU-693	IEU-1268
μPD78328　アプリケーション・ノート　ハードウエア基礎編	IEA-716	IEA-1287
78K/IIIシリーズ　アプリケーション・ノート　ソフトウエア基礎編	このマニュアル	IEA-1272
78K/III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	U12119J	IEA-1291
μPD78328　特殊機能レジスタ活用表	IEM-5514	—
μPD78322　インストラクション・セット	IEM-601	—
μPD78322　インストラクション活用表	IEM-602	—

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

保守／廃止

μPD78334サブシリーズ

資　料　名	資料番号	
	和文	英文
μPD78334　プロダクト・レター	IF-6340	—
μPD78334(A)　プロダクト・レター	IF-6292	IF-2027
μPD78330，78334　データ・シート	U10185J	U10185E
μPD78P334　データ・シート	IC-8075	IC-2648
μPD78330(A)，78334(A)　データ・シート	IC-8494	IC-3364
μPD78P334(A)，(A1)，(A2)　データ・シート	IC-8804	IC-3264
μPD78334　ユーザーズ・マニュアル	IEU-729	IEU-1315
78K/IIIシリーズ　アプリケーション・ノート　ソフトウエア基礎編	このマニュアル	IEA-1272
78K/III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	U12119J	IEA-1291
μPD78334　特殊機能レジスタ活用表	IEM-5518	—
μPD78322　インストラクション・セット	IEM-601	—
μPD78322　インストラクション活用表	IEM-602	—

μPD78352Aサブシリーズ

資　料　名	資料番号	
	和文	英文
μPD78352A　プロダクト・レター	IF-6335	IF-2036
μPD78350　データ・シート	IC-8279	IC-2845
μPD78350A，78352A　データ・シート	IC-8823	IC-3391
μPD78P352　データ・シート	IC-8423	IC-2957
μPD78352A　ユーザーズ・マニュアル　ハードウエア編	IEU-781	IEU-1327
μPD78356　ユーザーズ・マニュアル　命令編	U12117J	IEU-1395
78K/IIIシリーズ　アプリケーション・ノート　ソフトウエア基礎編	このマニュアル	IEA-1272
78K/III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	U12119J	IEA-1291
μPD78352A　特殊機能レジスタ活用表	IEM-5540	IEM-1215
μPD78352A　インストラクション・セット	U11955J	—

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

保守／廃止

μPD78356サブシリーズ

資　料　名	資料番号	
	和文	英文
μPD78356　プロダクト・レター	IF-6298	—
μPD78355, 78356　データ・シート	U10155J	U10155E
μPD78P356　データ・シート	U10325J	U10325E
μPD78356(A)　データ・シート	U11148J	U11148E
μPD78P356(A)　データ・シート	U11149J	U11149E
μPD78356　ユーザーズ・マニュアル　ハードウエア編	U10669J	U10669E
μPD78356　ユーザーズ・マニュアル　命令編	U12117J	IEU-1395
78K/IIIシリーズ　アプリケーション・ノート　ソフトウエア基礎編	このマニュアル	IEA-1272
78K/III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	U12119J	IEA-1291
μPD78356　特殊機能レジスタ活用表	IEM-5576	IEM-1214
μPD78352A　インストラクション・セット	U11955J	—

μPD78366Aサブシリーズ

資　料　名	資料番号	
	和文	英文
μPD78362A　データ・シート	U10098J	U10098E
μPD78P364A　データ・シート	U10106J	U10106E
μPD78363A, 78365A, 78366A　データ・シート	U11109J	U11109E
μPD78P368A　データ・シート	U11373J	U11373E
μPD78362A　ユーザーズ・マニュアル　ハードウエア編	U10745J	U10745E
μPD78366A　ユーザーズ・マニュアル　ハードウエア編	U10205J	U10205E
μPD78356　ユーザーズ・マニュアル　命令編	U12117J	IEU-1395
78K/IIIシリーズ　アプリケーション・ノート　ソフトウエア基礎編	このマニュアル	IEA-1272
78K/III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	U12119J	IEA-1291
μPD78362A　特殊機能レジスタ活用表	U10210J	—
μPD78366A　特殊機能レジスタ活用表	U10107J	—
μPD78352A　インストラクション・セット	U11955J	—

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

保守／廃止

μPD78372サブシリーズ

資　料　名	資料番号	
	和文	英文
μPD78372　プロダクト・レター	IF-6351	—
μPD78370(A), 78372(A)　データ・シート	U10789J	U10789E
μPD78P372(A)　データ・シート	U12029J	U12029E
μPD78372　ユーザーズ・マニュアル　ハードウエア編	U10642J	U10642E
μPD78356　ユーザーズ・マニュアル　命令編	U12117J	IEU-1395
78K/IIIシリーズ　アプリケーション・ノート　ソフトウエア基礎編	このマニュアル	IEA-1272
78K/III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	U12119J	IEA-1291
μPD78372　特殊機能レジスタ活用表	U10631J	U10631E
μPD78352A　インストラクション・セット	U11955J	—

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

保守／廃止

〔メ　モ〕

保守／廃止

目　　次

保守／廃止

図　の　目　次

表　の　目　次

第１章　78K/IIIシリーズの概要

78K/IIIシリーズは，16ビットCPUを搭載した高速，高機能なCMOS16/8ビット・シングルチップ・マイクロコンピュータです。

以下は主な特徴です。

○制御用途に適した命令セット

- ●16ビット演算命令
- ●乗除算命令（16ビット×16ビット，32ビット÷16ビット）
- ●ビット操作命令
- ●ストリング命令，etc.

○機械制御に適した周辺ハードウエア

- ●A/Dコンバータ
- ●パルス信号の入出力制御に有効なリアルタイム・パルス・ユニット

○高機能割り込みコントローラ内蔵

- ●ベクタ割り込み機能
- ●コンテキスト・スイッチング機能
  （割り込み要求発生時にレジスタ・バンクを切り替え，高速割り込み処理を実現）
- ●マクロ・サービス機能
  （ハードウエアによりデータの転送などの特殊な処理を実行）

○汎用シリアル・インタフェース

各製品の概要を，表１－１に示します。

この資料は，μPD78320の命令セットをベースとして，各プログラムを記述しています。

保守／廃止

表1－1　78K/IIIシリーズ製品概要（1/5）

サブシリーズ名		μPD78312A			μPD78322		
品名		μPD78310A	μPD78312A	μPD78P312A	μPD78320	μPD78322	μPD78P322
命令数		96種			111種		
最小命令実行時間		500 ns/12 MHz			250 ns/16 MHz		
内蔵	ROM容量	——	8Kバイト	8Kバイト（PROM）	——	16Kバイト	16Kバイト（PROM）
	RAM容量	256バイト			640バイト		
割り込み機能	外部要因	4			8		
	内部要因	13			14（外部兼用2）		
		8レベル・プログラマブル・プライオリティベクタ割り込み機能 マクロ・サービス（高速データ転送）機能（1種） コンテキスト・スイッチング機能			3レベル・プログラマブル・プライオリティベクタ割り込み機能 マクロ・サービス（高速データ転送／演算）機能（9種） コンテキスト・スイッチング機能		
テスト要因		——			内部：1		
I/O端子	入力	8本			16本		
	入出力	24本	40本		21本	39本	
A/Dコンバータ		8ビット分解能4チャネル			10ビット分解能8チャネル		
タイマ／カウンタ		16ビット・アップ/ダウン・カウンタ×2 16ビット・インターバル・タイマ×2			18/16ビット・フリー・ランニング・タイマ×1 16ビット・タイマ／イベント・カウンタ×1		
シリアル・インタフェース		UART（専用ボー・レート・ジェネレータ内蔵）…1ch （クロック同期式）			UART（専用ボー・レート・ジェネレータ内蔵）…1ch クロック同期式（SBI対応）…1ch		
スタンバイ機能		HALT/STOPモード					
特徴		・DCサーボ制御に適したアップ／ダウン・カウンタを内蔵			・多彩なプログラマブル・パルス出力が可能なリアルタイム・パルス・ユニット		
パッケージ		64ピンSDIP 64ピンQUIP 64ピンQFP（14×20 mm） 68ピンQFJ	同左 64ピン窓付きSDIP 64ピン窓付きQUIP		68ピンQFJ 74ピンQFP（□20 mm） 80ピンQFP（14×20 mm）	同左 68ピンWQFN 74ピンWQFN 80ピンWQFN	

保守／廃止

表1－1　78K/IIIシリーズ製品概要（2/5）

μPD78322			μPD78328			μPD78334		
μPD78323	μPD78324	μPD78P324	μPD78327	μPD78328	μPD78P328	μPD78330	μPD78334	μPD78P334
111種								
250 ns/16 MHz								
——	32 Kバイト	32 Kバイト（PROM）	——	16 Kバイト	16 Kバイト（PROM）	——	32 Kバイト	32 Kバイト（PROM）
1024バイト			512バイト			1024バイト		
8			4			8		
14			16			14（外部兼用2）		
3レベル・プログラマブル・プライオリティベクタ割り込み機能 マクロ・サービス（高速データ転送／演算）機能（9種） コンテキスト・スイッチング機能			3レベル・プログラマブル・プライオリティベクタ割り込み機能 マクロ・サービス（高速データ転送／演算）機能（8種） コンテキスト・スイッチング機能			3レベル・プログラマブル・プライオリティベクタ割り込み機能 マクロ・サービス（高速データ転送／演算）機能（9種） コンテキスト・スイッチング機能		
内部：1								
16本			11本			24本		
21本	39本		23本	41本		28本	46本	
10ビット分解能8チャネル						10ビット分解能16チャネル		
18/16ビット・フリー・ランニング・タイマ×1 16ビット・タイマ／イベント・カウンタ×1			16ビット・フリー・ランニング・タイマ×2 16ビット・タイマ／イベント・カウンタ×1			18/16ビット・フリー・ランニング・タイマ×1 16ビット・タイマ／カウンタ×3		
UART（専用ボー・レート・ジェネレータ内蔵）…1 ch クロック同期式（SBI対応）…1 ch								
HALT/STOPモード								
・多彩なプログラマブル・パルス出力が可能なリアルタイム・パルス・ユニット ★ ・ECC回路内蔵（μPD78P324）			・インバータ制御に最適な6相PWM出力を容易に出力可能なリアルタイム・パルス・ユニット			・多彩なプログラマブル・パルス出力が可能なリアルタイム・パルス・ユニット ・ECC回路内蔵（μPD78P334）		
68ピンQFJ 74ピンQFP（□20 mm）	同左 68ピンWQFN 74ピンWQFN		64ピンSDIP 64ピンQFP（14×20 mm）	同左 64ピン窓付きSDIP		84ピンQFJ 94ピンQFP（□20 mm）	同左 84ピンWQFN 94ピンWQFN	

保守／廃止

表1－1　78K/IIIシリーズ製品概要（3/5）

	サブシリーズ名	μPD78352A				μPD78356		
★	品名	μPD78350	μPD78350A	μPD78352A	μPD78P352	μPD78355	μPD78356	μPD78P356
	命令数	113種				115種		
★	最小命令実行時間	160 ns/25 MHz	125 ns/32 MHz					
	内蔵 ROM容量	——		32 Kバイト	32 Kバイト（PROM）	——	48 Kバイト	48 Kバイト（PROM）
	内蔵 RAM容量	640バイト				2 Kバイト		
	外部要因	5				6		
	内部要因	4				25（外部兼用5）		
	割り込み機能	4レベル・プログラマブル・プライオリティ ベクタ割り込み機能 マクロ・サービス（高速データ転送／演算）機能（5種） コンテキスト・スイッチング機能						
	I/O端子 入力	6本				9本		
	I/O端子 入出力	24本		44本		48本	67本	
	A/Dコンバータ	——				10ビット分解能8チャネル		
	D/Aコンバータ	——				8ビット分解能2チャネル		
	タイマ／カウンタ	16ビット・フリー・ランニング・タイマ×1 16ビット・タイマ／イベント・カウンタ×1 16ビット・インターバル・タイマ×1				16ビット・タイマ／イベント・カウンタ×2 16ビット・インターバル・タイマ×2 16ビット・アップ／ダウン・カウンタ×1 10ビット・インターバル・タイマ×1		
	シリアル・インタフェース	——				UART（専用ボー・レート・ジェネレータ内蔵）…1 ch クロック同期式（SBI対応）…1 ch クロック同期式（端子切り替え機能付き）…1 ch		
	スタンバイ機能	HALT/STOPモード						
	特徴	・高速積和演算機能 ・ASICとの組み合わせにより高速な制御システムを実現				・高速，高性能16ビットCPU ・超高速A/D，高速D/Aコンバータ内蔵 ・ECC回路内蔵（μPD78P356）		
	パッケージ	64ピンQFP（□14 mm）	64ピンQFP（薄型）（□14 mm）		64ピンQFP（薄型）（□14 mm） 64ピンWQFN注	100ピンQFP（□14 mm） 120ピンQFP（□28 mm）		同左 120ピンWQFN

注　開発中

保守／廃止

★ 表1－1　78K/IIIシリーズ製品概要（4/5）

μPD78366A							
μPD78361A[注]	μPD78362A	μPD78P364A	μPD78363A	μPD78365A	μPD78366A	μPD78368A[注]	μPD78P368A
115種							
125 ns/8 MHz							
32 Kバイト	24 Kバイト	48 Kバイト（PROM）	24 Kバイト	——	32 Kバイト	48 Kバイト	48 Kバイト（PROM）
2 Kバイト	768バイト	2 Kバイト	768バイト	2 Kバイト			
6							
14（外部兼用2）							
4レベル・プログラマブル・プライオリティ ベクタ割り込み機能 マクロ・サービス（高速データ転送／演算）機能（5種） コンテキスト・スイッチング機能							
14本							
38本			49本	31本	49本		
10ビット分解能8チャネル							
——							
16ビット・フリー・ランニング・タイマ×2 16ビット・インターバル・タイマ／イベント・カウンタ×1 16ビット・インターバル・タイマ×1 16ビット・タイマ／アップ／ダウン・カウンタ×1							
UART（専用ボー・レート・ジェネレータ内蔵）…1 ch クロック同期式（SBI対応）…1 ch			UART（専用ボー・レート・ジェネレータ内蔵，端子切り替え機能付き）…1 ch クロック同期式（SBI対応）…1 ch				
HALT/STOPモード							
・高速，高性能16ビットCPU ・インバータ制御に最適な6チャネルPWMパルス出力機能内蔵 ・PLL制御回路内蔵							
64ピンSDIP			80ピンQFP（14×20 mm）				同左 80ピンWQFN

注　開発中

保守／廃止

★

表1－1　78K/IIIシリーズ製品概要（5/5）

サブシリーズ名		μPD78372	
品名		μPD78372(A)	μPD78P372(A)
命令数		115種	
最小命令実行時間		160 ns/25 MHz	
内蔵	ROM容量	24 Kバイト	24 Kバイト（PROM）
	RAM容量	768バイト	
割り込み機能	外部要因	11	
	内部要因	18（外部兼用6）	
		4レベル・プログラマブル・プライオリティ ベクタ割り込み機能 マクロ・サービス（高速データ転送／演算）機能（5種） コンテキスト・スイッチング機能	
I/O端子	入力	17本	
	入出力	43本	
A/Dコンバータ		10ビット分解能16チャネル	
タイマ／カウンタ		18/16ビット・フリー・ランニング・タイマ×1 16ビット・タイマ／イベント・カウンタ×1	
シリアル・インタフェース		UART(専用ボー・レート・ジェネレータ内蔵)…1 ch クロック同期式…1 ch	
スタンバイ機能		HALT/STOPモード，スタンバイ機能無効モード	
特徴		・高速，高性能16ビットCPU ・多彩なプログラマブル・パルス出力が可能なリアルタイム・パルス・ユニット ・ECC回路内蔵（μPD78P372(A)）	
パッケージ		80ピンQFP（14×20 mm） 80ピンQFP（□14 mm）	

第２章　μPD78312Aとの相違点

この章は，μPD78320の命令について，特にμPD78312Aとの違いに関して説明します。μPD78312Aで作成したソース・プログラムをμPD78320に変更する場合には，特にこの章の内容にご注意ください。

2.1　プログラム・ステータス・ワード（PSW）

図２－１はPSWの内部割り付けの違いです。

μPD78320では，ユーザ・フラグ（UF）をビット15に，割り込み要求許可フラグ（IE）をビット３に移動しています。また，減算フラグ(SUB)を削除し，新たに，割り込み制御用の割り込み優先順位レベル・トランジション・フラグ（LT）を設定しています。

図２－１　PSWの相違点

▒ はμPD78320では削除したフラグです。

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
μPD78312A	0	2	RBS 1	0	0	0	IE	0	S	Z	RSS	AC	UF	P/V	SUB	CY
μPD78320	UF	2	RBS 1	0	0	0	0	0	S	Z	RSS	AC	IE	P/V	LT	CY

2.2　命令の相違点

μPD78320の命令セットは，基本的にμPD78312Aに対しアップワード・コンパチブルです。

表2－1の11種類の命令はμPD78312Aから追加されたものです。表2－2の2種類の命令は一部異なります。

表2－1　μPD78320で追加された命令（1/2）

μPD78320で追加された命令		ニモニック	オペランド	オペレーション	フラグ				
					S	Z	AC	P/V	CY
1	メモリとレジスタ間の16ビット転送命令	MOVW	AX，mem	AX←(mem)					
			mem，AX	(mem)←AX					
		XCHW	AX，mem	AX↔(mem)					
2	符号拡張命令	CVTBW		A_7＝0のとき X←A，A←00H A_7＝1のとき X←A，A←FFH					
3	ソフトウエア割り込みからのリターン命令	RETB		PC_L←(SP)， PC_H←(SP＋1)， PSW_L←(SP＋2)， PSW_H←(SP＋3)， SP←SP＋4	R	R	R	R	R
4	ソフトウエア・コンテキスト・スイッチングからのリターン命令	RETCSB	!addr16	PC_H←R5，PC_L←R4， R5←$addr16_H$， R4←$addr16_L$， PSW_H←R7，PSW_L←R6	R	R	R	R	R
5	sfrpのプッシュ／ポップ命令	PUSH	sfrp	(SP－1)←sfr_H， (SP－2)←sfr_L， SP←SP－2					
		POP	sfrp	sfr_L←(SP)， sfr_H←(SP＋1)， SP←SP＋2					
6	ポート・テスト命令	CHKL	sfr	(端子レベル)∀(出力バッファの前段の信号レベル)	×	×		P	
		CHKLA	sfr	A←{(端子レベル)∀(出力バッファの前段の信号レベル)}	×	×		P	
7	符号付き乗算命令	MULW	rp1	AX(上位16ビット)， rp1(下位16ビット)← AX×rp1					

表2－1　μPD78320で追加された命令（2/2）

μPD78320で追加された命令		ニモニック	オペランド	オペレーション	フラグ				
					S	Z	AC	P/V	CY
8	積和演算命令[注1]	MACW[注1]	n	AXDE←(B)×(C)＋AXDE, B←B＋2, C←C＋2, n←n－1, End if n＝0 or P/V＝1	×	×	×	V	×
9	飽和付き積和演算命令[注2]	MACSW[注2]	n	AXDE←(B)×(C)＋AXDE, B←B＋2, C←C＋2, n←n－1, if overflow(P/V＝1) then AXDE←7FFFFFFFH, if underflow (P/V＝1) then AXDE←80000000H, End if n＝0 or P/V＝1	×	×	×	V	×
10	相関演算命令[注2]	SACW[注2]	[DE＋], [HL＋]	AX←AX＋｜(DE)－(HL)｜, DE←DE＋2, HL←HL＋2, C←C－1, End if C＝0 or CY＝1	×	×	×	V	×
11	テーブル・シフト命令[注1]	MOVTBLW[注1]	! addr16, n	(addr16＋2)←(addr16), n←n－1, addr16←addr16－2, End if n＝0					

注1．μPD78352Aサブシリーズ，78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズのみ。

2．μPD78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズのみ。

表２－２　一部が異なる命令

一部が異なる命令		ニモニック	オペランド	オペレーション	フラグ				
					S	Z	AC	P/V	CY
1	BCD補正命令	ADJBA		Decimal Adjust Accumulator	×	×	×	P	×
		ADJBS							
2	スタンバイ制御命令	MOV	STBC, ＃byte	STBC←byte[注]					
	ウォッチドッグ・タイマ制御命令		WDM, ＃byte	WDM←byte[注]					

注　STBC，WDMへのライト・アクセスは，４バイトの専用命令となります。これらの命令のオブジェクト・コードが異常のときには，次のオペコード・トラップ割り込みを発生します。

トラップ時のオペレーション：

$(SP-1)\leftarrow PSW_H$，$(SP-2)\leftarrow PSW_L$，

$(SP-3)\leftarrow (PC-4)_H$，$(SP-4)\leftarrow (PC-4)_L$，

$PC_L\leftarrow (003CH)$，$PC_H\leftarrow (003DH)$，

$SP\leftarrow SP-4$，$IE\leftarrow 0$

フラグ欄の記号

記　号	意　　味
	変化なし
P	P/Vフラグが，パリティ・フラグとして動作する
V	P/Vフラグが，オーバフロー・フラグとして動作する
R	以前に退避した値がリストアされる
×	結果に従ってセット／クリアされる

μPD78312Aにはない命令，および一部が異なる命令の機能の詳細は，それぞれ次のとおりです。

(1) メモリとレジスタ間の16ビット転送命令

MOVW AX, mem

機　　能：AX← (mem)　　mem＝0000H-FDFFH（任意のアドレスが可能）
mem＝FE00H-FFFFH（偶数アドレスに限る）

レジスタ・ペアAXに，第2オペランドに記述したメモリ・アドレシングによりアドレスされるメモリの内容を転送します。

memとして，オートインクリメント（[DE+]，[HL+]）/オートデクリメント（[DE−]，[HL−]）を指定した場合は，データの転送後にレジスタ・ペアDEまたはレジスタ・ペアHLの内容を自動的に＋2または−2します。

フラグの動作：変化なし

MOVW mem, AX

機　　能：(mem) ←AX　　mem＝0000H-FDFFH（任意のアドレスが可能）
mem＝FE00H-FFFFH（偶数アドレスに限る）

第1オペランドに記述したメモリ・アドレシングによりアドレスされるメモリに，レジスタ・ペアAXの内容を転送します。

memとして，オートインクリメント（[DE+]，[HL+]）/オートデクリメント（[DE−]，[HL−]）を指定した場合は，データの転送後にレジスタ・ペアDEまたはレジスタ・ペアHLの内容を自動的に＋2または−2します。

フラグの動作：変化なし

XCHW AX, mem

機　　能：AX↔ (mem)　　mem＝0000H-FDFFH（任意のアドレスが可能）
mem＝FE00H-FFFFH（偶数アドレスに限る）

レジスタ・ペアAXの内容と第2オペランドに記述したメモリ・アドレシングによりアドレスされるメモリの2バイトのエリアの内容を交換します。

memとして，オートインクリメント（[DE+]，[HL+]）/オートデクリメント（[DE−]，[HL−]）を指定した場合は，データの交換後にレジスタ・ペアDEまたはレジスタ・ペアHLの内容を自動的に＋2または−2します。

フラグの動作：変化なし

(2) 符号拡張命令

CVTBW

機　　能：・A_7＝0のとき

X←A，A←00H

・A_7＝1のとき

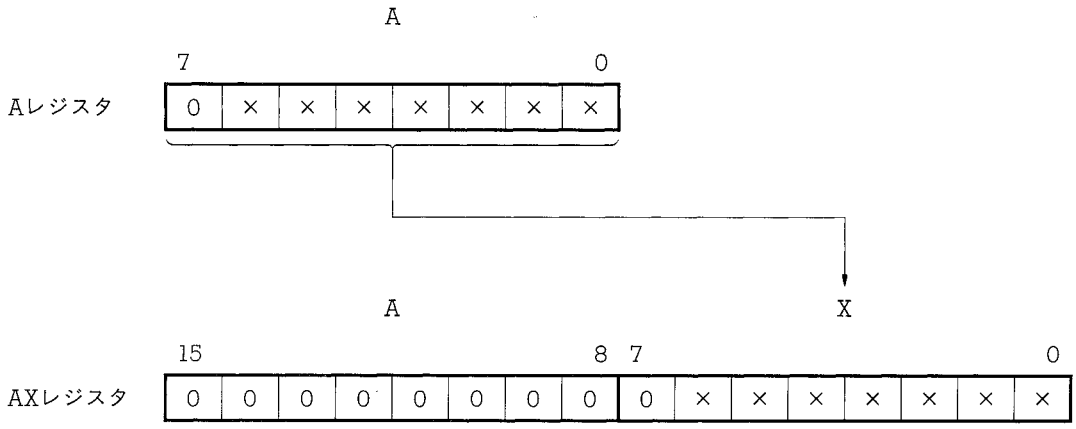
X←A，A←FFH

Aレジスタ内の符号付き8ビット・データをAXレジスタ内の符号付き16ビット・データに拡張します（**図２－２**　参照）。

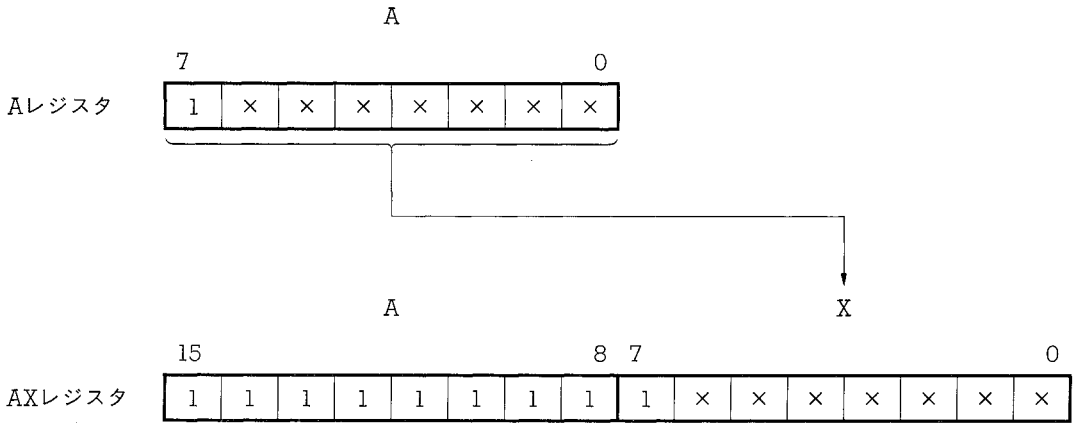
フラグの動作：変化なし

図２－２　CVTBW命令によるデータ変換

(a) A_7＝0のとき



(b) A_7＝1のとき



(3) ソフトウエア割り込みからのリターン命令

RETB

機　　能：PC_L ← (SP)，PC_H ← (SP+1)，
PSW_L ← (SP+2)，PSW_H ← (SP+3)，SP←SP+4
スタック・ポインタ (SP) でアドレスされるメモリ (スタック) の内容をプログラム・カウンタおよびプログラム・ステータス・ワード (PSW) にリストアします。その後，SPの内容をインクリメントします。
BRK命令，オペコード・トラップからの復帰時に使用します。

フラグの動作：

S	Z	AC	P/V	CY
R	R	R	R	R

注意　**BRK命令，およびオペコード・トラップに伴う割り込み処理ルーチンからの復帰には，必ずRETB命令を使用してください。RETI命令を使用すると，割り込み制御回路が正常動作しません。**

(4) ソフトウエア・コンテキスト・スイッチングからのリターン命令

RETCSB !addr16

機　　能：PC_H←R5，PC_L←R4，R5←addr16$_H$，R4←addr16$_L$，PSW_H←R7，PSW_L←R6
addr16＝0000H－FDFFH
この命令の実行時に指定されているレジスタ・バンク内の8ビット・レジスタ (R7，R6，R5，R4) の内容をそれぞれプログラム・ステータス・ワード (PSW)，プログラム・カウンタ (PC) に転送します。その後，R5，R4にセットしたアドレスに復帰します。
BRKCS命令からの復帰時に使用します。

フラグの動作：

S	Z	AC	P/V	CY
R	R	R	R	R

注意　**BRKCS命令による割り込み処理ルーチンからの復帰には，必ずRETCSB命令を使用してください。RETCS命令を用いると，割り込み制御回路が正常動作しません。**

(5) sfrpのプッシュ／ポップ命令

PUSH sfrp

機　　能：(SP－1) ←sfr_H，(SP－2) ←sfr_L，SP←SP－2

特殊機能レジスタ（16ビット操作可能レジスタ）の内容をスタック・ポインタ（SP）でアドレスされるメモリ（スタック）に退避します。その後，SPをデクリメントします。

フラグの動作：変化なし

POP sfrp

機　　能：sfr_L ←SP，sfr_H ← (SP＋1)，SP←SP＋2

スタック・ポインタ（SP）でアドレスされるメモリ（スタック）の内容を特殊機能レジスタ（16ビット操作可能レジスタ）にリストアします。その後，SPをインクリメントします。

フラグの動作：変化なし

(6) ポート・テスト命令

CHKL sfr

機　　能：(端子レベル) ⊻ (出力バッファの前段の信号レベル)

端子レベルと出力バッファの前段の信号レベルとの排他的論理和をとり，結果をフラグ（S，Z）にセットします。

フラグの動作：

S	Z	AC	P/V	CY
×	×		P	

CHKLA sfr

機　　能：A←{(端子レベル) ⊻ (出力バッファの前段の信号レベル)}

端子レベルと出力バッファの前段の信号レベルとの排他的論理和をとり，結果をAレジスタにセットします。

フラグの動作：

S	Z	AC	P/V	CY
×	×		P	

(7) 符号付き乗算命令

MULW rp1

機　　能：AX, rp1←AX×rp1 (符号付き)

レジスタ・ペアAXの内容とオペランドで指定される16ビット・レジスタ・ペアの内容との符号付き乗算を行い，結果の上位16ビットをレジスタ・ペアAXに，下位16ビットをオペランドで指定された16ビット・レジスタ・ペアにセットします。

フラグの動作：変化なし

(8) 積和演算命令

MACW n

機　　能：AXDE←(B)×(C)+AXDE, B←B+2, C←C+2, n←n−1

end if n=0 or P/V=1

Bレジスタでアドレスされる2バイトのエリアの内容と，Cレジスタでアドレスされる2バイトのエリアの内容との符号付き乗算を行い，その結果とレジスタ・ペアAXDEの内容を2進加算します。加算の結果をレジスタ・ペアAXDEにセットします。

その後，BレジスタとCレジスタの内容を+2します。

以上の動作をオペランドに記述された8ビット・イミーディエト・データの回数分繰り返します。

加算処理中にオーバフローが発生した場合，オーバフロー・フラグがセットされ，AXDEレジスタの値は不定になります。また，Bレジスタ，Cレジスタはオーバフローする直前の値を保持しています。

フラグの動作：

S	Z	AC	P/V	CY
×	×	×	V	×

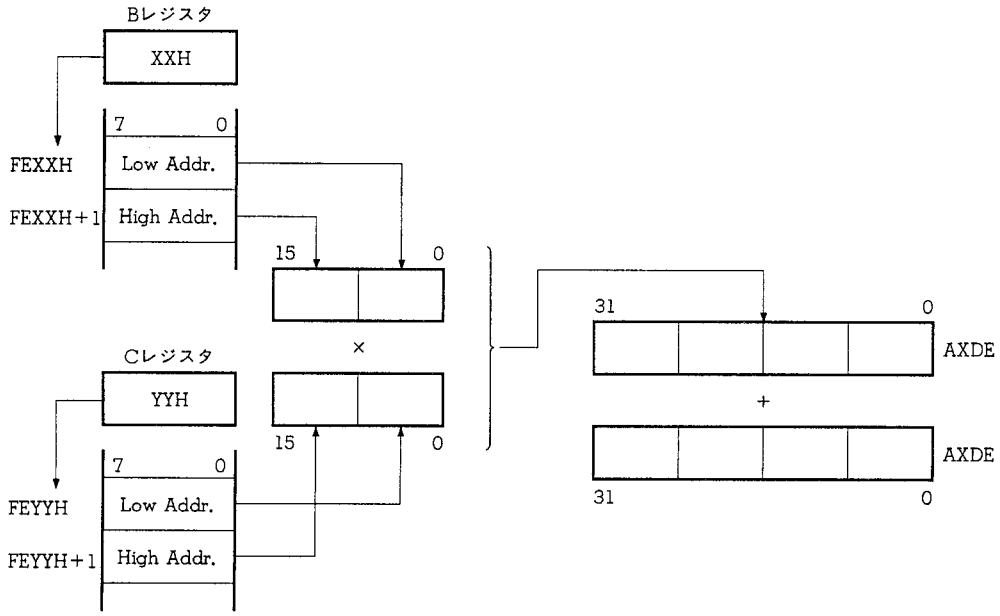
注意1.　MACW命令は，μPD78352Aサブシリーズ，78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズにのみ追加されています。

2.　MACW命令でアドレスされる領域はFE00H-FEFFH番地に限られ，Bレジスタおよび Cレジスタではアドレスの下位1バイトを指定します。また，FE80H-FEFFH番地は汎用レジスタと兼用しています。

MACW命令実行中は割り込み，マクロ・サービスを受け付けません。

MACW命令実行開始時にはAXDEレジスタ・ペアの値をクリアしませんので，必要な場合はプログラムでクリアするようにしてください。

図２－３　MACW命令によるデータの流れ



〔符号付き整数と積和演算命令実行時のオーバフロー〕

(a) 符号付き整数

積和演算命令で扱えるデータは，16ビット長の符号付き整数です。16ビットのデータ範囲とデータ記述例，および演算の結果AXDEレジスタにセットするデータの範囲を次に示します。

16ビット・データ範囲：－32768（10進）≦X≦32767（10進）
：　8000H≦X≦7FFFH

16ビット・データ記述例：15（10進）＝000FH
：－1（10進）＝FFFFH

AXDEレジスタ範囲：－2147483648（10進）≦X≦2147483647（10進）
：　80000000H≦X≦7FFFFFFFH

(b) オーバフロー

演算結果が正または負の最大値を越えると，オーバフローが発生します。

(9) 飽和付き積和演算命令

MACSW n

機　　能：AXDE←(B)×(C)+AXDE, B←B+2, C←C+2, n←n−1
if n=0 then end
if P/V=1 then AXDE←7FFFFFFFH(overflow), end
or AXDE←80000000H(underflow), end
overflow　：加算結果の符号が '+' から '−' に変化
underflow：加算結果の符号が '−' から '+' に変化
Bレジスタでアドレスされる2バイトのエリアの内容と，Cレジスタでアドレスされる2バイトのエリアの内容との符号付き乗算を行い，その結果とレジスタ・ペアAXDEの内容を2進加算します。加算の結果をレジスタ・ペアAXDEにセットします。
その後，BレジスタとCレジスタの内容を+2します。
以上の動作をオペランドに記述された8ビット・イミーディエト・データの回数分繰り返します。
加算処理中にオーバフローかアンダフローが発生した場合オーバフロー・フラグがセットされ，演算を終了します。
オーバフローが発生した場合はレジスタ・ペアAXDEに正の最大値（7FFFFFFFH）が格納されます。
アンダフローが発生した場合はレジスタ・ペアAXDEに負の最大値（80000000H）が格納されます。
Bレジスタ，Cレジスタはオーバフローまたは，アンダフローする直前の値を保持しています。

フラグの動作：

S	Z	AC	P/V	CY
×	×	×	V	×

注意1. **MACSW命令は，μPD78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズにのみ追加されています。**

2. **MACSW命令でアドレスされる領域はFE00H-FEFFH番地に限られ，Bレジスタおよび Cレジスタではアドレスの下位1バイトを指定します。また，FE80H-FEFFH番地は汎用レジスタと兼用しています。**
MACSW命令実行中は割り込み，マクロ・サービスを受け付けません。
MACSW命令実行開始時にはレジスタ・ペアAXDEの値をクリアしませんので，必要な場合はプログラムでクリアするようにしてください。

保守／廃止

(10) 相関演算命令

SACW [DE+], [HL+]

機　　能：AX←AX＋｜(DE)－(HL)｜，DE←DE+2，HL←HL+2，C←C－1
if C=0 or CY=1 then end
レジスタ・ペアDEでアドレスされるメモリの内容から，レジスタ・ペアHLでアドレスされるメモリの内容を減算します。減算結果の絶対値を取り，結果をレジスタ・ペアAXに加算します。
その後，レジスタ・ペアDEとレジスタ・ペアHLの内容を＋2し，Cレジスタの内容をデクリメントします。
以上の動作をCレジスタの内容が０になるまで繰り返します。

フラグの動作：

S	Z	AC	P/V	CY
×	×	×	V	×

注意　SACW命令は，μPD78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズにのみ追加されています。

保守／廃止

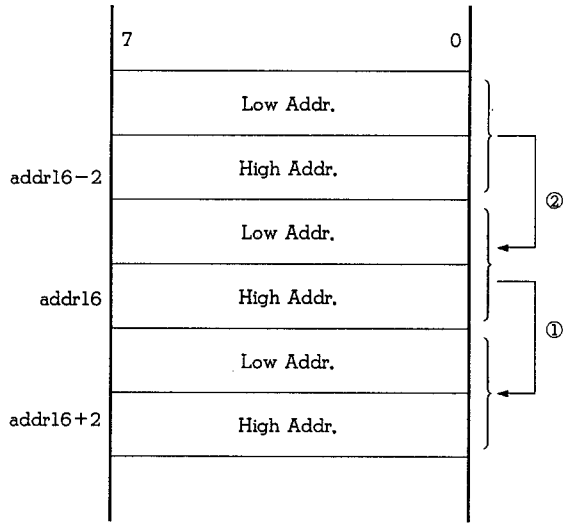
(11) テーブル・シフト命令

MOVTBLW　！addr16, n

機　　能：(addr16＋2)←(addr16), addr16←addr16－2, n←n－1
end if n＝0
第１オペランドで指定される16ビット・イミーディエト・データでアドレスされるメモリの内容を，＋2したアドレスに転送します。その後，addr16を－2します。
以上の動作を第２オペランドに記述された８ビット・イミーディエト・データの回数分繰り返し，MACW命令のデータ・テーブルをシフトします。
第１オペランドの！addr16には，データ・テーブルの最下位のアドレスをレーベルまたは数値で直接記述します。

フラグの動作：変化なし

図２－４　MOVTBLW命令によるデータの流れ



注意1. MOVTBLW命令は，μPD78352Aサブシリーズ，78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズにのみ追加されています。

2. MOVTBLW命令でアドレスされる領域はFE00H-FEFFH番地に限られ，第１オペランドの！addr16でアドレスの下位１バイトを指定します。
MOVTBLW命令実行中は割り込み，マクロ・サービスを受け付けません。

(12) BCD補正命令

μPD78320では，BCD補正命令ADJ4とPSW中のSUBフラグがなく，BCD補正命令が加算用のADJBAと減算用のADJBSとで別々になっています。

ADJBA

機　　能：Aレジスタ，キャリー・フラグ（CY），補助キャリー・フラグ（AC）の内容を判定して，表２-３のように10進補正します。この命令は，10進（BCD）データ同士の加算を実行したのちにのみ意味を持ちます。

表２-３　10進データの補正（ADJBA命令）

条　　件		オペレーション	補正後	
			CY	AC
$A_{3-0} \leqq 9$ AC=0	$A_{7-4} \leqq 9$ and CY=0	A←A	0	0
	$A_{7-4} \geqq 10$ or CY=1	A←A+60H	1	0
$A_{3-0} \geqq 10$ AC=0	$A_{7-4} < 9$ and CY=0	A←A+06H	0	1
	$A_{7-4} \geqq 9$ or CY=1	A←A+66H	1	1
AC=1	$A_{7-4} \leqq 9$ and CY=0	A←A+06H	0	1
	$A_{7-4} \geqq 10$ or CY=1	A←A+66H	1	1

フラグの動作：

S	Z	AC	P/V	CY
×	×	×	P	×

保守／廃止

ADJBS

機　　能：Aレジスタ，キャリー・フラグ（CY），補助キャリー・フラグ（AC）の内容を判定して，表２-４のように10進補正します。この命令は，10進（BCD）データ同士の減算を実行したのちにのみ意味を持ちます。

表２-４　10進データの補正（ADJBS命令）

条　　件		オペレーション	補正後	
			CY	AC
$A_{3-0} \leqq 9$ AC=0	$A_{7-4} \leqq 9$ and CY=0	A←A	0	0
	$A_{7-4} \geqq 10$ or CY=1	A←A−60H	1	0
$A_{3-0} \geqq 10$ AC=0	$A_{7-4} < 9$ and CY=0	A←A−06H	0	1
	$A_{7-4} \geqq 9$ or CY=1	A←A−66H	1	1
AC=1	$A_{7-4} \leqq 9$ and CY=0	A←A−06H	0	1
	$A_{7-4} \geqq 10$ or CY=1	A←A−66H	1	1

フラグの動作：

S	Z	AC	P/V	CY
×	×	×	P	×

(13) スタンバイ制御命令，ウォッチドッグ・タイマ制御命令

μPD78320では２バイト目の命令コードが異なります。表２-５が相違点です。

表２-５　命令コードの相違点

　　の部分が互いに異なります。

品　名	ニモニック	オペランド	命令コード			
			B1	B2	B3	B4
μPD78320	MOV	STBC, #byte	0000　1001	1100　0000	$\overline{\text{data}}$	data
		WDM, #byte	0000　1001	1100　0010	$\overline{\text{data}}$	data
μPD78312A	MOV	STBC, #byte	0000　1001	0100　0100	$\overline{\text{data}}$	data
		WDM, #byte	0000　1001	0100　0010	$\overline{\text{data}}$	data

また，μPD78320は命令コードの３バイト目と４バイト目が互いに補数でないときには，STBC, WDMに対して書き込みは行わずに，オペコード・トラップ割り込みを発生します。

この場合，スタック領域に退避されるリターン・アドレスは，トラップの原因となった命令のアドレスです。したがって，RETB命令により，トラップの原因となった命令のアドレスから再びプログラムを実行できます。

ただし，ハードウエア・エラーの場合など，オペコード・トラップの発生原因が取り除けていない状態では，RETB命令により無限ループとなります。

保守／廃止

2.3　μPD78312Aで作成したプログラムをμPD78320に移植する際の注意

(1) BCD補正

μPD78312Aは，SUBフラグを内蔵しており，ADJ4命令を実行するとSUBフラグのレベルにより自動的にADDかSUB演算を選択して実行することができましたが，μPD78320では，ADD演算後のBCD補正とSUB演算後のBCD補正は別命令となっており，ソース・プログラムの変更が必要です。

例 1.

μPD78320			μPD78312A	
MOV	A, saddr		MOV	A, saddr
ADD	A, B	←	ADD	A, B
ADJBA			ADJ4	
⋮			⋮	

2.

μPD78320			μPD78312A	
MOV	A, saddr		MOV	A, saddr
SUBC	A, B	←	SUBC	A, B
ADJBS			ADJ4	
⋮			⋮	

(2) BRK命令，BRKCS命令からのリターン

μPD78312Aでは，割り込み制御回路の正常動作を保証するために，CCW.0(EOSフラグ)をセット(1)してからリターン命令を実行していますが，μPD78320では，それぞれ専用命令を設定していますので，ソース・プログラムの変更が必要です。

例 1.

μPD78320			μPD78312A		
SET1	CY		SET1	CY	
RETB		←	SET1	CCW.0	; EOS←1
			RETI		
⋮		2命令を1命令に置き換え	⋮		

例 2.

μPD78320		μPD78312A	
CLR1　CY		CLR1　CY	
RETCSB　!addr16	←	SET1　CCW.0	;EOS←1
⋮	2命令を1命令に置き換え	RETCS　!addr16	
		⋮	

(3) 外部SFRを利用したプログラム

μPD78312Aの外部SFR（FFB0H-FFBFH）とμPD78320の外部SFR（FFD0H-FFDFH）はアドレスが異なるため，ソース・プログラムの変更が必要です。

(4) PSW操作（主にUFフラグ）

μPD78320ではPSWの構成を変更していますので，PSWのビット操作命令を使用しているときは，ビット位置を変更する必要があります。

IEフラグについては，EI，DI命令を用いた場合は，ソース・プログラムの見直しは必要ありませんが，UFフラグについては，必ずソース・プログラムを変更する必要があります。

例 1.

μPD78320		μPD78312A	
MOV　MK0，#5AH		MOV　MK0，#5AH	
SET1　PSWL.3	←	SET1　PSWH.1	;IE←1
⋮		⋮	

2.

μPD78320		μPD78312A	
MOV　A，Rn		MOV　A，Rn	
SET1　PSWH.7	←	SET1　PSWL.3	;UF←1
⋮		⋮	

第3章　ソフトウエア

3.1　符号付き２進演算

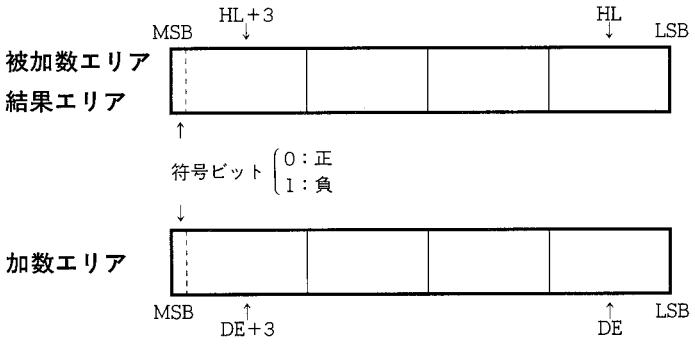
ここでは，符号付き２進演算について，加減乗除算を紹介します。

最上位ビットを符号ビットとし，残りのビットで数値を表します。負の数は，２の補数表現で表します。

3.1.1　２進加算

32ビット←32ビット＋32ビット

(1) 使用メモリ・エリア



(2) 使用レジスタ

A，C，D，E，H，L

(3) 入力条件

(1)のように，加数，被加数のメモリ・アドレスは，それぞれ次のレジスタ・ペアで指定します。

HLレジスタ・ペア←被加数32ビットが格納してあるエリアの最下位アドレス

DEレジスタ・ペア←加数32ビットが格納してあるエリアの最下位アドレス

(4) 出力条件

演算結果は，(1)の結果エリア（HL，HL＋1，HL＋2，HL＋3）に格納します。

ただし，P/V(パリティ／オーバフロー・フラグ)＝1のときは，演算結果がオーバフロー，またはアンダフローしています。

(5) 処理手順

① バイト・カウンタ（Cレジスタ）に4を設定する。

② キャリー・フラグをあらかじめクリアする。

③ 加数アドレスで指定する加数1バイトをアキュームレータにロードし，その後加数アドレスを（+1）する。

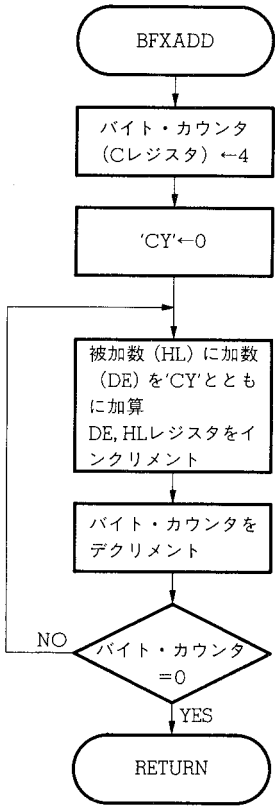
④ 被加数アドレスで指定する被加数1バイトをキャリー・フラグとともにアキュームレータに加え，その演算結果を被加数アドレスで指定するメモリに格納し，その後被加数アドレスを(+1）する。

⑤ バイト・カウンタを（−1）し，バイト・カウンタが0になるまで③から④の処理を繰り返す。

(6) ステップ数

9バイト

フロー・チャート



```
     Assemble list

 ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

    1     1                      $       TITLE   ('binary addition')
    2     2                              NAME    BFXADR
    3     3                      ;*************************************************
    4     4                      ;*      binary addition                          *
    5     5                      ;*           32 bit <- 32 bit + 32 bit           *
    6     6                      ;*       input condition                         *
    7     7                      ;*             HL-register <- augnend top.address *
    8     8                      ;*             DE-register <- addend  top.address *
    9     9                      ;*       output condition                        *
   10    10                      ;*             result <- (HL,HL+1,HL+2,HL+3)      *
   11    11                      ;*                                                *
   12    12                      ;*              RSS <- 0                          *
   13    13                      ;*************************************************
   14    14
   15    15                              PUBLIC  BFXADD
   16    16                      ;
   17    17        (0004)        BYTNUM  EQU     4
   18    18                      ;
   19    19 ----                         CSEG
   20    20                              RSS     0
   21    21                      ;
   22    22 0000                 BFXADD:
   23    23 0000  BA04                   MOV     C,#BYTNUM
   24    24 0002                 BFXAD1:
   25    25 0002  40                     CLR1    CY
   26    26 0003                 BFXAD2:
   27    27 0003  58                     MOV     A,[DE+]
   28    28 0004  1699                   ADDC    [HL+],A
   29    29 0006  32FB                   DBNZ    C,$BFXAD2
   30    30 0008  56                     RET
   31    31                      ;
   32    32                              ENDS
   33    33                              END

Segment informations:

ADRS  LEN    NAME

0000  0009H  ?CSEG
```

```
          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG       ?CSEG       19#
BFXAD1      2H   R ADDR       ?CSEG       24#
BFXAD2      3H   R ADDR       ?CSEG       26#     29
BFXADD      0H   R ADDR  PUB  ?CSEG       15@     22#
BFXADR             MOD                     2#
BYTNUM      4H     NUM                     17#     23

Target chip:uPD78320
Assembly complete,      0 error(s) and      0 warning(s) found. (     0)
```

3.1.2　2進減算

32ビット←32ビット－32ビット

(1) 使用メモリ・エリア

(2) 使用レジスタ

A, C, D, E, H, L

(3) 入力条件

(1)のように，減数，被減数のメモリ・アドレスは，それぞれ次のレジスタ・ペアで指定します。

HLレジスタ・ペア←被減数32ビットが格納してあるエリアの最下位アドレス

DEレジスタ・ペア←減数32ビットが格納してあるエリアの最下位アドレス

(4) 出力条件

演算結果は，(1)の結果エリア（HL，HL＋1，HL＋2，HL＋3）に格納します。

ただし，P/V＝1のときは，演算結果がオーバフロー，またはアンダフローしています。

(5) 処理手順

① バイト・カウンタ（Cレジスタ）に4を設定する。

② キャリー・フラグをあらかじめクリアする。

③ 減数アドレスで指定する減数1バイトをアキュームレータにロードし，その後減数アドレスを（＋1）する。

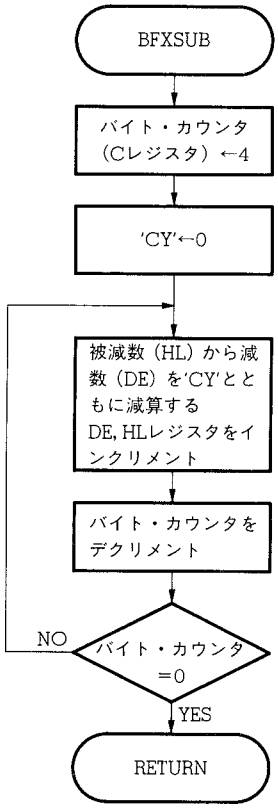
④ 被減数アドレスで指定する被減数1バイトからキャリー・フラグとともにアキュームレータの値を引き，その演算結果を被減数アドレスで指定するメモリに格納し，その後被減数アドレスを（＋1）する。

⑤ バイト・カウンタを（－1）し，バイト・カウンタが0になるまで③から④の処理を繰り返す。

(6) ステップ数

9バイト

フロー・チャート



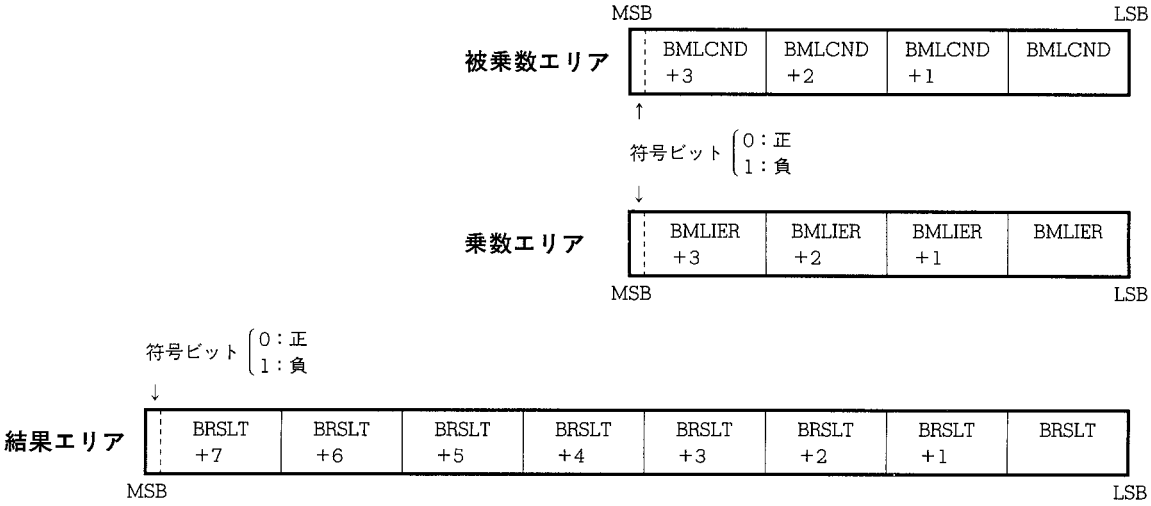
```
     Assemble list

 ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

    1     1                      $       TITLE   ('binary subtraction')
    2     2                              NAME    BFXSBR
    3     3                      ;*******************************************************
    4     4                      ;*      binary subtraction                             *
    5     5                      ;*           32 bit <- 32 bit - 32 bit                 *
    6     6                      ;*       input condition                               *
    7     7                      ;*             HL-register <- minuend    top.address  *
    8     8                      ;*             DE-register <- subtrahend top.address  *
    9     9                      ;*       output condition                              *
   10    10                      ;*             result <- (HL,HL+1,HL+2,HL+3)           *
   11    11                      ;*                                                     *
   12    12                      ;*              RSS <- 0                               *
   13    13                      ;*******************************************************
   14    14
   15    15                              PUBLIC  BFXSUB
   16    16                      ;
   17    17        (0004)        BYTNUM  EQU     4
   18    18                      ;
   19    19 ----                         CSEG
   20    20                              RSS     0
   21    21                      ;
   22    22 0000                 BFXSUB:
   23    23 0000  BA04                   MOV     C,#BYTNUM
   24    24 0002                 BFXSU1:
   25    25 0002  40                     CLR1    CY
   26    26 0003                 BFXSU2:
   27    27 0003  58                     MOV     A,[DE+]
   28    28 0004  169B                   SUBC    [HL+],A
   29    29 0006  32FB                   DBNZ    C,$BFXSU2
   30    30 0008  56                     RET
   31    31                      ;
   32    32                              ENDS
   33    33                              END

Segment informations:

ADRS  LEN    NAME

0000  0009H  ?CSEG
```

```
          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG             CSEG       ?CSEG       19#
BFXSBR            MOD                     2#
BFXSU1      2H    R ADDR       ?CSEG       24#
BFXSU2      3H    R ADDR       ?CSEG       26#    29
BFXSUB      0H    R ADDR  PUB  ?CSEG       15@    22#
BYTNUM      4H    NUM                     17#    23

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

3.1.3　2進乗算

64ビット←32ビット×32ビット

(1) 使用メモリ・エリア



(2) 使用するレジスタ

X, A, C, B, R4, R5, R8, R9, R10, R11, D, E, H, L

(3) 入力条件

被乗数32ビット，乗数32ビットをそれぞれ(1)項で次のメモリ・エリアに入れます。

被乗数（BMLCND，BMLCND＋1，BMLCND＋2，BMLCND＋3）

乗　数（BMLIER，BMLIER＋1，BMLIER＋2，BMLIER＋3）

(4) 出力条件

演算結果を結果エリア（BRSLT，BRSLT＋1，……，BRSLT＋7）に格納します。

(5) 処理手順

この演算プログラムでは，μPD78320特有の乗算命令を用いるため，乗数32ビットを上位，下位それぞれ16ビットに分けて被乗数32ビット×乗数16ビットを２回行います。

① 結果エリアを０クリアしておく。

② 乗数，被乗数の絶対値を取る。乗数，被乗数が異符号なら符号フラグ（ユーザ・フラグ）をセットし，同符号ならリセットする。

③ ⑤で行う演算の結果を結果エリアに格納しておくため，結果エリアの最下位番地をHLレジスタに設定し，スタックに退避する。

④ 乗数２バイトに（BMLIER，BMLIER＋1）を設定する。

⑤　被乗数４バイト×乗数２バイトに結果エリア（HL，HL＋1，……，HL＋5）を加え，演算結果を結果エリア（HL，HL＋1，……，HL＋5）へ格納する。

⑥　乗数が上位桁となるため④の結果を格納する。結果エリアの最下位番地をスタックよりHLレジスタに復帰させ，（＋2）する。

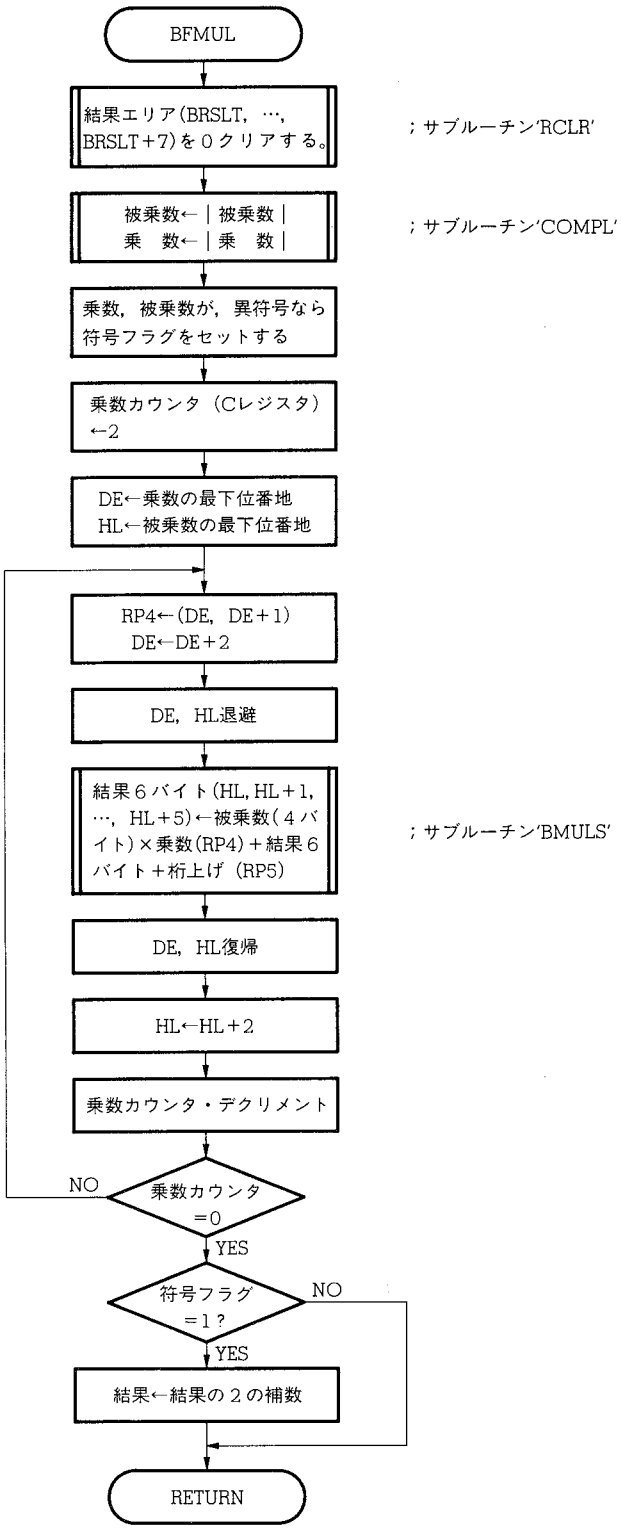
⑦　乗数２バイトを（BMLIER＋2，BMLIER＋3）に設定し，⑤から⑥の処理を行う。

⑧　符号フラグを調べセットされていれば結果の２の補数を取る。

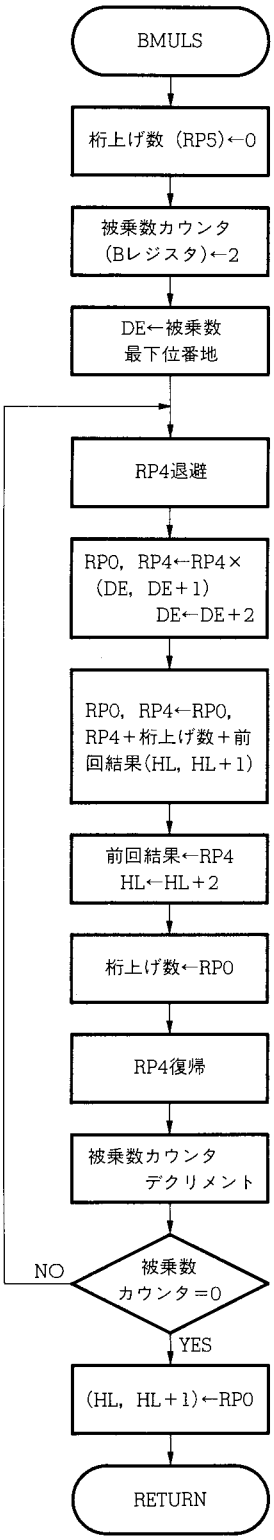
(6) ステップ数

151バイト（BFMULR：122バイト，CLR：29バイト）

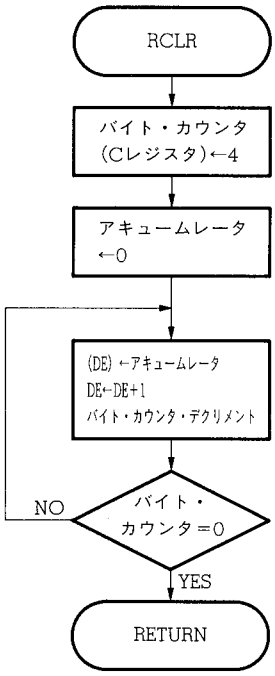
フロー・チャート



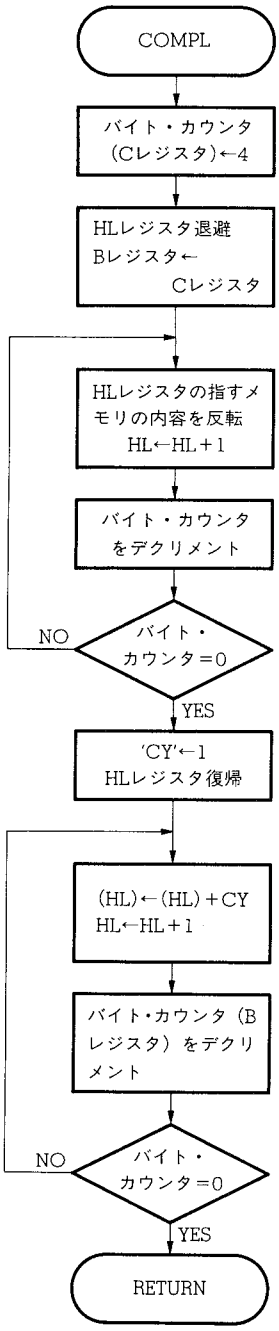
フロー・チャート



フロー・チャート



フロー・チャート



保守／廃止

Assemble list

```
ALNO  STNO ADRS  OBJECT    M I  SOURCE STATEMENT

   1     1                      $       TITLE   ('binary multiplication')
   2     2                              NAME    BFMULR
   3     3                      ;*******************************************************
   4     4                      ;*      binary multiplication                          *
   5     5                      ;*         input condition                             *
   6     6                      ;*              multiplicand <- (BMLCND+3,...,BMLCND)  *
   7     7                      ;*              multiplier   <- (BMLIER+3,...,BMLIER)  *
   8     8                      ;*         output condition                            *
   9     9                      ;*              result <- (BRSLT+7,BRSLT+6...,BRSLT)   *
  10    10                      ;*                                                     *
  11    11                      ;*              RSS <- 0                               *
  12    12                      ;*******************************************************
  13    13
  14    14                              PUBLIC  BFMUL
  15    15                              PUBLIC  BMULS
  16    16                              EXTRN   BMLCND,BMLIER,BRSLT
  17    17                              EXTRN   RCLR,RCLR1,COMPL,COMP1
  18    18                      ;
  19    19        (01FF.7)      SFLAG   EQU     PSWH.7
  20    20                      ;
  21    21 ----                         CSEG
  22    22                              RSS     0
  23    23 0000                 BFMUL:
  24    24                      ;
  25    25                      ;       ****  result area 0-clear  ****
  26    26                      ;
  27    27 0000  BA08                   MOV     C,#8            ;
  28    28 0002 R650000                 MOVW    DE,#BRSLT       ; DE-register <- BRSLT
  29    29 0005 R280000                 CALL    !RCLR1          ; clear subroutine
  30    30                      ;
  31    31                      ;       ****  complement convert  ****
  32    32                      ;
  33    33 0008  029F                   CLR1    SFLAG           ; sign-flag <- 0
  34    34 000A R670000                 MOVW    HL,#BMLCND      ; HL-register <- BMLCND
  35    35 000D  062003                 MOV     A,[HL+3]
  36    36 0010  03AF05                 BF      A.7,$BFMUL1
  37    37 0013 R280000                 CALL    !COMPL          ; complement subroutine
  38    38 0016  027F                   NOT1    SFLAG           ; not sign-flag
  39    39 0018                 BFMUL1:
  40    40 0018 R670000                 MOVW    HL,#BMLIER      ; HL-register <- BMLIER
  41    41 001B  062003                 MOV     A,[HL+3]
  42    42 001E  03AF05                 BF      A.7,$BFMUL2
  43    43 0021 R280000                 CALL    !COMPL          ; complement subroutine
  44    44 0024  027F                   NOT1    SFLAG           ; not sign-flag
```

```
45    45                    ;
46    46                    ;       ****  multiplicand counter set  ****
47    47                    ;
48    48 0026               BFMUL2:
49    49 0026  BA02                 MOV     C,#2            ; C-register <- 2
50    50 0028 R650000               MOVW    DE,#BMLIER      ; multiplier top.address
51    51 002B R670000               MOVW    HL,#BRSLT       ; result top.address
52    52                    ;
53    53 002E               BFMUL3:
54    54 002E  1601                 MOVW    AX,[DE+]        ; multiplier set
55    55 0030  2488                 MOVW    VP,AX
56    56                    ;
57    57 0032  35C0                 PUSH    DE,HL           ; SEVE DE,HL
58    58                    ;
59    59                    ;       ****  48 bit <- 32 bit * 16 bit  ****
60    60                    ;
61    61 0034 R284900               CALL    !BMULS
62    62                    ;
63    63 0037  34C0                 POP     DE,HL           ; LOAD HL,DE
64    64 0039  47                   INCW    HL              ; HL <- HL+2
65    65 003A  47                   INCW    HL
66    66                    ;
67    67                    ;       ****  check / multiply end ?  ****
68    68                    ;
69    69 003B  32F1                 DBNZ    C,$BFMUL3
70    70                    ;
71    71 003D  02AF08               BF      SFLAG,$BFMUL4   ; sign-flag = 1 ?
72    72 0040 R670000               MOVW    HL,#BRSLT
73    73 0043  BA08                 MOV     C,#8
74    74 0045 R280000               CALL    !COMP1          ; complement subroutine
75    75 0048               BFMUL4:
76    76 0048  56                   RET
77    77                            ENDS
78    78
79    79                    $       EJECT
```

```
 80    80
 81    81                    ;
 82    82                    ;************************************************
 83    83                    ;*      binary multiply subroutine              *
 84    84                    ;*           --- 48 bit <- 32 bit * 16 bit ---  *
 85    85                    ;************************************************
 86    86                    ;
 87    87 ----               ;       CSEG
 88    88                    ;       RSS     0
 89    89 0049               BMULS:
 90    90 0049  630000       ;       MOVW    UP,#0           ; carry <- 0
 91    91                    ;
 92    92                    ;       ****  multiplier counter set  ****
 93    93                    ;
 94    94 004C  BB02         ;       MOV     B,#2
 95    95                    ;
 96    96                    ;       ****  multiplicand set  ****
 97    97                    ;
 98    98 004E R650000       ;       MOVW    DE,#BMLCND
 99    99                    ;
100   100                    ;       ****  16 bit * 16 bit + carry + result  ****
101   101                    ;
102   102 0051               BMULS1:
103   103 0051  3510         ;       PUSH    VP
104   104 0053  1601         ;       MOVW    AX,[DE+]
105   105                    ;
106   106 0055  0529         ;       MULUW   VP
107   107                    ;
108   108                    ;       RSS     1
109   109 0057  43           ;       SWRS
110   110                    ;
111   111 0058  1651         ;       MOVW    AX,[HL]
112   112                    ;
113   113                    ;       RSS     0
114   114 005A  43           ;       SWRS
115   115                    ;
116   116 005B  8888         ;       ADDW    VP,AX
117   117 005D  8203         ;       BNC     $BMULS2
118   118 005F  2D0100       ;       ADDW    AX,#1           ; increment AX
119   119                    ;
120   120 0062               BMULS2:
121   121 0062  888B         ;       ADDW    VP,UP
122   122                    ;
123   123 0064  8203         ;       BNC     $BMULS3
124   124 0066  2D0100       ;       ADDW    AX,#1           ; increment AX
125   125                    ;
126   126 0069               BMULS3:
127   127                    ;       RSS     1
128   128 0069  43           ;       SWRS
129   129                    ;
130   130 006A  2549         ;       XCHW    AX,VP           ; AX <-> VP
131   131 006C  1691         ;       MOVW    [HL+],AX
```

```
  132   132                     ;
  133   133                             RSS     0
  134   134 006E  43                     SWRS                    ; RSS <- 0
  135   135                     ;
  136   136 006F  24A8                   MOVW    UP,AX
  137   137 0071  3410                   POP     VP
  138   138                     ;
  139   139 0073  33DC                   DBNZ    B,$BMULS1
  140   140                     ;
  141   141 0075  D8                     XCH     A,X
  142   142 0076  51                     MOV     [HL+],A
  143   143 0077  D8                     XCH     A,X
  144   144 0078  55                     MOV     [HL],A
  145   145 0079  56                     RET
  146   146
  147   147                             ENDS
  148   148                             END

Segment informations:

ADRS  LEN    NAME

0000  007AH  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG             CSEG        ?CSEG       21#    87#
BFMUL       0H    R ADDR  PUB  ?CSEG       14@    23#
BFMUL1     18H    R ADDR       ?CSEG       36     39#
BFMUL2     26H    R ADDR       ?CSEG       42     48#
BFMUL3     2EH    R ADDR       ?CSEG       53#    69
BFMUL4     48H    R ADDR       ?CSEG       71     75#
BFMULR            MOD                      2#
BMLCND    ----H   E       EXT              16@    34     98
BMLIER    ----H   E       EXT              16@    40     50
BMULS      49H    R ADDR  PUB  ?CSEG       15@    61     89#
BMULS1     51H    R ADDR       ?CSEG      102#   139
BMULS2     62H    R ADDR       ?CSEG      117    120#
BMULS3     69H    R ADDR       ?CSEG      123    126#
BRSLT     ----H   E       EXT              16@    28     51     72
COMP1     ----H   E       EXT              17@    74
COMPL     ----H   E       EXT              17@    37     43
RCLR      ----H   E       EXT              17@
RCLR1     ----H   E       EXT              17@    29
SFLAG      1FFH.7  RBIT                    19#    33     38     44     71

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (     0)
```

Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

  1     1                      $       TITLE   ('0-clear routine')
  2     2                              NAME    CLR
  3     3                      ;*****************************************************
  4     4                      ;*      0-clear process                              *
  5     5                      ;*          input condition                          *
  6     6                      ;*              DE-register <- 0-clear start adress  *
  7     7                      ;*                                                   *
  8     8                      ;*              RSS <- 0                             *
  9     9                      ;*****************************************************
 10    10
 11    11                              PUBLIC  RCLR,RCLR1,RCLR2
 12    12                              PUBLIC  COMPL,COMP1
 13    13                      ;
 14    14       (0004)         BYTNUM  EQU     4
 15    15                      ;
 16    16 ----                         CSEG
 17    17                              RSS     0
 18    18 0000                 RCLR:
 19    19 0000  BA04                   MOV     C,#4            ; C-register <- 4
 20    20 0002                 RCLR1:
 21    21 0002  B900                   MOV     A,#0            ; Acc <- 0
 22    22 0004                 RCLR2:
 23    23 0004  1500                   MOVM    [DE+],A
 24    24                      ;
 25    25 0006  56                     RET
 26    26                              ENDS
 27    27
 28    28                      ;*********************************************************
 29    29                      ;*      complement convert subroutine                    *
 30    30                      ;*              input  condition                         *
 31    31                      ;*                HL-register <- complement top.address  *
 32    32                      ;*              output condition                         *
 33    33                      ;*                (HL+3,HL+2,...,HL) <- convert data     *
 34    34                      ;*                                                       *
 35    35                      ;*                RSS <- 0                               *
 36    36                      ;*********************************************************
 37    37
 38    38 ----                         CSEG
 39    39                              RSS     0
 40    40 0007                 COMPL:
 41    41 0007  BA04                   MOV     C,#BYTNUM
 42    42 0009                 COMP1:
 43    43 0009  3580                   PUSH    HL              ; save HL-register
 44    44 000B  2432                   MOV     B,C             ; B-register <- C-register
 45    45 000D  B9FF                   MOV     A,#0FFH         ; Acc <- FFH
```

```
   46    46 000F                COMP2:
   47    47 000F  169D                  XOR     [HL+],A
   48    48 0011  32FC                  DBNZ    C,$COMP2
   49    49                     ;
   50    50 0013  B900                  MOV     A,#0            ; Acc <- 0
   51    51 0015  41                    SET1    CY
   52    52 0016  3480                  POP     HL              ; load HL-register
   53    53                     ;
   54    54 0018                COMP3:
   55    55 0018  1699                  ADDC    [HL+],A
   56    56 001A  33FC                  DBNZ    B,$COMP3
   57    57                     ;
   58    58 001C  56                    RET
   59    59                             ENDS
   60    60                             END
   61    61

Segment informations:

ADRS  LEN    NAME

0000  001DH  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG             CSEG       ?CSEG       16#    38#
BYTNUM      4H    NUM                    14#    41
CLR               MOD                     2#
COMP1       9H  R ADDR  PUB  ?CSEG       12@    42#
COMP2       FH  R ADDR       ?CSEG       46#    48
COMP3      18H  R ADDR       ?CSEG       54#    56
COMPL       7H  R ADDR  PUB  ?CSEG       12@    40#
RCLR        0H  R ADDR  PUB  ?CSEG       11@    18#
RCLR1       2H  R ADDR  PUB  ?CSEG       11@    20#
RCLR2       4H  R ADDR  PUB  ?CSEG       11@    22#

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

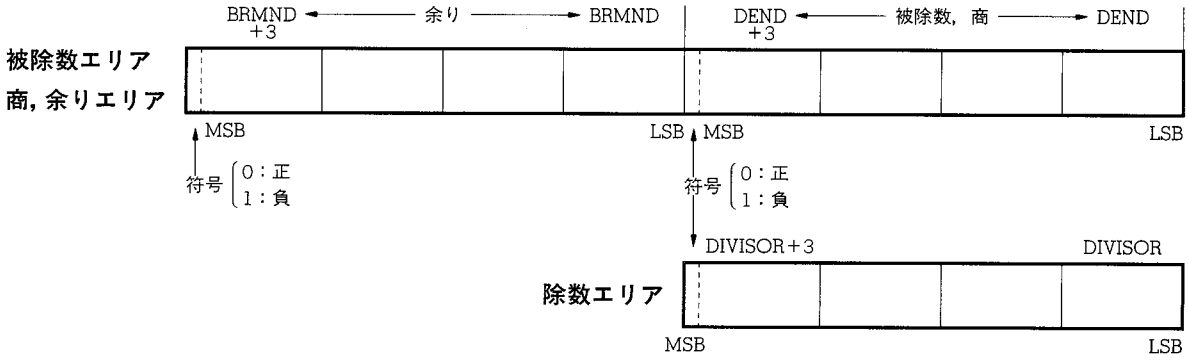
保守／廃止

3.1.4　2進除算

商　32ビット←32ビット÷32ビット

余り32ビット

(1) 使用メモリ・エリア



(2) 使用するレジスタ

X，A，C，B，D，E，H，L

(3) 入力条件

被除数32ビット，除数32ビットをそれぞれ(1)の次のメモリ・エリアに格納します。

被除数（DEND，DEND＋1，DEND＋2，DEND＋3）

除　数（DIVISOR，DIVISOR＋1，DIVISOR＋2，DIVISOR＋3）

(4) 出力条件

それぞれ(1)の次のメモリ・エリアに格納します。

商は，商エリア（DEND，DEND＋1，……，DEND＋3）

余りは，余りエリア（BRMND，BRMND＋1，……，BRMND＋3）

除数が０の場合は，エラー処理へ分岐します。

(5) 処理手順

この演算プログラムでは被除数(DEND，DEND＋1，……，DEND＋3)と余り(BRMND，BRMND＋1，……，BRMND＋3）を８バイトの連続した領域とします。被除数と余りの１桁（４ビット）を左シフトすることにより被除数の最上位桁を余りの最下位エリアに転送し，商が最上位から１桁ずつ被除数の最下位エリアに入ります。

また，商の各桁は“余り－除数”の結果が負になるまで（ボローが生じるまで）繰り返したときの回数とします。

①　除数の０チェックを行う。０ならエラー処理へ分岐する。

② 余りエリアを０クリアする。

③ 被除数，除数の絶対値を取る。商符号フラグ（Xの０ビット），余り符号フラグ（Xの１ビット）被除数，除数のどちらか一方だけが負ならば，商符号フラグをセットする。

被除数が負なら余り符号フラグをセットする。

④ 商のバイト・カウンタ（Bレジスタ）を８に設定する。

⑤ ８バイト続きの商・余りエリアを４ビット左シフトする。

⑥ 余り←余り－除数を行う。

負になれば⑧へジャンプする。

⑦ 商（DEND）をインクリメントする。⑤へジャンプする。

⑧ 引きすぎたので元に戻すため，余り←余り＋除数を行う。

⑨ 商のバイト・カウンタをデクリメントし，０になるまで⑤から⑧の処理を繰り返す。

⑩ 商符号フラグを調べセットされていれば商の２の補数を取る。

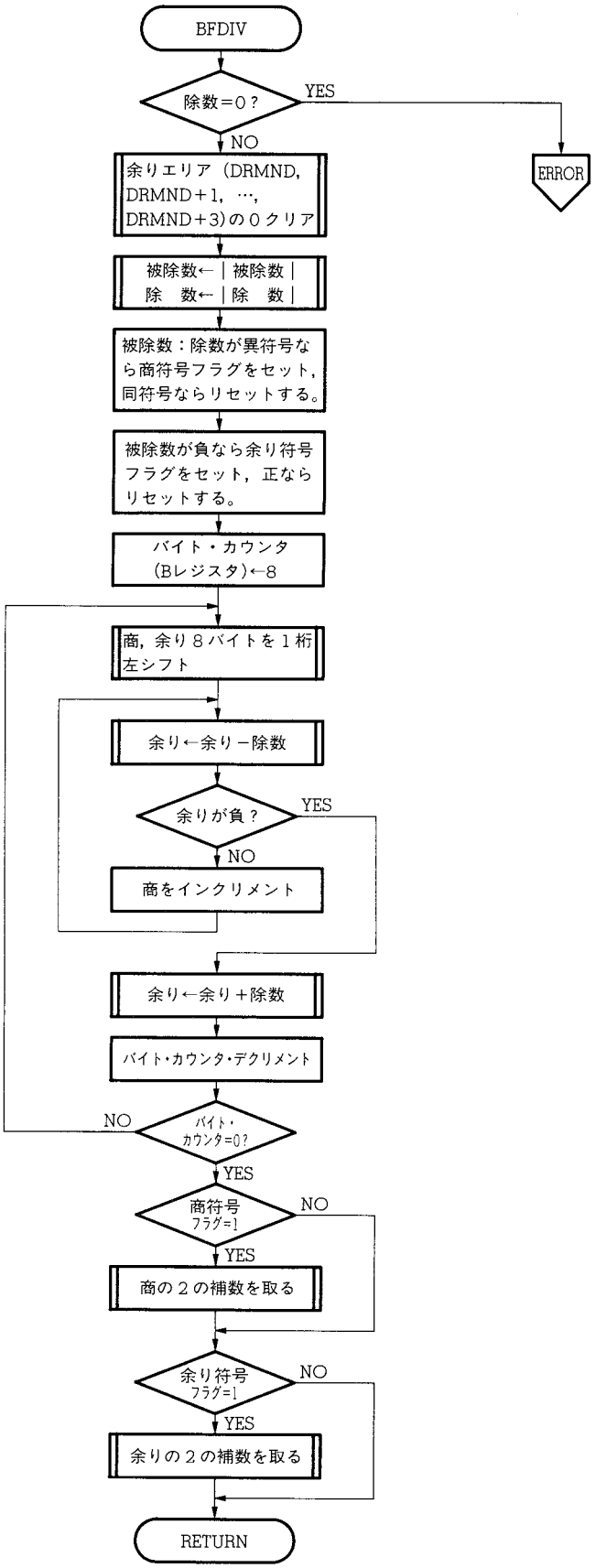
余り符号フラグを調べセットされていれば余りの２の補数を取る。

(6) ステップ数

119バイト

保守／廃止

フロー・チャート



保守／廃止

Assemble list

```
ALNO  STNO ADRS  OBJECT    M I  SOURCE STATEMENT

   1     1                      $       TITLE   ('binary division')
   2     2                              NAME    BFDIVR
   3     3                      ;*************************************************
   4     4                      ;*      binary division                          *
   5     5                      ;*           32 bit <- 32 bit / 32 bit           *
   6     6                      ;*       input condition                         *
   7     7                      ;*            dividend <- (DEND+3,...,DEND)      *
   8     8                      ;*            divisor  <- (DVISOR+3,...,DVISOR)  *
   9     9                      ;*       output condition                        *
  10    10                      ;*            quotient  <- (DEND+3,...,DEND)     *
  11    11                      ;*            remainder <- (DRMND+3,...,DRMND)   *
  12    12                      ;*                                               *
  13    13                      ;*             RSS <- 0                          *
  14    14                      ;*************************************************
  15    15
  16    16                              PUBLIC  BFDIV
  17    17                              EXTRN   BFXADD,BFXSUB
  18    18                              EXTRN   RCLR,COMPL,BCDLS,ERROR
  19    19                              EXTRN   DEND,DVISOR,DRMND
  20    20                      ;
  21    21       (0000.0)       SF_REM  EQU     X.0
  22    22       (0000.1)       SF_QUO  EQU     X.1
  23    23       (0004)         BYTNUM  EQU     4
  24    24                      ;
  25    25 ----                         CSEG
  26    26                              RSS     0
  27    27 0000                BFDIV:
  28    28                      ;
  29    29                      ;       ****  check / divisor = 0 ?  ****
  30    30                      ;
  31    31 0000  BA04                   MOV     C,#BYTNUM       ; C-register <- 4
  32    32 0002  B900                   MOV     A,#0            ; Acc <- 0
  33    33 0004 R670000                 MOVW    HL,#DVISOR      ; HL <- DVISOR
  34    34                      ;
  35    35 0007                BFDIV1:
  36    36 0007  169F                   CMP     [HL+],A
  37    37 0009  8005                   BNZ     $BFDIV2         ; [HL] = 0 ?
  38    38 000B  32FA                   DBNZ    C,$BFDIV1
  39    39                      ;
  40    40                      ;       ****  divisor = 0  ****
  41    41                      ;
  42    42 000D R2C0000                 BR      !ERROR          ; OVER FLOW
  43    43                      ;
  44    44                      ;       ****  quotient 0-clear  ****
  45    45                      ;
  46    46 0010                BFDIV2:
  47    47 0010 R650000                 MOVW    DE,#DRMND       ; DE-register <- DRMND
  48    48 0013 R280000                 CALL    !RCLR
```

```
49    49                        ;
50    50                        ;       ****  complement convert  ****
51    51                        ;
52    52 0016  0390                     CLR1    SF_REM          ; clear remainder sign-flag
53    53 0018  0391                     CLR1    SF_QUO          ; clear quotient sign-flag
54    54 001A R670000                   MOVW    HL,#DEND        ; HL-register <- DEND
55    55 001D  062003                   MOV     A,[HL+3]
56    56 0020  03AF07                   BF      A.7,$BFDIV3
57    57 0023 R280000                   CALL    !COMPL          ; complement subroutine
58    58 0026  0380                     SET1    SF_REM          ; set remainder sign-flag
59    59 0028  0371                     NOT1    SF_QUO          ; not quotient sign-flag
60    60 002A                   BFDIV3:
61    61 002A R670000                   MOVW    HL,#DVISOR      ; HL-register <- DVISOR
62    62 002D  062003                   MOV     A,[HL+3]
63    63 0030  03AF05                   BF      A.7,$BFDIV4
64    64 0033 R280000                   CALL    !COMPL          ; complement subroutine
65    65 0036  0371                     NOT1    SF_QUO          ; not quotient sign-flag
66    66                        ;
67    67                        ;       ****  byte counter set  ****
68    68                        ;
69    69 0038                   BFDIV4:
70    70 0038  BB08                     MOV     B,#8            ; B-register <- 8
71    71                        ;
72    72                        ;       ****  dividend,remainder 1-byte left shift  ****
73    73                        ;
74    74 003A                   BFDIV5:
75    75 003A R670000                   MOVW    HL,#DEND        ; HL <- DEND
76    76 003D  BA08                     MOV     C,#8            ; C-register <- 8
77    77 003F R280000                   CALL    !BCDLS
78    78                        ;
79    79                        ;       ****  subtract divisor from dividend  ****
80    80                        ;
81    81 0042                   BFDIV6:
82    82 0042 R650000                   MOVW    DE,#DVISOR      ; DE <- DVISOR
83    83 0045 R670000                   MOVW    HL,#DRMND       ; HL <- DRMND
84    84 0048 R280000                   CALL    !BFXSUB
85    85                        ;
86    86 004B  4F                       DECW    HL              ; decrement HL
87    87 004C  5D                       MOV     A,[HL]
88    88 004D  03BF09                   BT      A.7,$BFDIV7     ; if borrow
89    89                        ;
90    90 0050  B901                     MOV     A,#1            ; ACC ← 1
91    91 0052 R670000                   MOVW    HL,#DEND
92    92 0055  16D8                     ADD     [HL],A          ; increment DEND
93    93                        ;
94    94 0057  14E9                     BR      $BFDIV6
```

```
 95    95                    ;
 96    96                    ;       ****  if borrow divisor + dividend  ****
 97    97                    ;
 98    98 0059               BFDIV7:
 99    99 0059 R650000               MOVW    DE,#DVISOR      ; DE <- DVISOR
100   100 005C R670000               MOVW    HL,#DRMND       ; HL <- DRMND
101   101 005F R280000               CALL    !BFXADD
102   102                   ;
103   103                   ;       ****  check / division end ?  ****
104   104                   ;
105   105 0062  33D6                 DBNZ    B,$BFDIV5
106   106                   ;
107   107 0064  03A006               BF      SF_REM,$BFDIV8
108   108 0067 R670000               MOVW    HL,#DRMND
109   109 006A R280000               CALL    !COMPL
110   110                   ;
111   111 006D              BFDIV8:
112   112 006D  03A106               BF      SF_QUO,$BFDIV9
113   113 0070 R670000               MOVW    HL,#DEND
114   114 0073 R280000               CALL    !COMPL
115   115                   ;
116   116 0076              BFDIV9:
117   117 0076  56                   RET
118   118
119   119                            ENDS
120   120                            END

Segment informations:

ADRS  LEN    NAME

0000  0077H  ?CSEG
```

```
          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG       ?CSEG       25#
BCDLS     ----H    E        EXT           18@    77
BFDIV       0H     R ADDR   PUB  ?CSEG      16@    27#
BFDIV1      7H     R ADDR        ?CSEG      35#    38
BFDIV2     10H     R ADDR        ?CSEG      37     46#
BFDIV3     2AH     R ADDR        ?CSEG      56     60#
BFDIV4     38H     R ADDR        ?CSEG      63     69#
BFDIV5     3AH     R ADDR        ?CSEG      74#   105
BFDIV6     42H     R ADDR        ?CSEG      81#    94
BFDIV7     59H     R ADDR        ?CSEG      88     98#
BFDIV8     6DH     R ADDR        ?CSEG     107    111#
BFDIV9     76H     R ADDR        ?CSEG     112    116#
BFDIVR             MOD                      2#
BFXADD    ----H    E        EXT           17@   101
BFXSUB    ----H    E        EXT           17@    84
BYTNUM      4H       NUM                    23#    31
COMPL     ----H    E        EXT           18@    57     64    109    114
DEND      ----H    E        EXT           19@    54     75     91    113
DRMND     ----H    E        EXT           19@    47     83    100    108
DVISOR    ----H    E        EXT           19@    33     61     82     99
ERROR     ----H    E        EXT           18@    42
RCLR      ----H    E        EXT           18@    48
SF_QUO      0H.1   RBIT                   22#    53     59     65    112
SF_REM      0H.0   RBIT                   21#    52     58    107

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

保守／廃止

3.2　符号付き10進演算

ここでは，符号付き10進演算について，加減乗除算を紹介します。

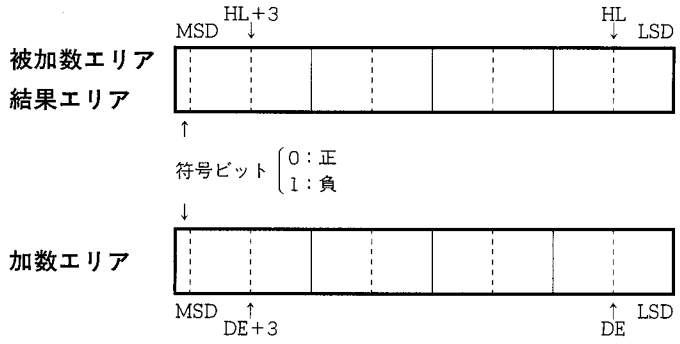
最上位桁の最上位ビットを符号ビットとし，残りのビットで数値を表します。

ただし，数値は絶対値とします。

3.2.1　10進加算

８桁←８桁＋８桁

(1) 使用メモリ・エリア



(2) 使用レジスタ

A, B, C, D, E, H, L

(3) 入力条件

(1)のように，加数，被加数のメモリ・アドレスは，それぞれ次のレジスタ・ペアで指定します。

HLレジスタ・ペア←被加数８桁（４バイト）が格納してあるエリアの最下位アドレス

DEレジスタ・ペア←加数８桁（４バイト）が格納してあるエリアの最下位アドレス

(4) 出力条件

演算結果を，(1)の結果エリアに格納します。

ただし，オーバフローまたはアンダフローが発生したときは，エラー処理へ分岐します。

注意　演算範囲は，−79999999～79999999となります。

(5) 処理手順

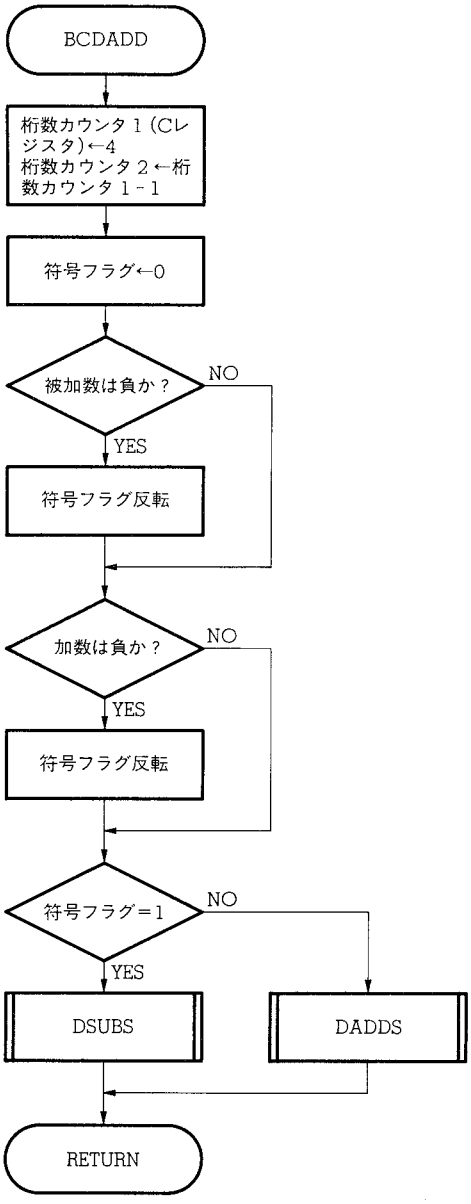
この演算プログラムでは，加数，被加数が同符号なら加算を行い，異符号なら減算を行います。

① 桁数カウンタ１（Cレジスタ）に４を設定する。
　桁数カウンタ２（Bレジスタ）に“桁数カウンタ１の値－１”を設定する。
② 加数，被加数が異符号なら⑮の処理へ
③ キャリー・フラグ，符号フラグ（ユーザ・フラグ）をクリアする。
④ 被加数アドレスで指定する被加数１バイトをアキュームレータにロードする。
⑤ 加数アドレスで指定する加数１バイトをキャリー・フラグとともにアキュームレータに加え，加数アドレスをインクリメントする。演算結果を10進補正し，被加数アドレスで指定するメモリに格納し，その後被加数アドレスをインクリメントする。
⑥ 桁数カウンタ２をデクリメントし０になるまで④から⑤までの処理を繰り返す。
⑦ 被加数アドレスで指定する被加数１バイトをアキュームレータにロードする。
⑧ 加数アドレスで指定する加数１バイトをキャリー・フラグとともにアキュームレータに加える。
⑨ キャリー・フラグが０なら⑪の処理へ
⑩ 符号フラグをセットし，キャリー・フラグをクリアする。
⑪ アキュームレータを10進補正する。
⑫ キャリー・フラグが１またはアキュームレータのbit7が１ならオーバフローなのでエラー処理へ。
⑬ 符号フラグが１ならアキュームレータのbit7をセットする。
⑭ アキュームレータの内容を被加数アドレスで指定するメモリに格納し，演算を終了する。
⑮ 減数を正にし，符号フラグをクリアする。
⑯ 被減数が負なら被減数を正にし，符号フラグをセットする。
⑰ キャリー・フラグをクリアする。
⑱ 被減数アドレスで指定する被減数１バイトをアキュームレータにロードする。
⑲ アキュームレータから減数アドレスで指定する減数１バイトをキャリー・フラグとともに引き，減数アドレスを（＋1）する。演算結果を10進補正し，被減数アドレスで指定するメモリに格納し，その後被減数アドレスをインクリメントする。
⑳ 桁数カウンタ１を（－1）し，０になるまで，⑱から⑲までの処理を繰り返す。
㉑ キャリー・フラグが０なら㉓の処理へ
㉒ 結果の10の補数をとり，符号フラグを反転する。
㉓ 結果が０なら演算終了となる。
㉔ 符号フラグが１なら㉕の処理へ，０なら演算終了となる。
㉕ 結果の符号ビットをセットし，演算終了となる。

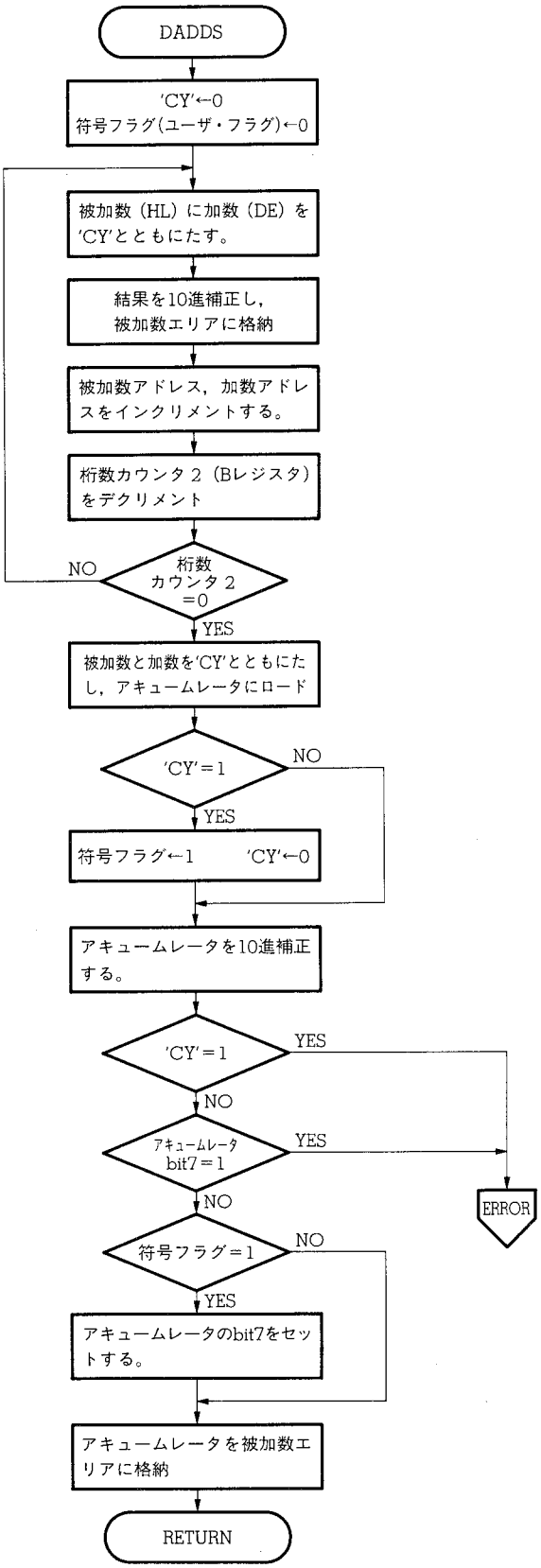
(6) ステップ数

147バイト

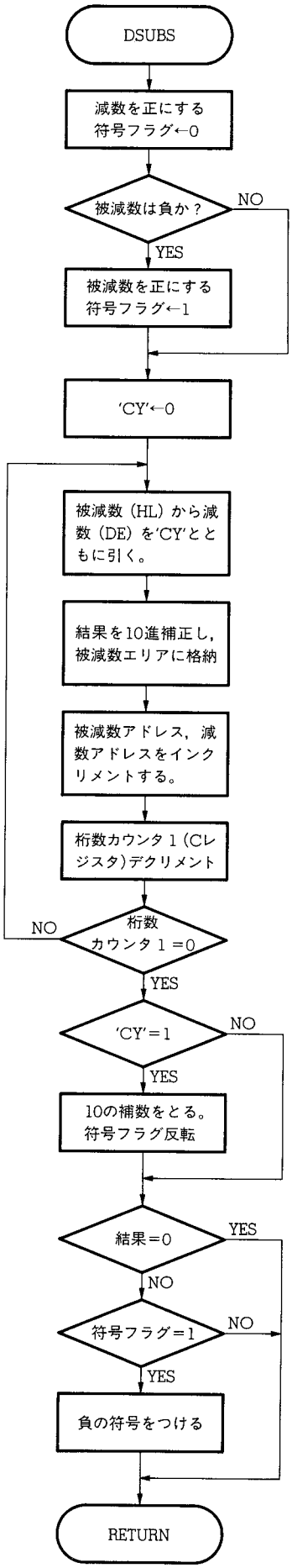
フロー・チャート



フロー・チャート



フロー・チャート



Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

  1     1                      $       TITLE   ('decimal addition')
  2     2                              NAME    BCDADR
  3     3                      ;*******************************************************
  4     4                      ;*      decimal addition                               *
  5     5                      ;*           8 digit <- 8 digit + 8 digit              *
  6     6                      ;*       input condition                               *
  7     7                      ;*            HL-register <- augend area top.address   *
  8     8                      ;*            DE-register <- addend  area top.address  *
  9     9                      ;*       output condition                              *
 10    10                      ;*            result <- (HL+3,HL+2,HL+1,HL)            *
 11    11                      ;*                                                     *
 12    12                      ;*             RSS <- 0                                *
 13    13                      ;*******************************************************
 14    14                              PUBLIC  BCDADD,BCDAD1,BCDAD2
 15    15                              PUBLIC  DADDS
 16    16                              PUBLIC  DSUBS
 17    17                              EXTRN   ERROR
 18    18       (0004)         BYTNUM  EQU     4
 19    19       (01FF.7)       SFLAG   EQU     PSWH.7
 20    20                      ;
 21    21 ----                         CSEG
 22    22                              RSS     0
 23    23 0000                BCDADD:
 24    24 0000  BA04                   MOV     C,#BYTNUM       ; C-register <- 4
 25    25 0002                BCDAD1:
 26    26 0002  2432                   MOV     B,C             ; B-register <- C-register - 1
 27    27 0004  CB                     DEC     B
 28    28 0005                BCDAD2:
 29    29 0005  1730                   MOV     A,[HL+B]
 30    30 0007  172D                   XOR     A,[DE+B]
 31    31                      ;
 32    32 0009  03BF04                 BT      A.7,$BCDAD3
 33    33 000C R281400                 CALL    !DADDS
 34    34 000F  56                     RET
 35    35                      ;
 36    36 0010                BCDAD3:
 37    37 0010 R283B00                 CALL    !DSUBS
 38    38 0013  56                     RET
 39    39
 40    40                              ENDS
```

```
41    41
42    42                    ;=================================================
43    43                    ;       ***** decimal addition subroutine *****
44    44                    ;=================================================
45    45 ----                       CSEG
46    46                          RSS     0
47    47 0014               DADDS:
48    48 0014  40                 CLR1    CY
49    49 0015  029F               CLR1    SFLAG
50    50 0017               DADDS1:
51    51 0017  5D                 MOV     A,[HL]
52    52 0018  1609               ADDC    A,[DE+]
53    53 001A  05FE               ADJBA                   ; decimal adjust
54    54 001C  51                 MOV     [HL+],A
55    55 001D  33F8               DBNZ    B,$DADDS1
56    56                    ;
57    57 001F  5D                 MOV     A,[HL]
58    58 0020  1649               ADDC    A,[DE]
59    59                    ;
60    60 0022               DADDS2:
61    61 0022  8203               BNC     $DADDS3
62    62 0024  028F               SET1    SFLAG           ; set sign-flag
63    63 0026  40                 CLR1    CY
64    64                    ;
65    65 0027               DADDS3:
66    66 0027  05FE               ADJBA                   ; decimal adjust
67    67 0029  8203               BNC     $DADDS4
68    68 002B R2C0000             BR      !ERROR
69    69 002E               DADDS4:
70    70 002E  03AF03             BF      A.7,$DADDS5
71    71 0031 R2C0000             BR      !ERROR
72    72                    ;
73    73 0034               DADDS5:
74    74 0034  02AF02             BF      SFLAG,$DADDS6
75    75 0037  038F               SET1    A.7
76    76 0039               DADDS6:
77    77 0039  55                 MOV     [HL],A
78    78 003A  56                 RET
79    79
80    80                          ENDS
81    81
82    82                    $     EJECT
```

```
 83    83
 84    84                    ;=====================================================
 85    85                    ;       ***** decimal subtraction subroutine *****
 86    86                    ;=====================================================
 87    87                    ;
 88    88 ----                       CSEG
 89    89                           RSS     0
 90    90 003B               DSUBS:
 91    91 003B  244F                MOVW    RP2,HL          ; save HL-register
 92    92 003D  029F                CLR1    SFLAG           ; clear sign-flag
 93    93 003F  1720                MOV     A,[DE+B]
 94    94 0041  039F                CLR1    A.7
 95    95 0043  17A0                MOV     [DE+B],A
 96    96 0045  1730                MOV     A,[HL+B]
 97    97 0047  03AF06              BF      A.7,$DSUBS1
 98    98                    ;
 99    99 004A  039F                CLR1    A.7
100   100 004C  17B0                MOV     [HL+B],A
101   101 004E  028F                SET1    SFLAG           ; set sign-flag
102   102                   ;
103   103 0050              DSUBS1:
104   104 0050  2462                MOV     R6,C            ; R6ﾚｼﾞｽﾀ ← Cﾚｼﾞｽﾀ
105   105 0052  40                  CLR1    CY
106   106 0053              DSUBS2:
107   107 0053  5D                  MOV     A,[HL]
108   108 0054  160B                SUBC    A,[DE+]
109   109 0056  05FF                ADJBS
110   110 0058  51                  MOV     [HL+],A
111   111 0059  32F8                DBNZ    C,$DSUBS2
112   112                   ;
113   113 005B  821D                BNC     $DSUBS5
114   114 005D  24EC                MOVW    HL,RP2          ; load HL-register
115   115 005F  2426                MOV     C,R6            ; load C-register
116   116 0061              DSUBS3:
117   117 0061  B999                MOV     A,#99H          ; (HL) <- 9 - (HL)
118   118 0063  165A                SUB     A,[HL]
119   119 0065  05FF                ADJBS
120   120 0067  51                  MOV     [HL+],A
121   121 0068  32F7                DBNZ    C,$DSUBS3
122   122                   ;
123   123 006A  24EC                MOVW    HL,RP2          ; load HL-register
124   124 006C  41                  SET1    CY
125   125                   ;
126   126 006D  2426                MOV     C,R6            ; load C-register
127   127 006F              DSUBS4:
128   128 006F  B900                MOV     A,#0            ; Acc <- 0
129   129 0071  1659                ADDC    A,[HL]
130   130 0073  05FE                ADJBA
131   131 0075  51                  MOV     [HL+],A
132   132 0076  32F7                DBNZ    C,$DSUBS4
133   133 0078  027F                NOT1    SFLAG
```

```
  134   134                     ;
  135   135                     ;       ****  check / result = 0  ****
  136   136                     ;
  137   137 007A                DSUBS5:
  138   138 007A  2426                  MOV     C,R6            ; load C-register
  139   139 007C  24EC                  MOVW    HL,RP2
  140   140 007E  B900                  MOV     A,#0
  141   141 0080                DSUBS6:
  142   142 0080  169F                  CMP     [HL+],A
  143   143 0082  8003                  BNZ     $DSUBS7
  144   144 0084  32FA                  DBNZ    C,$DSUBS6
  145   145 0086  56                    RET
  146   146 0087                DSUBS7:
  147   147 0087  02AF08                BF      SFLAG,$DSUBS8
  148   148 008A  24EC                  MOVW    HL,RP2
  149   149 008C  1730                  MOV     A,[HL+B]
  150   150 008E  038F                  SET1    A.7
  151   151 0090  17B0                  MOV     [HL+B],A
  152   152 0092                DSUBS8:
  153   153 0092  56                    RET
  154   154                             ENDS
  155   155                             END
  156   156

Segment informations:

ADRS  LEN    NAME

0000  0093H  ?CSEG
```

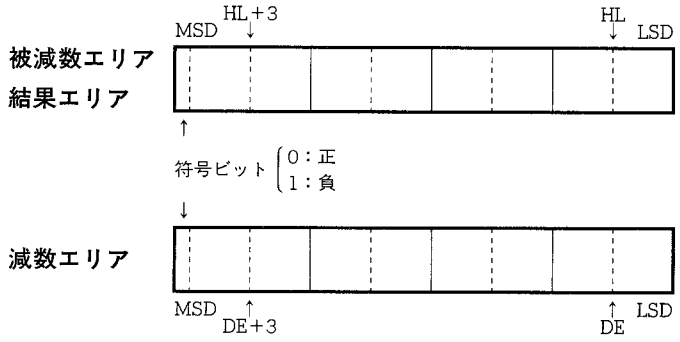
```
          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG             CSEG       ?CSEG       21#    45#    88#
BCDAD1     2H   R ADDR  PUB  ?CSEG       14@    25#
BCDAD2     5H   R ADDR  PUB  ?CSEG       14@    28#
BCDAD3    10H   R ADDR       ?CSEG       32     36#
BCDADD     0H   R ADDR  PUB  ?CSEG       14@    23#
BCDADR            MOD                     2#
BYTNUM     4H     NUM                     18#    24
DADDS     14H   R ADDR  PUB  ?CSEG       15@    33     47#
DADDS1    17H   R ADDR       ?CSEG       50#    55
DADDS2    22H   R ADDR       ?CSEG       60#
DADDS3    27H   R ADDR       ?CSEG       61     65#
DADDS4    2EH   R ADDR       ?CSEG       67     69#
DADDS5    34H   R ADDR       ?CSEG       70     73#
DADDS6    39H   R ADDR       ?CSEG       74     76#
DSUBS     3BH   R ADDR  PUB  ?CSEG       16@    37     90#
DSUBS1    50H   R ADDR       ?CSEG       97    103#
DSUBS2    53H   R ADDR       ?CSEG      106#   111
DSUBS3    61H   R ADDR       ?CSEG      116#   121
DSUBS4    6FH   R ADDR       ?CSEG      127#   132
DSUBS5    7AH   R ADDR       ?CSEG      113    137#
DSUBS6    80H   R ADDR       ?CSEG      141#   144
DSUBS7    87H   R ADDR       ?CSEG      143    146#
DSUBS8    92H   R ADDR       ?CSEG      147    152#
ERROR    ----H   E       EXT             17@    68     71
SFLAG     1FFH.7  RBIT                    19#    49     62     74     92    101    133    147

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

保守／廃止

3.2.2　10進減算

8桁←8桁－8桁

(1) 使用メモリ・エリア



(2) 使用レジスタ

A, B, C, D, E, H, L

(3) 入力条件

(1)のように，減数，被減数のメモリ・アドレスは，それぞれ次のレジスタ・ペアで指定します。

HLレジスタ・ペア←被減数8桁（4バイト）が格納してあるエリアの最下位アドレス

DEレジスタ・ペア←減数8桁（4バイト）が格納してあるエリアの最下位アドレス

(4) 出力条件

演算結果を(1)で示す結果エリアに格納します。

ただし，オーバフローまたは，アンダフローが発生したときは，エラー処理へ分岐します。

注意　演算範囲は，－79999999～79999999となります。

(5) 処理手順

この演算プログラムでは，被減数－減数を被減数＋（－減数）という加算に変換して行います。

次に処理手順を示します。

① 桁数カウンタ（Cレジスタ）に4を設定する。

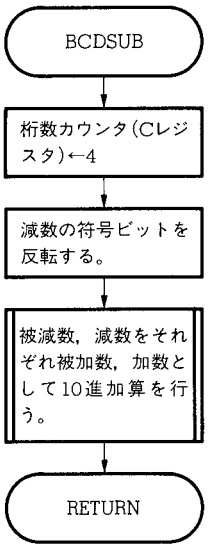
② 減数の符号ビットを反転する。

③ 被減数，減数をそれぞれ被加数，加数として10進加算を行う。

(6) ステップ数

15バイト

フロー・チャート



Assemble list

```
 ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

    1     1                      $       TITLE   ('decimal subtraction')
    2     2                              NAME    BCDSUR
    3     3                      ;*************************************************************
    4     4                      ;*      decimal subtraction                                  *
    5     5                      ;*           8 digit <- 8 digit - 8 digit                    *
    6     6                      ;*       input condition                                     *
    7     7                      ;*            HL-register <- minuend    area top.address    *
    8     8                      ;*            DE-register <- subtrahend area top.address    *
    9     9                      ;*       output condition                                    *
   10    10                      ;*            result <- (HL+3,HL+2,HL+1,HL)                  *
   11    11                      ;*                                                           *
   12    12                      ;*             RSS <- 0                                      *
   13    13                      ;*************************************************************
   14    14                      ;
   15    15                              PUBLIC  BCDSUB
   16    16                              EXTRN   BCDADD,BCDAD2
   17    17        (0004)        BYTNUM  EQU     4
   18    18                      ;
   19    19 ----                         CSEG
   20    20                              RSS     0
   21    21 0000                BCDSUB:
   22    22 0000  BA04                   MOV     C,#BYTNUM       ; C-register <- 4
   23    23 0002                BCDSU1:
   24    24 0002  2432                   MOV     B,C             ; B-register <- C-register - 1
   25    25 0004  CB                     DEC     B
   26    26                      ;
   27    27 0005  1720                   MOV     A,[DE+B]
   28    28 0007  037F                   NOT1    A.7
   29    29 0009  17A0                   MOV     [DE+B],A
   30    30                      ;
   31    31 000B R280000                 CALL    !BCDAD2
   32    32                      ;
   33    33 000E  56                     RET
   34    34                              ENDS
   35    35                              END

Segment informations:

ADRS  LEN    NAME

0000  000FH  ?CSEG
```

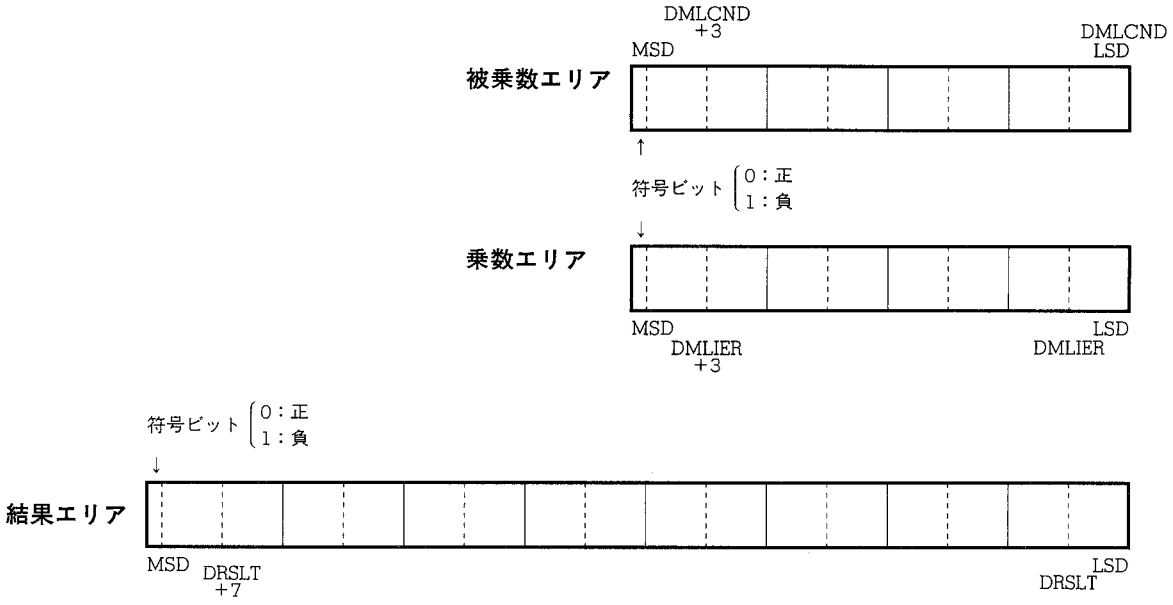
```
          Cross-Reference List

NAME      VALUE   R ATTR  RTYP SEGNAME  XREFS

?CSEG               CSEG       ?CSEG       19#
BCDAD2    ----H   E          EXT              16@    31
BCDADD    ----H   E          EXT              16@
BCDSU1      2H    R ADDR          ?CSEG       23#
BCDSUB      0H    R ADDR  PUB     ?CSEG       15@    21#
BCDSUR              MOD                       2#
BYTNUM      4H      NUM                      17#    22

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

保守／廃止

3.2.3　10進乗算

16桁←８桁×８桁

(1) 使用メモリ・エリア



(2) 使用するレジスタ

A, C, B, R4, D, E, H, L

(3) 入力条件

(1)で示すように被乗数８桁，乗数８桁をそれぞれ，

被乗数（DMLCND，DMLCND＋1，……，DMLCND＋3）

乗　数（DMLIER，DMLIER＋1，……，DMLIER＋3）

に格納します。

(4) 出力条件

演算結果を結果エリア（DRSLT，DRSLT＋1，……，DRSLT＋7）に格納します。

注意1．乗数，被乗数の有効範囲は，－79999999～79999999となります。

2．演算結果範囲は，－6399999840000001～6399999840000001となります。

保守／廃止

(5) 処理手順

この演算プログラムでは，乗数を１桁（４ビット）右シフトすることにより，最下位桁から１桁ずつ加算カウンタにロードし，これをカウンタとして，(結果←結果＋被乗数）を繰り返します。

また，加算カウンタによる加算が終了後，１桁上位の乗数によって加算を行うため，結果エリアを１桁（４ビット）右シフトしておきます。

次に処理手順を示します。

① 結果エリアを０クリアしておく。

② 乗数，被乗数の絶対値を取る。乗数，被乗数が異符号なら符号フラグ（ユーザ・フラグ）をセット，同符号なら符号フラグをリセットする。

③ 桁カウンタ（Bレジスタ）を８に設定する。

④ 乗数を１桁右シフトすることにより，最下位桁を加算カウンタ（Cレジスタ）にロードする。

⑤ 桁上げ数（R4）をクリアしておく。

⑥ 加算カウンタが，０なら⑨の処理へ。

⑦ 10進加算（結果（上位８桁）←結果（上位８桁)＋被乗数）を行い，オーバフローが生じたら桁上げ数を（＋1）する。

⑧ 加算カウンタを（－1）し，０になるまで⑦の処理を繰り返す。

⑨ 桁上げ数を含めて，結果エリアを１桁（４ビット）右シフトする。
桁上げ数が，結果エリアの最上位桁に格納される。

⑩ 桁カウンタを（－1）し，０になるまで④から⑨までの処理を繰り返す。

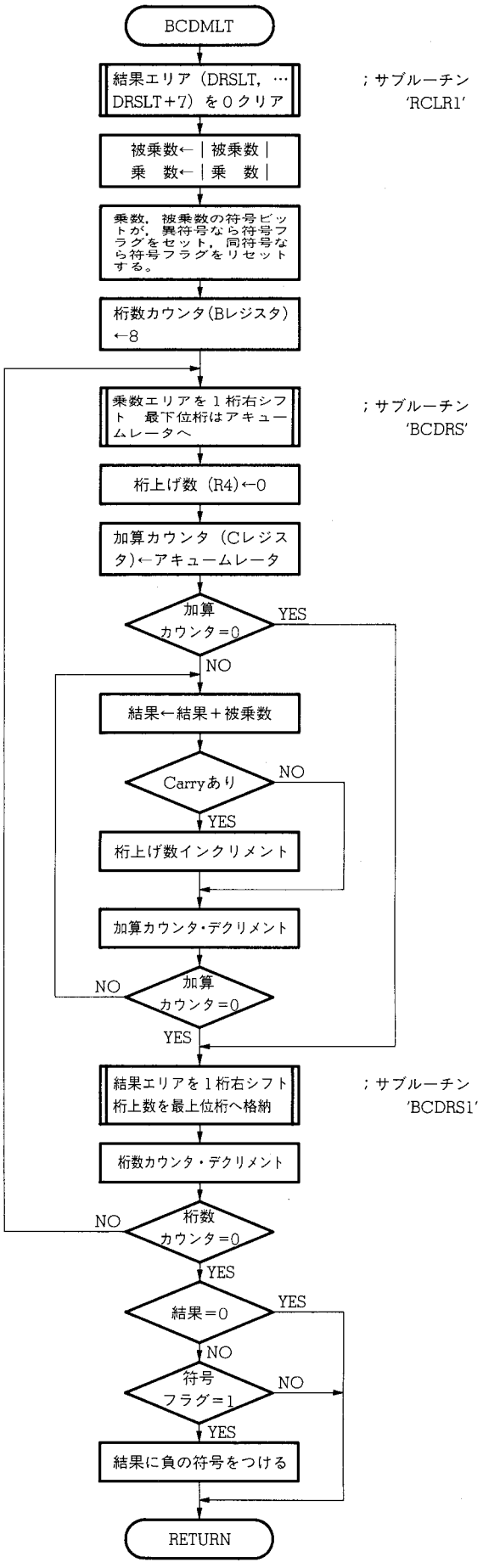
⑪ 結果が０なら演算終了とする。

⑫ 符号フラグが１なら結果エリアの符号ビットをセットする。

(6) ステップ数

121バイト

フロー・チャート



保守／廃止

Assemble list

```
ALNO  STNO ADRS  OBJECT    M I  SOURCE STATEMENT

   1     1                      $       TITLE   ('decimal multiplication')
   2     2                              NAME    BCDMLR
   3     3                      ;*************************************************************
   4     4                      ;*      decimal multiplication                               *
   5     5                      ;*           16 digit <- 8 digit * 8 digit                   *
   6     6                      ;*       input condition                                     *
   7     7                      ;*             multiplicand <- (DMLCND+3,...,DMLCND)         *
   8     8                      ;*             multiplier   <- (DMLIER+3,...,DMLIER)         *
   9     9                      ;*       output condition                                    *
  10    10                      ;*             result <- (DRSLT+7,...,DRSLT)                 *
  11    11                      ;*                                                           *
  12    12                      ;*              RSS <- 0                                     *
  13    13                      ;*************************************************************
  14    14                      ;
  15    15                              PUBLIC  BCDMLT
  16    16                              EXTRN   RCLR1,BCDRS,BCDRS1
  17    17                              EXTRN   DMLCND,DMLIER,DRSLT
  18    18        (01FF.7)      SFLAG   EQU     PSWH.7
  19    19                      ;
  20    20 ----                         CSEG
  21    21                              RSS     0
  22    22 0000                 BCDMLT:
  23    23                      ;
  24    24                      ;       ****  result area 0-clear  ****
  25    25                      ;
  26    26 0000  BA08                   MOV     C,#8            ; C-register <- 8
  27    27 0002 R650000                 MOVW    DE,#DRSLT       ; DE <- DRSLT
  28    28 0005 R280000                 CALL    !RCLR1
  29    29                      ;
  30    30                      ;       ****  check / sign  ****
  31    31                      ;
  32    32 0008  029F                   CLR1    SFLAG           ; clear sign-flag
  33    33 000A R09F00300               MOV     A,!DMLCND+3
  34    34 000E  03AF08                 BF      A.7,$BCDML1
  35    35 0011  039F                   CLR1    A.7
  36    36 0013 R09F10300               MOV     !DMLCND+3,A
  37    37 0017  027F                   NOT1    SFLAG           ; not sign-flag
  38    38                      ;
  39    39 0019                 BCDML1:
  40    40 0019 R09F00300               MOV     A,!DMLIER+3
  41    41 001D  03AF08                 BF      A.7,$BCDML2
  42    42 0020  039F                   CLR1    A.7
  43    43 0022 R09F10300               MOV     !DMLIER+3,A
  44    44 0026  027F                   NOT1    SFLAG           ; not sign-flag
```

```
45    45                    ;
46    46                    ;       ****  digit counter set  ****
47    47                    ;
48    48 0028               BCDML2:
49    49 0028  BB08                 MOV     B,#8            ; B-register <- 8
50    50                    ;
51    51                    ;       ****  multiplier right shift  ****
52    52                    ;
53    53 002A               BCDML3:
54    54 002A R670300               MOVW    HL,#DMLIER+3
55    55 002D  BA04                 MOV     C,#4            ; C-register <- 4
56    56 002F R280000               CALL    !BCDRS
57    57 0032  2421                 MOV     C,A             ; C-register <- Acc
58    58                    ;
59    59 0034  BC00                 MOV     R4,#0           ; carry <- 0
60    60                    ;
61    61                    ;       ****  check / multiplier = 0 ?  ****
62    62                    ;
63    63 0036  A800                 ADD     A,#0
64    64 0038  8118                 BZ      $BCDML7
65    65                    ;
66    66                    ;       ****  result <- DMLCND + result  ****
67    67                    ;
68    68 003A               BCDML4:
69    69 003A R650000               MOVW    DE,#DMLCND      ; DE <- DMLCND
70    70 003D R670400               MOVW    HL,#DRSLT+4     ; HL <- DRSLT+4
71    71 0040  40                   CLR1    CY              ; clear carry
72    72                            RSS     1
73    73 0041  43                   SWRS
74    74 0042  BE04                 MOV     C,#4            ; C-register <- 4
75    75                    ;
76    76 0044               BCDML5:
77    77 0044  5D                   MOV     A,[HL]
78    78 0045  1609                 ADDC    A,[DE+]
79    79 0047  05FE                 ADJBA                   ; decimal adjust
80    80 0049  51                   MOV     [HL+],A
81    81 004A  32F8                 DBNZ    C,$BCDML5
82    82                            RSS     0
83    83 004C  43                   SWRS
84    84 004D  8201                 BNC     $BCDML6
85    85 004F  C4                   INC     R4
86    86 0050               BCDML6:
87    87 0050  32E8                 DBNZ    C,$BCDML4
88    88                    ;
89    89                    ;       ****  result right shift with carry  ****
90    90                    ;
91    91 0052               BCDML7:
92    92 0052  D4                   MOV     A,R4
93    93 0053 R670700               MOVW    HL,#DRSLT+7     ; HL <- DRSLT+7
94    94 0056  BA08                 MOV     C,#8
95    95 0058 R280000               CALL    !BCDRS1
```

```
 96    96                    ;
 97    97                    ;       ****  check / multiply end ?  ****
 98    98                    ;
 99    99 005B  33CD                 DBNZ    B,$BCDML3
100   100                    ;
101   101                    ;       ****  check / multiply = 0  ****
102   102                    ;
103   103 005D R670000              MOVW    HL,#DRSLT
104   104 0060  BA08                MOV     C,#8
105   105 0062  B900                MOV     A,#0
106   106 0064              BCDML8:
107   107 0064  169F                CMP     [HL+],A
108   108 0066  8003                BNZ     $BCDML9
109   109 0068  32FA                DBNZ    C,$BCDML8
110   110 006A  56                  RET
111   111                    ;
112   112                    ;       ****  check / sign-flag  ****
113   113                    ;
114   114 006B              BCDML9:
115   115 006B  02AF0A              BF      SFLAG,$BCDM10
116   116 006E R09F00700            MOV     A,!DRSLT+7
117   117 0072  038F                SET1    A.7
118   118 0074 R09F10700            MOV     !DRSLT+7,A
119   119                    ;
120   120 0078              BCDM10:
121   121 0078  56                  RET
122   122                           ENDS
123   123                           END

Segment informations:

ADRS  LEN    NAME

0000  0079H  ?CSEG
```

```
          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG        ?CSEG       20#
BCDM10     78H   R ADDR        ?CSEG      115    120#
BCDML1     19H   R ADDR        ?CSEG       34     39#
BCDML2     28H   R ADDR        ?CSEG       41     48#
BCDML3     2AH   R ADDR        ?CSEG       53#    99
BCDML4     3AH   R ADDR        ?CSEG       68#    87
BCDML5     44H   R ADDR        ?CSEG       76#    81
BCDML6     50H   R ADDR        ?CSEG       84     86#
BCDML7     52H   R ADDR        ?CSEG       64     91#
BCDML8     64H   R ADDR        ?CSEG      106#   109
BCDML9     6BH   R ADDR        ?CSEG      108    114#
BCDMLR             MOD                     2#
BCDMLT      0H   R ADDR  PUB   ?CSEG       15@    22#
BCDRS     ----H   E       EXT               16@    56
BCDRS1    ----H   E       EXT               16@    95
DMLCND    ----H   E       EXT               17@    33     36     69
DMLIER    ----H   E       EXT               17@    40     43     54
DRSLT     ----H   E       EXT               17@    27     70     93    103    116    118
RCLR1     ----H   E       EXT               16@    28
SFLAG      1FFH.7   RBIT                    18#    32     37     44    115

Target chip:uPD78320
Assembly complete,      0 error(s) and      0 warning(s) found. (    0)
```

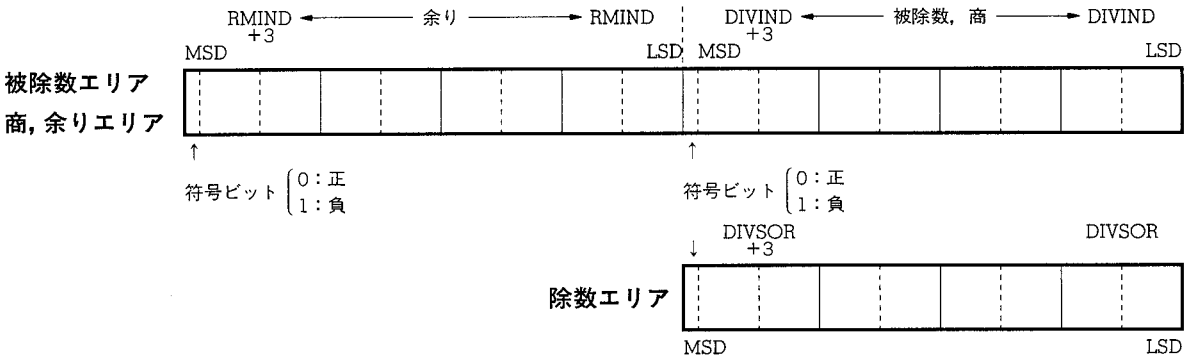
保守／廃止

3.2.4　10進除算

商　8桁←8桁÷8桁

余り8桁

(1) 使用メモリ・エリア



(2) 使用レジスタ

X，A，B，C，D，E，H，L

(3) 入力条件

被除数8桁，除数8桁をそれぞれ，

被除数（DIVIND，DIVDND＋1，……，DIVIND＋3）

除　数（DIVSOR，DIVSOR＋1，……，DIVSOR＋3）

に格納します。

(4) 出力条件

演算結果を，(1)で示す商，余りエリアに格納します。

ただし，除数が0ならエラー処理へ分岐します。

注意　演算範囲は，−79999999〜79999999となります。

(5) 処理手順

この演算プログラムでは，被除数(DIVIND，DIVIND＋1，……，DIVIND＋3)と余り（RMIND，RMIND＋1，……，RMIND＋3）エリアを8バイトの連続した領域とします。被除数と余りの1桁左シフトにより，被除数の最上位桁を余りの最下位エリアに転送し，商が最上位から1桁ずつ被除数の最下位エリアに入ります。

また，商の各桁は“余り−除数”の結果が負になるまで（ボローが生じるまで）繰り返したときの回数となります。

次に処理手順を示します。

① 除数の０チェックを行う。０ならエラー処理へ

② 余りエリア（RMIND，RMIND＋1……，RMIND＋3）を０クリアする。

③ 除数，被除数の絶対値を取る。除数，被除数が異符号なら商符号フラグ(Xの０ビット)をセット，同符号なら商符号フラグをリセットする。

被除数が，負なら余り符号フラグ(Xの１ビット)をセット，正なら余り符号フラグをリセットする。

④ 商の桁カウンタ（Bレジスタ）を８に設定する。

⑤ ８バイト続きの商，余りを１桁（４ビット）左シフトする。

⑥ 10進減算（余り←余り－除数）を行う。

負になれば⑧の処理へ

⑦ 商（DIVIND）を（＋1）する。⑥の処理へ

⑧ 引きすぎたので元に戻す。

10進加算（余り←余り＋除数）を行う。

⑨ 商の桁カウンタを（－1）し，０になるまで⑤から⑧の処理を繰り返します。

⑩ 商が０なら⑫の処理へ

⑪ 商符号フラグが１なら商エリアの符号ビットをセットする。

⑫ 余りが０なら演算を終了する。

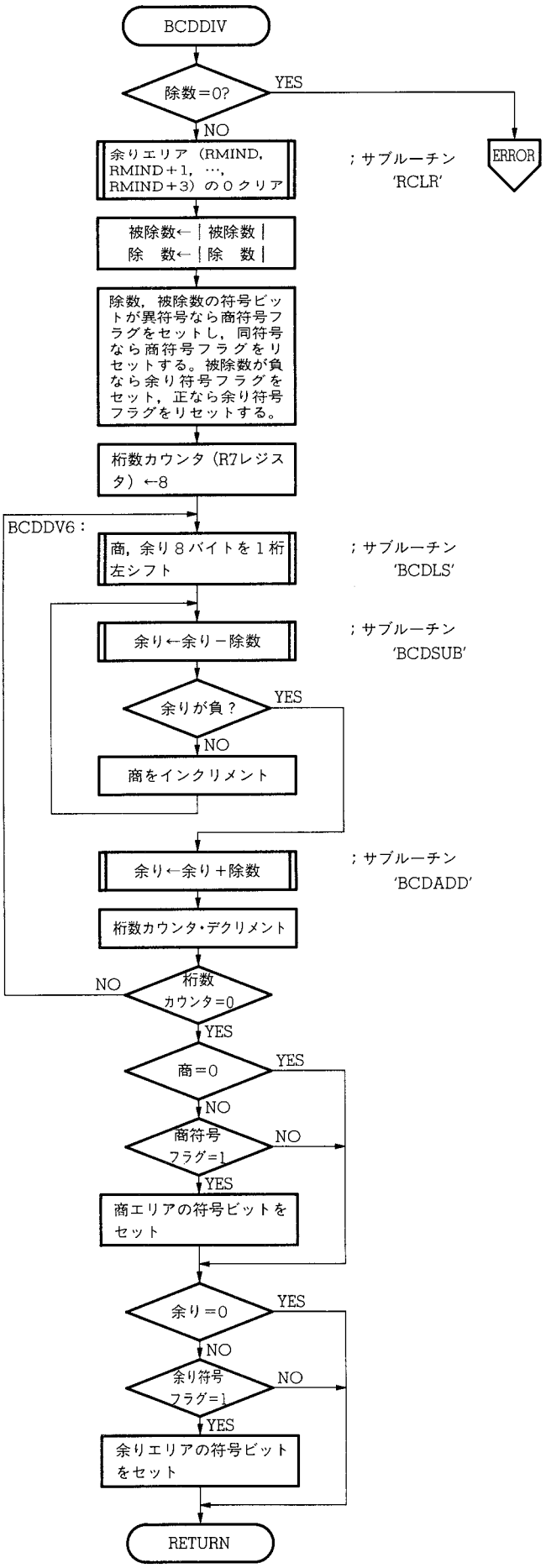
⑬ 余り符号フラグが１なら余りエリアの符号ビットをセットする。

(6) ステップ数

164バイト

保守／廃止

フロー・チャート



保守／廃止

Assemble list

```
ALNO  STNO ADRS  OBJECT    M I  SOURCE STATEMENT

  1     1                      $       TITLE   ('decimal division')
  2     2                              NAME    BCDIVR
  3     3                      ;*****************************************************
  4     4                      ;*      decimal division                             *
  5     5                      ;*           8 digit <- 8 digit / 8 digit            *
  6     6                      ;*       input condition                             *
  7     7                      ;*             dividend <- (DIVIND+3,...,DIVIND)     *
  8     8                      ;*             divisor  <- (DIVSOR+3,...,DIVSOR)     *
  9     9                      ;*       output condition                            *
 10    10                      ;*             quotient  <- (DIVIND+3,...,DIVIND)    *
 11    11                      ;*             remainder <- (RMIND+3,...,RMIND)      *
 12    12                      ;*                                                   *
 13    13                      ;*              RSS <- 0                             *
 14    14                      ;*****************************************************
 15    15                              PUBLIC  BCDDIV
 16    16                              EXTRN   ERROR,RCLR,BCDLS,BCDSUB,BCDADD
 17    17                              EXTRN   DIVIND,DIVSOR,RMIND
 18    18        (0000.0)      SF_QUO  EQU     X.0
 19    19        (0000.1)      SF_REM  EQU     X.1
 20    20                      ;
 21    21 ----                         CSEG
 22    22                              RSS     0
 23    23 0000                 BCDDIV:
 24    24                      ;
 25    25                      ;       ****  check / divisor = 0 ?  ****
 26    26                      ;
 27    27 0000  BA04                   MOV     C,#4            ; C-register <- 4
 28    28 0002  B900                   MOV     A,#0            ; Acc <- 0
 29    29 0004 R670000                 MOVW    HL,#DIVSOR      ; HL <- DIVSOR
 30    30 0007                 BCDDV1:
 31    31 0007  169F                   CMP     [HL+],A
 32    32 0009  8005                   BNZ     $BCDDV2         ; (HL) = 0 ?
 33    33 000B  32FA                   DBNZ    C,$BCDDV1
 34    34                      ;
 35    35 000D R2C0000                 BR      !ERROR          ; OVER FLOW
 36    36                      ;
 37    37                      ;       ****  result,remind 0-clear  ****
 38    38                      ;
 39    39 0010                 BCDDV2:
 40    40 0010 R650000                 MOVW    DE,#RMIND       ; DE <- RMIND
 41    41 0013 R280000                 CALL    !RCLR
```

```
42    42                    ;
43    43                    ;       ****  check / sign  ****
44    44                    ;
45    45 0016 R670300               MOVW    RP7,#DIVSOR+3
46    46 0019  0390                 CLR1    SF_QUO          ; clear quotient sign-flag
47    47 001B  0391                 CLR1    SF_REM          ; clear remainder sign-flag
48    48 001D R09F00300             MOV     A,!DIVIND+3
49    49 0021  03AF0A               BF      A.7,$BCDDV3
50    50 0024  039F                 CLR1    A.7
51    51 0026 R09F10300             MOV     !DIVIND+3,A
52    52 002A  0381                 SET1    SF_REM          ; set remainder sign-flag
53    53 002C  0370                 NOT1    SF_QUO          ; not quotient sign-flag
54    54 002E               BCDDV3:
55    55 002E R09F00300             MOV     A,!DIVSOR+3
56    56 0032  03AF08               BF      A.7,$BCDDV4
57    57 0035  039F                 CLR1    A.7
58    58 0037 R09F10300             MOV     !DIVSOR+3,A
59    59 003B  0370                 NOT1    SF_QUO
60    60                    ;
61    61                    ;       ****  digit counter set  ****
62    62                    ;
63    63 003D               BCDDV4:
64    64 003D  BF08                 MOV     R7,#8
65    65                    ;
66    66                    ;       ****  quotient,remind left shift  ****
67    67                    ;
68    68 003F               BCDDV5:
69    69 003F R670000               MOVW    HL,#DIVIND      ; HL <- DIVIND
70    70 0042  BA08                 MOV     C,#16/2         ; C-register <- 8
71    71 0044 R280000               CALL    !BCDLS          ; N-digit data left shift
72    72                    ;
73    73                    ;       ****  subtract divisor from dividend  ****
74    74                    ;
75    75 0047               BCDDV6:
76    76 0047 R650000               MOVW    DE,#DIVSOR      ; DE <- DIVSOR
77    77 004A R670000               MOVW    HL,#RMIND       ; HL <- RMIND
78    78 004D R280000               CALL    !BCDSUB         ; decimal subtraction
79    79                    ;
80    80 0050 R09F00300             MOV     A,!RMIND+3
81    81 0054  03BF09               BT      A.7,$BCDDV7     ; if borrow then goto BCDDV7
82    82                    ;
83    83 0057  B901                 MOV     A,#1
84    84 0059 R670000               MOVW    HL,#DIVIND
85    85 005C  16D8                 ADD     [HL],A          ; increment (DIVIND)
86    86                    ;
87    87 005E  14E7                 BR      $BCDDV6
```

```
 88   88                    ;
 89   89                    ;       ****  if borrow then divisor + dividend  ****
 90   90                    ;
 91   91 0060               BCDDV7:
 92   92 0060 R650000               MOVW    DE,#DIVSOR      ; DE <- DIVSOR
 93   93 0063 R670000               MOVW    HL,#RMIND       ; HL <- RMIND
 94   94 0066 R280000               CALL    !BCDADD         ; decimal addition
 95   95                    ;
 96   96                    ;       ****  check / division end ?  ****
 97   97                    ;
 98   98 0069  CF                   DEC     R7
 99   99 006A  80D3                 BNZ     $BCDDV5
100  100                    ;
101  101                    ;       ****  check / quotient = 0  ****
102  102                    ;
103  103 006C R670000              MOVW    HL,#DIVIND
104  104 006F  B900                MOV     A,#0
105  105 0071  BA04                MOV     C,#4
106  106 0073               BCDDV8:
107  107 0073  169F                CMP     [HL+],A
108  108 0075  8004                BNZ     $BCDDV9
109  109 0077  32FA                DBNZ    C,$BCDDV8
110  110 0079  140D                BR      $BCDV10
111  111                    ;
112  112                    ;       ****  check / quotient sign-flag  ****
113  113                    ;
114  114 007B               BCDDV9:
115  115 007B  03A00A              BF      SF_QUO,$BCDV10
116  116 007E R09F00300            MOV     A,!DIVIND+3
117  117 0082  038F                SET1    A.7
118  118 0084 R09F10300            MOV     !DIVIND+3,A
119  119                    ;
120  120                    ;       ****  check / remainder = 0  ****
121  121                    ;
122  122 0088               BCDV10:
123  123 0088 R670000              MOVW    HL,#RMIND
124  124 008B  B900                MOV     A,#0
125  125 008D  BA04                MOV     C,#4
126  126 008F               BCDV11:
127  127 008F  169F                CMP     [HL+],A
128  128 0091  8003                BNZ     $BCDV12
129  129 0093  32FA                DBNZ    C,$BCDV11
130  130 0095  56                  RET
131  131                    ;
132  132                    ;       ****  check / remainder sign-flag  ****
133  133                    ;
134  134 0096               BCDV12:
135  135 0096  03A10A              BF      SF_REM,$BCDV13
136  136 0099 R09F00300            MOV     A,!RMIND+3
137  137 009D  038F                SET1    A.7
138  138 009F R09F10300            MOV     !RMIND+3,A
```

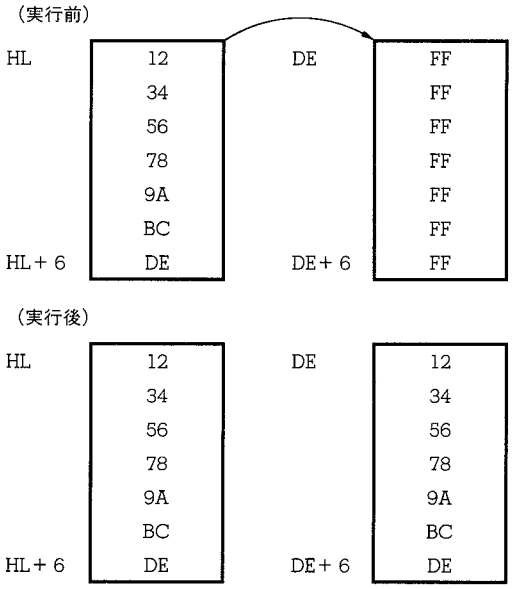
```
  139   139 00A3                BCDV13:
  140   140 00A3  56                    RET
  141   141                             ENDS
  142   142                             END

Segment informations:

ADRS  LEN    NAME

0000  00A4H  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG        ?CSEG       21#
BCDADD   ----H   E         EXT             16@    94
BCDDIV     0H    R ADDR    PUB  ?CSEG       15@    23#
BCDDV1     7H    R ADDR         ?CSEG       30#    33
BCDDV2    10H    R ADDR         ?CSEG       32     39#
BCDDV3    2EH    R ADDR         ?CSEG       49     54#
BCDDV4    3DH    R ADDR         ?CSEG       56     63#
BCDDV5    3FH    R ADDR         ?CSEG       68#    99
BCDDV6    47H    R ADDR         ?CSEG       75#    87
BCDDV7    60H    R ADDR         ?CSEG       81     91#
BCDDV8    73H    R ADDR         ?CSEG      106#   109
BCDDV9    7BH    R ADDR         ?CSEG      108    114#
BCDIVR             MOD                       2#
BCDLS    ----H   E         EXT             16@    71
BCDSUB   ----H   E         EXT             16@    78
BCDV10     88H    R ADDR         ?CSEG      110    115    122#
BCDV11     8FH    R ADDR         ?CSEG      126#   129
BCDV12     96H    R ADDR         ?CSEG      128    134#
BCDV13     A3H    R ADDR         ?CSEG      135    139#
DIVIND   ----H   E         EXT             17@    48     51     69     84    103    116    11
8
DIVSOR   ----H   E         EXT             17@    29     45     55     58     76     92
ERROR    ----H   E         EXT             16@    35
RCLR     ----H   E         EXT             16@    41
RMIND    ----H   E         EXT             17@    40     77     80     93    123    136    13
8
SF_QUO     0H.0   RBIT                      18#    46     53     59    115
SF_REM     0H.1   RBIT                      19#    47     52    135

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

3.3　データ転送

ここでは，メモリ内に設定されたデータのエリア間の転送について，「任意エリアから任意エリアへの転送」の例を紹介します。

例　図3－1のようにメモリ内でHL～HL＋6で指し示すメモリ内の7バイト・データをDE～DE＋6で指定するメモリ内へ転送する。

図3－1　データ転送図



μPD78320では，ストリング命令（MOVBK）を用いて，メモリ間のNバイト転送を行うことができます。

なお，実行前に，転送元と転送先の最下位番地または最上位番地をHLレジスタとDEレジスタに，転送バイト数をCレジスタに設定しておく必要があります。

ただし，転送バイト数は256以下とします。

このプログラム例では，HLレジスタに1000Hを，DEレジスタに1100Hを，Cレジスタに8をそれぞれ設定しておきます。

ステップ数　12バイト

```
     Assemble list

 ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

    1     1                      $       TITLE   ('data transmission')
    2     2                              NAME    TRANSR
    3     3                      ;*************************************************
    4     4                      ;*      data transmission                        *
    5     5                      ;*************************************************
    6     6
    7     7                      ;=================================================
    8     8                      ;       data table area
    9     9                      ;=================================================
   10    10
   11    11       (1000)        ADDRS1  EQU     1000H
   12    12       (2000)        ADDRS2  EQU     2000H
   13    13
   14    14                      ;=================================================
   15    15                      ;               data transmission routine
   16    16                      ;=================================================
   17    17 ----                        CSEG
   18    18                             RSS     0
   19    19                      ;
   20    20 0000  670010                MOVW    HL,#ADDRS1
   21    21 0003  650020                MOVW    DE,#ADDRS2
   22    22                      ;
   23    23 0006  BA07                  MOV     C,#7            ; counter set
   24    24
   25    25 0008  1520                  MOVBK   [DE+],[HL+]
   26    26
   27    27 000A                DTL1:
   28    28 000A  14FE                  BR      $DTL1
   29    29                             ENDS
   30    30                             END
   31    31

Segment informations:

ADRS  LEN    NAME

0000  000CH  ?CSEG
```

```
          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG       ?CSEG       17#
ADDRS1   1000H     NUM                    11#    20
ADDRS2   2000H     NUM                    12#    21
DTL1       AH    R ADDR       ?CSEG       27#    28
TRANSR             MOD                     2#

Target chip:uPD78320
Assembly complete,      0 error(s) and     0 warning(s) found. (     0)
```

3.4　シフト処理

μPD78320はレジスタ（R0，R1，……，R15），およびレジスタ・ペア（RP0，RP1，……RP7）の1ビット単位シフト命令（ROR，ROL，etc）と4ビット単位シフト命令（ROR4，ROL4）の2種のシフト命令を持っています。

本プログラム例では，次の2種類のシフト例を示します。

○バイト単位のシフト

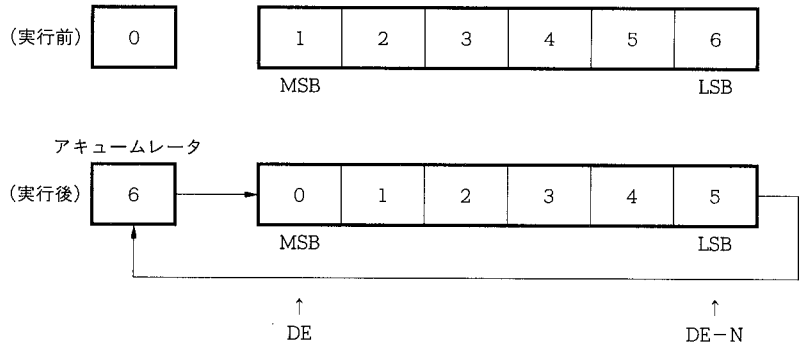
○4ビット単位のシフト

3.4.1　Nバイト・データの右シフト

メモリ内にあるNバイト・データの右シフトを行う例です。

このプログラムの実行後は，下図のように，アキュームレータにあらかじめ設定しておいた値を最上位の1バイトに格納し，最下位の1バイトの内容をアキュームレータに出力します。

(1) 使用メモリ・エリア



(2) 使用レジスタ

A，C，D，E

(3) 入力条件

この処理に必要な設定を，次のレジスタにします。

アキュームレータ←最上位メモリに転送する値

Cレジスタ←バイト数（N）

DEレジスタ←Nバイト・データの最上位番地

(4) 出力条件

1バイト右シフトした結果を，結果エリアに格納します。

最上位メモリにはアキュームレータの内容を転送します。

アキュームレータ←LSBの内容

(5) ステップ数

5バイト

Assemble list

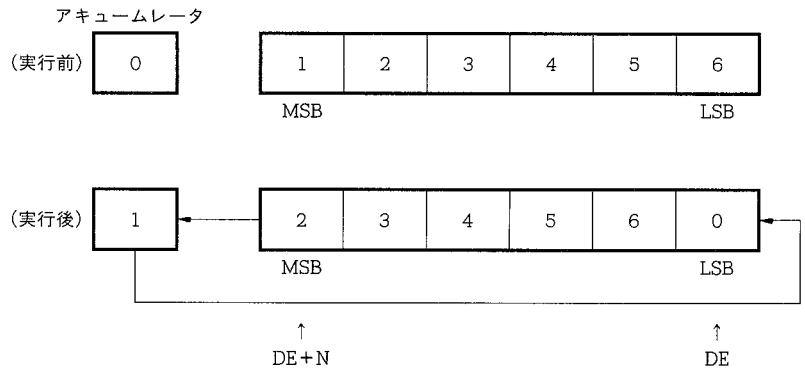
```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                     $       TITLE   ('N-byte data left shift')
   2     2                             NAME    BYTRSR
   3     3                     ;*******************************************************
   4     4                     ;*      N-byte data right shift                        *
   5     5                     ;*          input condition                             *
   6     6                     ;*              HL-register <- MSD of N-digit data      *
   7     7                     ;*              C -register <- digit counter            *
   8     8                     ;*          output condition                            *
   9     9                     ;*              Acc <- LSD of N-digit data              *
  10    10                     ;*                                                      *
  11    11                     ;*              RSS <- 0                                *
  12    12                     ;*******************************************************
  13    13                             PUBLIC  BYTRST,BYTRS1
  14    14                             PUBLIC  BYTLST,BYTLS1
  15    15                     ;
  16    16 ----                        CSEG
  17    17                             RSS     0
  18    18 0000                BYTRST:
  19    19 0000  B900                  MOV     A,#0            ; Acc <- 0
  20    20 0002                BYTRS1:
  21    21 0002  1511                  XCHM    [DE-],A
  22    22                     ;
  23    23 0004  56                    RET
  24    24                             ENDS
```

3.4.2　Nバイト・データの左シフト

メモリ内にあるNバイト・データの左シフトを行う例を示します。

このプログラムの実行後は，下図のように，アキュームレータにあらかじめ設定しておいた値を最上位の1バイトに格納し，最上位の1バイトの内容をアキュームレータに出力します。

(1) 使用メモリ・エリア



(2) 使用レジスタ

A，C，D，E

(3) 入力条件

この処理に必要な設定を，次のレジスタにします。

アキュームレータ←最下位メモリに転送する値

Cレジスタ←バイト数（N）

DEレジスタ←Nバイト・データの最下位番地

(4) 出力条件

1バイト左シフトした結果を，結果エリアに格納します。

最下位メモリにはアキュームレータの内容を転送します。

アキュームレータ←MSBの内容

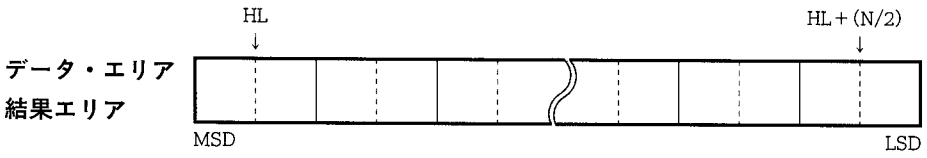
(5) ステップ数

5バイト

```
25    25
26    26                    ;*************************************************
27    27                    ;*      N-byte data left shift                   *
28    28                    ;*          input condition                      *
29    29                    ;*              DE-register <- LSB of N-byte data *
30    30                    ;*              C -register <- byte counter       *
31    31                    ;*          output condition                     *
32    32                    ;*              Acc <- MSB of N-byte data        *
33    33                    ;*                                               *
34    34                    ;*              RSS <- 0                         *
35    35                    ;*************************************************
36    36 ----                       CSEG
37    37                            RSS     0
38    38 0005               BYTLST:
39    39 0005  B900                 MOV     A,#0            ; Acc <- 0
40    40 0007               BYTLS1:
41    41 0007  1501                 XCHM    [DE+],A
42    42                    ;
43    43 0009  56                   RET
44    44                            ENDS
45    45                            END
```

3.4.3　10進N桁データの1桁右シフト

(1) 使用メモリ・エリア



(2) 使用レジスタ

A, C, H, L

(3) 入力条件

(1)のように，この処理に必要な設定を次のレジスタにします。

HLレジスタ・ペア←N桁データの最上位のアドレス

Cレジスタ←バイト数（N/2）

(4) 出力条件

1桁右シフトした結果を，(1)の結果エリアに格納します。

アキュームレータ←LSDの内容

MSDには0を格納します。

(5) ステップ数

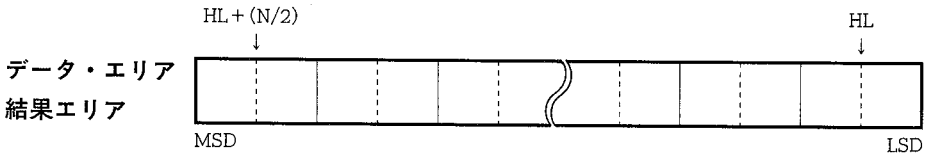
8バイト

Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                     $       TITLE   ('N-digit shift')
   2     2                             NAME    BCDRSR
   3     3                     ;*****************************************************
   4     4                     ;*      N-digit data right shift                     *
   5     5                     ;*          input condition                          *
   6     6                     ;*              HL-register <- MSD of N-digit data   *
   7     7                     ;*              C -register <- digit counter         *
   8     8                     ;*          output condition                         *
   9     9                     ;*              Acc <- LSD of N-digit data           *
  10    10                     ;*                                                   *
  11    11                     ;*              RSS <- 0                             *
  12    12                     ;*****************************************************
  13    13                             PUBLIC  BCDRS,BCDRS1
  14    14                             PUBLIC  BCDLS,BCDLS1
  15    15                     ;
  16    16 ----                        CSEG
  17    17                             RSS     0
  18    18 0000                BCDRS:
  19    19 0000  B900                  MOV     A,#0            ; Acc <- 0
  20    20 0002                BCDRS1:
  21    21 0002  058F                  ROR4    [HL]
  22    22 0004  4F                    DECW    HL              ; decrement (HL)
  23    23 0005  32FB                  DBNZ    C,$BCDRS1
  24    24                     ;
  25    25 0007  56                    RET
  26    26                             ENDS
```

3.4.4　10進N桁データの1桁左シフト

(1) 使用メモリ・エリア



(2) 使用レジスタ

A, C, H, L

(3) 入力条件

(1)のように，この処理に必要な設定を次のレジスタにします。

HLレジスタ・ペア←N桁データの最下位のアドレス

Cレジスタ←バイト数（N/2）

(4) 出力条件

1桁左シフトした結果を(1)の結果エリアに格納します。

アキュームレータ←MSDの内容

LSDには0を格納します。

(5) ステップ数

8バイト

```
27    27
28    28                    ;*****************************************************
29    29                    ;*      N-digit data left shift                      *
30    30                    ;*          input condition                          *
31    31                    ;*              HL-register <- LSD of N-digit data   *
32    32                    ;*              C -register <- digit counter         *
33    33                    ;*          output condition                         *
34    34                    ;*              Acc <- MSD of N-digit data           *
35    35                    ;*                                                   *
36    36                    ;*              RSS <- 0                             *
37    37                    ;*****************************************************
38    38 ----                       CSEG
39    39                            RSS     0
40    40 0008               BCDLS:
41    41 0008  B900                 MOV     A,#0
42    42 000A               BCDLS1:
43    43 000A  059F                 ROL4    [HL]
44    44 000C  47                   INCW    HL              ; increment (HL)
45    45 000D  32FB                 DBNZ    C,$BCDLS1
46    46                    ;
47    47 000F  56                   RET
48    48                            ENDS
49    49                            END
```

保守／廃止

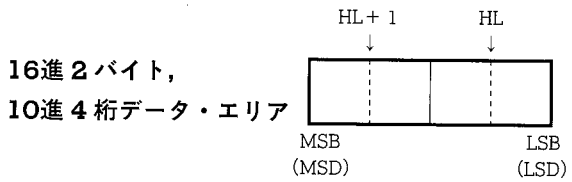
3.5　データ変換処理

ここでは数値データの表現形式を，16進と10進，16進とASCIIの各形式間で変換するプログラムについて説明します。

3.5.1　16進（HEX）を10進（BCD）に変換

16進２バイトのデータを10進４桁へ変換します。

（1）使用メモリ・エリア



（2）使用レジスタ

X，A，C，B，R4，R5，R6，D，E，H，L

（3）入力条件

(1)のように，次の設定をします。

HLレジスタ・ペア←入力データ16進２バイトが格納してあるエリアのLSDのアドレス

（4）出力条件

CY＝１：16進データが270FH（＝9999）より大きいため，変換できません。

CY＝０：変換した10進４桁を（HL，HL＋1）に格納します。

（5）処理手順

この変換サブルーチンでは，10進４桁（２バイト）の変換値を最上位桁から１桁ずつ求めています。

４桁目は1000，３桁目は100，２桁目は10，１桁目は１と除数を設定し，16進入力データを各除数で割ったときの商を変換値として，最上位桁から１桁ずつ求めていきます。

① 除数を10000Dに設定する。

② 16進入力データと除数を比較し，16進入力データが10000以上であれば２バイトで表せないので，変換不能として終了する。

③ 桁カウンタ（Bレジスタ）を４に設定する。

④ 除数を10にする。

⑤　16進入力データを除数で割る。

⑥　変換値格納エリアを１桁左シフトし，最下位桁に商を格納する。

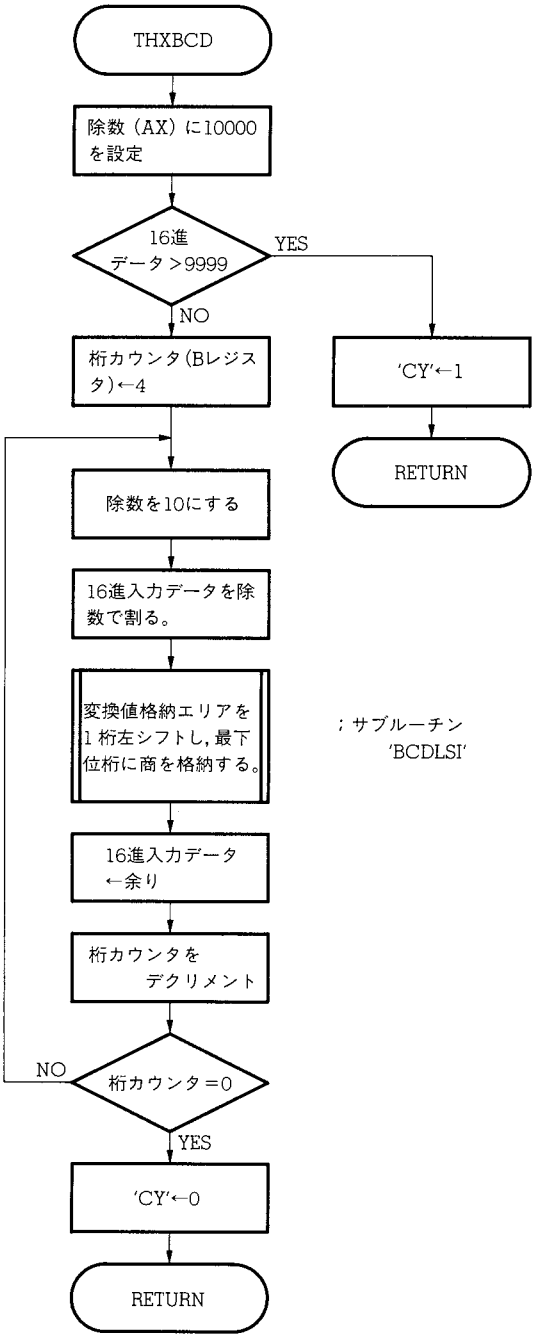
⑦　余りを16進入力データとする。

⑧　桁カウンタをデクリメントし，０になるまで④から⑦を繰り返す。

(6) ステップ数

47バイト

フロー・チャート



保守／廃止

Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                      $       TITLE   ('transform BCD <- HEX')
   2     2                              NAME    TRBCDR
   3     3                      ;*************************************************
   4     4                      ;*      transform BCD <- HEX                     *
   5     5                      ;*         input condition                       *
   6     6                      ;*              HL-register <- HEX-2byte data    *
   7     7                      ;*                                   LSB address *
   8     8                      ;*         output condition                      *
   9     9                      ;*              normal ... cy = 0                *
  10    10                      ;*                  decimal 4-digit -> (HL,HL+1) *
  11    11                      ;*              overflow ... cy = 1              *
  12    12                      ;*                  HEX data > 9999              *
  13    13                      ;*                                               *
  14    14                      ;*              RSS <- 0                         *
  15    15                      ;*************************************************
  16    16                              PUBLIC  THXBCD
  17    17                              EXTRN   BCDLS1
  18    18                      ;
  19    19 ----                         CSEG
  20    20                              RSS     0
  21    21 0000                 THXBCD:
  22    22 0000  1651                   MOVW    AX,[HL]         ; DE <- hex data
  23    23 0002  24C8                   MOVW    DE,AX
  24    24 0004  601027                 MOVW    AX,#10000
  25    25                      ;
  26    26 0007  8FC8                   CMPW    DE,AX
  27    27 0009  8302                   BC      $THXBD1
  28    28 000B  41                     SET1    CY
  29    29 000C  56                     RET
  30    30                      ;
  31    31 000D                 THXBD1:
  32    32 000D  248F                   MOVW    VP,HL           ; save HL
  33    33                      ;
```

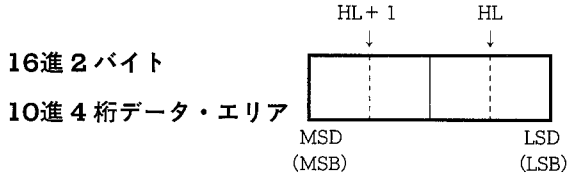
```
 34    34                      ;       ****  digit counter set  ****
 35    35                      ;
 36    36 000F  BB04                   MOV     B,#4
 37    37                      ;
 38    38 0011                 THXBD2:
 39    39 0011  BE0A                   MOV     R6,#10
 40    40 0013  051E                   DIVUW   R6              ; AX <- AX / 10
 41    41 0015  24A8                   MOVW    UP,AX           ; save AX
 42    42                      ;
 43    43                              RSS     1
 44    44 0017  43                     SWRS
 45    45                      ;
 46    46 0018  640000                 MOVW    AX,#0           ; AX <- 0
 47    47 001B  05E8                   DIVUX   RP0             ; AXDE <- AXDE / RP0
 48    48                                                      ; RP0  <- remainder
 49    49 001D  25C5                   XCH     E,A             ; A <- E
 50    50 001F  24E9                   MOVW    HL,VP
 51    51 0021  BE02                   MOV     C,#2            ; C <- 2
 52    52 0023 R280000                 CALL    !BCDLS1         ; digit left shift
 53    53                      ;
 54    54                              RSS     0
 55    55 0026  43                     SWRS
 56    56                      ;
 57    57 0027  24C8                   MOVW    DE,AX           ; DE <- remainder
 58    58 0029  240B                   MOVW    AX,UP           ; load AX
 59    59 002B  33E4                   DBNZ    B,$THXBD2
 60    60                      ;
 61    61 002D  40                     CLR1    CY
 62    62 002E  56                     RET
 63    63
 64    64                              ENDS
 65    65                              END

Segment informations:

ADRS  LEN    NAME

0000  002FH  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG       ?CSEG       19#
BCDLS1   ----H   E          EXT           17@    52
THXBCD     0H    R ADDR  PUB  ?CSEG       16@    21#
THXBD1     DH    R ADDR       ?CSEG       27     31#
THXBD2    11H    R ADDR       ?CSEG       38#    59
TRBCDR           MOD                       2#

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

保守／廃止

3.5.2　10進（BCD）を16進（HEX）に変換

10進４桁のデータを16進２バイトに変換します。

(1) 使用メモリ・エリア



(2) 使用レジスタ

X，A，C，B，R5，R6，D，E，H，L

(3) 入力条件

(1)のように，次の設定をします。

HLレジスタ・ペア←入力データ10進４桁が格納してあるエリアのLSDのアドレス

(4) 出力条件

CY＝１：入力データが10進でないので変換できません。

CY＝０：変換した16進２バイトを（HL，HL＋1）に格納します。

(5) 処理手順

変換値格納エリアを０クリアし，10進入力データを最上位桁から１桁左シフトにより，１桁ずつアキュームレータへロードし，次の操作を４回繰り返して，変換します。

格納エリア←格納エリア×10＋アキュームレータ

① 変換値格納レジスタ（DEレジスタ）を０クリアしておく。

② 桁カウンタ（Bレジスタ）を４に設定する。

③ 10進入力データを１桁左シフトし，最上位桁をアキュームレータへロードする。

④ 最上位桁が10進データ(0～9)かをチェックする。10進でなければ，変換不可能として，CYをセットする。

⑤ 変換値格納レジスタ←変換値格納レジスタ×10＋アキュームレータを行う。

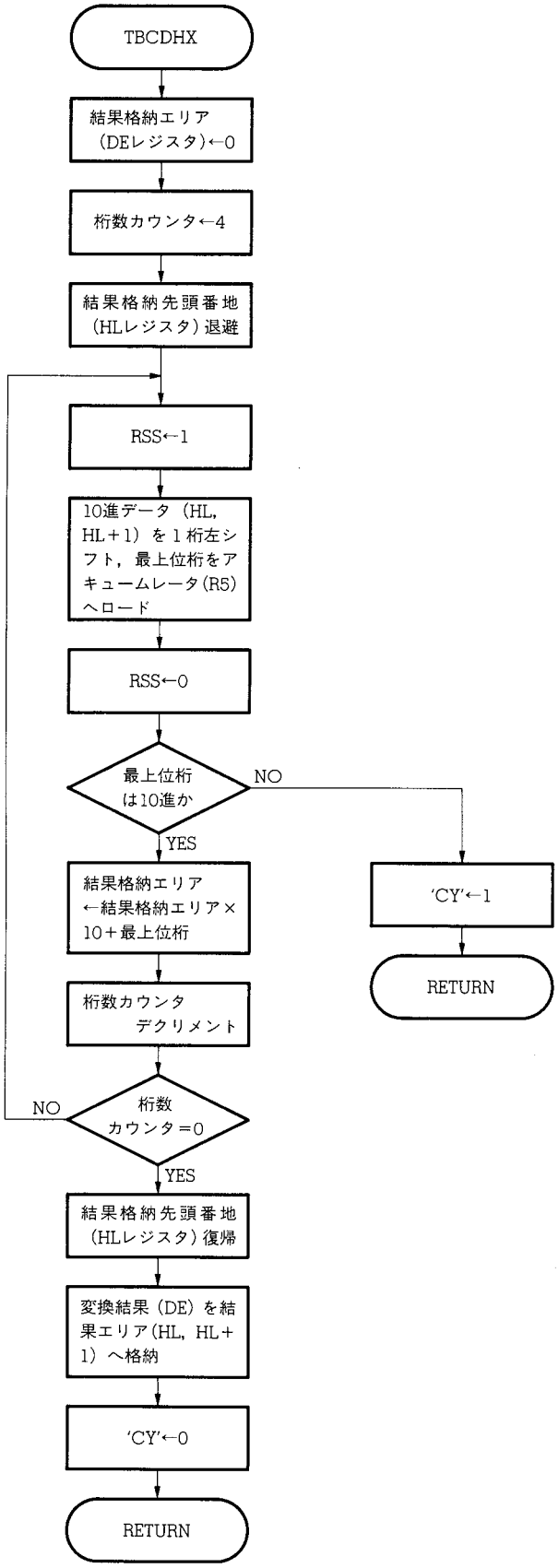
⑥ 桁カウンタをデクリメントし，０になるまで③から⑤を繰り返す。

⑦ 変換値格納レジスタの内容を変換結果エリアに格納し，CYをリセットする。

(6) ステップ数

45バイト

フロー・チャート



保守／廃止

Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                      $       TITLE   ('transform HEX <- BCD')
   2     2                              NAME    TRHEXR
   3     3                      ;*************************************************************
   4     4                      ;*      transform HEX <- BCD                                 *
   5     5                      ;*          input condition                                  *
   6     6                      ;*              HL-register <- decimal 4 digit data          *
   7     7                      ;*                                      LSD address          *
   8     8                      ;*          output condition                                 *
   9     9                      ;*              normal ... cy = 0                            *
  10    10                      ;*                    HEX 2 byte -> (HL,HL+1)                *
  11    11                      ;*              error  ... cy = 1                            *
  12    12                      ;*                                                           *
  13    13                      ;*              RSS <- 0                                     *
  14    14                      ;*************************************************************
  15    15                              PUBLIC  TBCDHX
  16    16                              EXTRN   BCDLS1
  17    17                      ;
  18    18 ----                         CSEG
  19    19                              RSS     0
  20    20 0000                 TBCDHX:
  21    21 0000  650000                 MOVW    DE,#0           ; DE <- 0
  22    22                      ;
  23    23                      ;       ****  digit counter set  ****
  24    24                      ;
  25    25 0003  BB04                   MOV     B,#4
  26    26                      ;
  27    27 0005  248F                   MOVW    VP,HL           ; save HL
  28    28                      ;
  29    29 0007                 TBDHX1:
  30    30                              RSS     1
  31    31 0007  43                     SWRS
  32    32                      ;
  33    33                      ;       ****  output 1 digit from MSD  ****
  34    34                      ;
  35    35 0008  BE02                   MOV     C,#2            ; C-register <- 2
  36    36 000A  BD00                   MOV     A,#0            ; Acc <- 0
  37    37 000C  24E9                   MOVW    HL,VP
  38    38 000E R280000                 CALL    !BCDLS1         ; N-digit data left shift
  39    39                      ;
  40    40                      ;       ****  check / BCD display ?  ****
  41    41                      ;
  42    42 0011  AF0A                   CMP     A,#10
  43    43                              RSS     0
  44    44 0013  43                     SWRS
  45    45                      ;
  46    46 0014  8302                   BC      $TBDHX2
  47    47                      ;
```

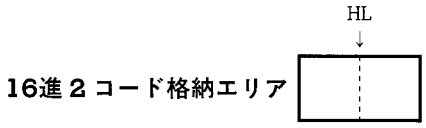
```
48    48 0016  41                    SET1    CY              ; not BCD
49    49 0017  56                    RET
50    50                     ;
51    51                     ;       ****  subroutine / HEX <- BCD  ****
52    52                     ;
53    53 0018                TBDHX2:
54    54 0018  BC00                  MOV     R4,#0
55    55 001A  2545                  XCH     R4,R5
56    56                     ;
57    57                     ;       ****  result * 10 + 1 digit  ****
58    58                     ;
59    59 001C  600A00                MOVW    AX,#10
60    60 001F  052D                  MULUW   DE
61    61 0021  88CC                  ADDW    DE,RP2
62    62                     ;
63    63 0023  33E2                  DBNZ    B,$TBDHX1
64    64 0025  24E9                  MOVW    HL,VP
65    65                     ;
66    66                     ;       ****  store result  ****
67    67                     ;
68    68 0027  25C8                  XCHW    DE,AX
69    69 0029  16D1                  MOVW    [HL],AX
70    70 002B  40                    CLR1    CY
71    71 002C  56                    RET
72    72
73    73                             ENDS
74    74                             END

Segment informations:

ADRS  LEN    NAME

0000  002DH  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG       ?CSEG       18#
BCDLS1    ----H    E        EXT           16@    38
TBCDHX      0H     R ADDR   PUB  ?CSEG      15@    20#
TBDHX1      7H     R ADDR        ?CSEG      29#    63
TBDHX2     18H     R ADDR        ?CSEG      46     53#
TRHEXR             MOD                      2#

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

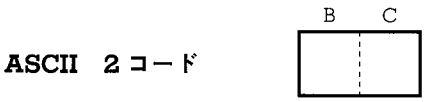
保守／廃止

3.5.3　ASCIIを16進（HEX）に変換

ASCII　2コード（30H－39H，41H－46H）をHEX　2コード（00H－FFH）に変換します。

(1) 使用メモリ・エリア





(2) 使用レジスタ

A，B，C，H，L

(3) 入力条件

(1)のように，次の設定をします。

BCレジスタ←ASCII　2コード

HLレジスタ←16進2コード格納エリアの番地

(4) 出力条件

CY＝1：入力データがASCIIコードでないので変換できません。

CY＝0：変換した16進2コードを(1)の格納エリアに格納します。

(5) 処理手順

① ASCII上位1コード（Bレジスタ）をアキュームレータにロードする。

② アキュームレータの内容が30H－39H，41H－46Hの範囲にあるかチェックする。なければ変換不能としてCYをセットする。

③ アキュームレータの内容が30H－39Hなら30Hを引く。

アキュームレータの内容が41H－46Hなら37Hを引く。

④ HLレジスタの指定する番地の内容を4ビットシフトし，下位ビットにアキュームレータの内容を格納する。

⑤ ASCII下位1コード（Cレジスタ）をアキュームレータにロードし，②から④の処理を行う。

(6) ステップ数

41バイト

Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                     $       TITLE   ('HEX <-  ASCII')
   2     2                             NAME    GHEXR
   3     3                     ;*************************************************
   4     4                     ;*      transform   HEX <-  ASCII                *
   5     5                     ;*                    (2code)  (2code)           *
   6     6                     ;*                                               *
   7     7                     ;*              input  condition                 *
   8     8                     ;*                BC-register <- ASCII           *
   9     9                     ;*                                               *
  10    10                     ;*              output condition                 *
  11    11                     ;*                (HL)  <-  hex                  *
  12    12                     ;*************************************************
  13    13                             PUBLIC  GETHEX
  14    14                             PUBLIC  SHEX
  15    15                     ;
  16    16 ----                        CSEG
  17    17                             RSS     0
  18    18 0000                GETHEX:
  19    19 0000  D3                    MOV     A,B             ; ASCII upper-code load
  20    20 0001 R281100                CALL    !SHEX           ; get  hex 1th code
  21    21 0004  830A                  BC      $GTHEX1
  22    22                     ;
  23    23 0006  059F                  ROL4    [HL]
  24    24 0008  D2                    MOV     A,C             ; ASCII lower-code load
  25    25 0009 R281100                CALL    !SHEX           ; get  hex 2th code
  26    26 000C  8302                  BC      $GTHEX1
  27    27 000E  059F                  ROL4    [HL]
  28    28                     ;
  29    29 0010                GTHEX1:
  30    30 0010  56                    RET
  31    31
  32    32                             ENDS
  33    33
  34    34                     ;*************************************************
  35    35                     ;*      subroutine  /  get hex 1-code(Acc)   *
  36    36                     ;*************************************************
  37    37 ----                        CSEG
  38    38                             RSS     0
  39    39 0011                SHEX:
  40    40 0011  AF30                  CMP     A,#'0'          ; check / ASCII > 30H
  41    41 0013  8312                  BC      $SHEX2
  42    42                     ;
  43    43 0015  AF39                  CMP     A,#'9'          ; check / ASCII > 39H
  44    44 0017  8203                  BNC     $SHEX1
  45    45 0019  AA30                  SUB     A,#30H
  46    46 001B  56                    RET
  47    47                     ;
```

保守／廃止

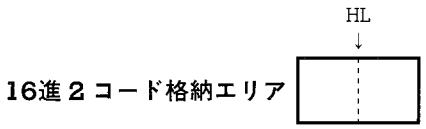
```
   48    48 001C                SHEX1:
   49    49 001C  AF41                  CMP     A,#'A'          ; check / ASCII > 41H
   50    50 001E  8307                  BC      $SHEX2
   51    51                     ;
   52    52 0020  AF46                  CMP     A,#'F'          ; check / ASCII < 46H
   53    53 0022  8203                  BNC     $SHEX2
   54    54 0024  AA37                  SUB     A,#37H
   55    55 0026  56                    RET
   56    56                     ;
   57    57 0027                SHEX2:
   58    58 0027  41                    SET1    CY              ; error
   59    59 0028  56                    RET
   60    60
   61    61                             ENDS
   62    62                             END

Segment informations:

ADRS  LEN    NAME

0000  0029H  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG               CSEG       ?CSEG       16#    37#
GETHEX      0H    R ADDR  PUB  ?CSEG       13@    18#
GHEXR               MOD                     2#
GTHEX1     10H    R ADDR       ?CSEG       21     26     29#
SHEX       11H    R ADDR  PUB  ?CSEG       14@    20     25     39#
SHEX1      1CH    R ADDR       ?CSEG       44     48#
SHEX2      27H    R ADDR       ?CSEG       41     50     53     57#

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

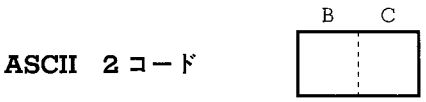
保守／廃止

3.5.4　16進（HEX）をASCIIに変換

HEX　2コード（00H－FFH）をASCII　2コード（30H－39H，41H－46H）に変換します。

(1) 使用メモリ・エリア





(2) 使用レジスタ

A，B，C，H，L

(3) 入力条件

(1)のように，次の設定をします。

HLレジスタ←16進2コード格納エリアの番地

(4) 出力条件

BCレジスタ←変換したASCII　2コードを格納します。

(5) 処理手順

① HLレジスタの指定する番地の上位4ビットをアキュームレータにロードする。

② アキュームレータが9以下かチェックする。9以下ならアキュームレータに37Hを加える。9以上ならアキュームレータに30Hを加える。

③ アキュームレータをBレジスタにロードする。

④ HLレジスタの指定する番地の下位4ビットをアキュームレータにロードする。

⑤ ②の処理を行い，アキュームレータの内容をCレジスタにロードする。

(6) ステップ数

28バイト

Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                      $       TITLE   ('ASCII  <-  HEX')
   2     2                              NAME    ASCII
   3     3                      ;*************************************************
   4     4                      ;*      transform    ASCII  <-  HEX              *
   5     5                      ;*                   (2code)    (2code)           *
   6     6                      ;*                                                *
   7     7                      ;*              input  condition                  *
   8     8                      ;*                 (HL) <- hex 2-code             *
   9     9                      ;*              output condition                  *
  10    10                      ;*                 BC-register <- ASCII 2-code    *
  11    11                      ;*************************************************
  12    12                              PUBLIC  GETASC
  13    13                              PUBLIC  SASC
  14    14                      ;
  15    15 ----                         CSEG
  16    16                              RSS     0
  17    17 0000                 GETASC:
  18    18 0000  B900                   MOV     A,#0
  19    19 0002  059F                   ROL4    [HL]            ; hex upper code load
  20    20 0004 R281300                 CALL    !SASC
  21    21 0007  2431                   MOV     B,A             ; store result
  22    22                      ;
  23    23 0009  B900                   MOV     A,#0
  24    24 000B  059F                   ROL4    [HL]            ; hex lower code load
  25    25 000D R281300                 CALL    !SASC
  26    26 0010  2421                   MOV     C,A             ; store result
  27    27 0012  56                     RET
  28    28
  29    29                              ENDS
  30    30
  31    31                      ;*******************************************************
  32    32                      ;*      subroutine  /  get ASCII 1-code(BC-register)   *
  33    33                      ;*******************************************************
  34    34 ----                         CSEG
  35    35                              RSS     0
  36    36 0013                 SASC:
  37    37 0013  AF0A                   CMP     A,#0AH          ; check / hex > 9
  38    38 0015  8302                   BC      $SASC1
  39    39 0017  A807                   ADD     A,#07H          ; bias (+7)
  40    40 0019                 SASC1:
  41    41 0019  A830                   ADD     A,#30H          ; bias (+30H)
  42    42 001B  56                     RET
  43    43
  44    44                              ENDS
  45    45                              END
```

```
Segment informations:

ADRS  LEN    NAME

0000  001CH  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG       ?CSEG       15#    34#
ASCII              MOD                     2#
GETASC      0H   R ADDR  PUB  ?CSEG       12@    17#
SASC       13H   R ADDR  PUB  ?CSEG       13@    20     25     36#
SASC1      19H   R ADDR       ?CSEG       38     40#

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

保守／廃止

3.6　比較処理

ここは，次に示す３種類の処理について説明します。

・２つの数値データ間の大小関係をチェックし，その結果によって分岐する処理

・データ検索処理

・メモリ上に設けたソフトウエア・フラグのビット単位のセット・リセット処理，テスト処理

3.6.1　16ビット（２バイト）データ比較

ここでは，次の３種類の比較処理を行います。

＝；EQUAL

＜；GREATER THAN

＞；LESS THAN

上記の３種の処理に対して，μPD78320では，CMPW rp，rp1という16ビット演算命令があり，レジスタ・ペア同士の比較ができます。

なお，比較した結果は，ZフラグおよびCYフラグに表し，条件分岐命令によりテストし，任意の番地へ分岐します。

ステップ数　30バイト

Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                     $       TITLE   ('data comparison')
   2     2                             NAME    DCOMPR
   3     3                     ;*************************************************************
   4     4                     ;*      16bit(2byte) data comparison                         *
   5     5                     ;*************************************************************
   6     6 ----                        CSEG
   7     7                             RSS     0
   8     8
   9     9 0000                DCOMP:
  10    10 0000  650200                MOVW    DE,#2
  11    11 0003  670300                MOVW    HL,#3
  12    12
  13    13 0006  8FCF                  CMPW    DE,HL
  14    14 0008  8108                  BZ      $ADRS1
  15    15 000A  830C                  BC      $ADRS2
  16    16
  17    17                     ;=============================================================
  18    18                     ;               RP > RP1
  19    19                     ;=============================================================
  20    20
  21    21 000C                DCL1:
  22    22 000C  00                    NOP
  23    23 000D  00                    NOP
  24    24 000E  00                    NOP
  25    25 000F  00                    NOP
  26    26 0010  14FA                  BR      $DCL1
  27    27
  28    28                     ;=============================================================
  29    29                     ;               RP = RP1
  30    30                     ;=============================================================
  31    31
  32    32 0012                ADRS1:
  33    33 0012  00                    NOP
  34    34 0013  00                    NOP
  35    35 0014  00                    NOP
  36    36 0015  00                    NOP
  37    37 0016  14FA                  BR      $ADRS1
```

保守／廃止

```
  38    38
  39    39                    ;=======================================================
  40    40                    ;               RP < RP1
  41    41                    ;=======================================================
  42    42
  43    43 0018               ADRS2:
  44    44 0018  00                   NOP
  45    45 0019  00                   NOP
  46    46 001A  00                   NOP
  47    47 001B  00                   NOP
  48    48 001C  14FA                 BR      $ADRS2
  49    49
  50    50                            ENDS
  51    51                            END

Segment informations:

ADRS  LEN    NAME

0000  001EH  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG       ?CSEG       6#
ADRS1      12H   R ADDR       ?CSEG      14     32#    37
ADRS2      18H   R ADDR       ?CSEG      15     43#    48
DCL1        CH   R ADDR       ?CSEG      21#    26
DCOMP       0H   R ADDR       ?CSEG       9#
DCOMPR             MOD                   2#

Target chip:uPD78320
Assembly complete,      0 error(s) and      0 warning(s) found. (     0)
```

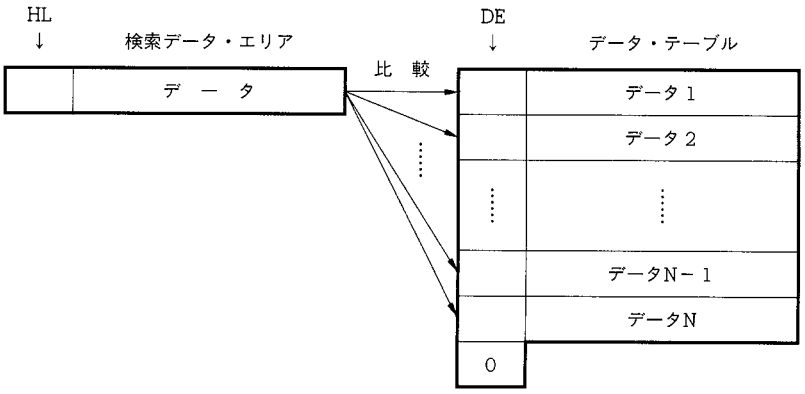
備考1. rp/rp1：RPn（n＝0－7）

2. オペランドに記述されるrpとrp1の対象となるレジスタ・ペアは，まったく同じですが実際に生成されるオブジェクト・プログラムでは，rpとrp1とで異なります。

3.6.2　データ検索

ここでは，図３-２のようにメモリ内の検索データをあらかじめ設定されたデータ・テーブルから検索する処理について説明します。

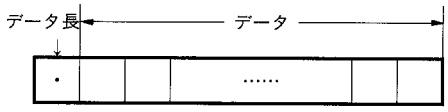
図３-２　データ検索説明



データ長については検索データ，データ・テーブル間で統一されている場合と統一されていない場合がありますが，ここでは各データのデータ長は不定とします。そのため，検索する際，データ長，データの内容の順にチェックが行えるように，データ形式を図３-３のように設定します。

また，データ・テーブルのターミネーションとして，図３-３のデータ形式の先頭番地（データ長）に０が入っていることとします。

図３-３　データ形式

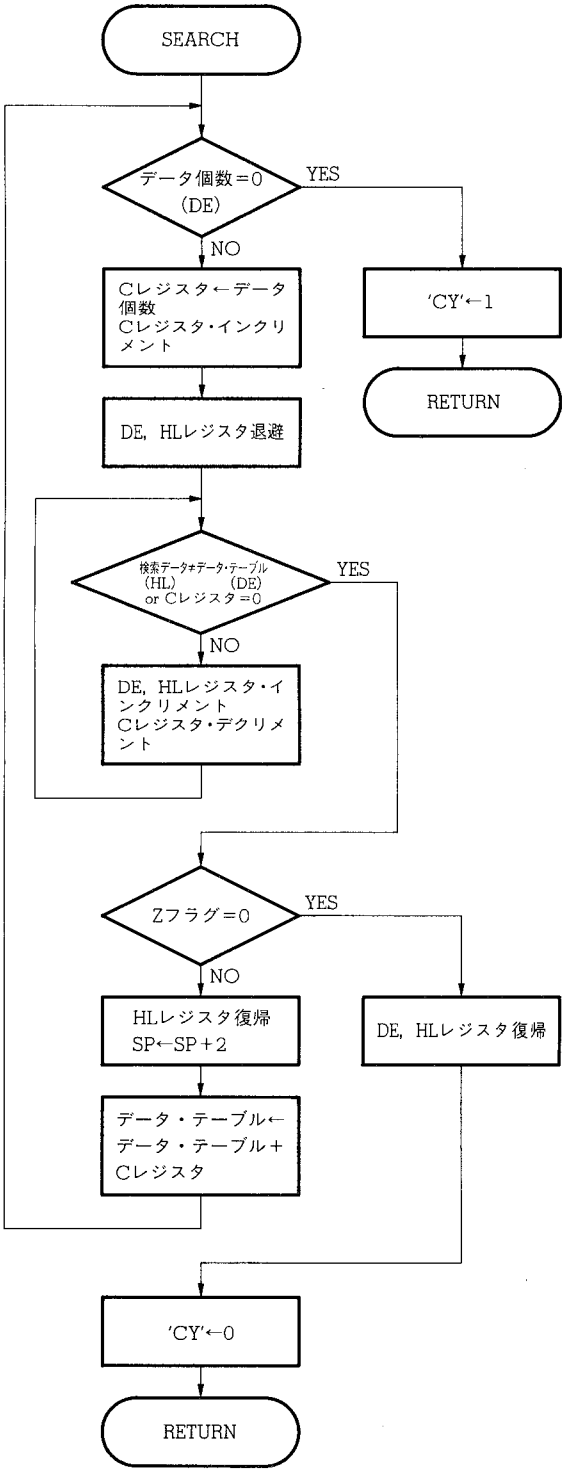


なお，データ検索処理を行う際，検索データ・エリアの最下位番地，データ・テーブルの最下位番地をそれぞれ，HL，DEレジスタで設定しておく必要があります。

また，データ・テーブルより検知した場合，そのデータ・テーブルの最下位番地をDEレジスタで出力します。

ステップ数　93バイト

フロー・チャート



Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

  1     1                     $       TITLE   ('data searching')
  2     2                             NAME    SEARC
  3     3                     ;*************************************************************
  4     4                     ;*      data searching                                       *
  5     5                     ;*       input  condition                                    *
  6     6                     ;*              HL-register <- searh data address            *
  7     7                     ;*              DE-register <- table top address             *
  8     8                     ;*       output condition                                    *
  9     9                     ;*              cy = 1                                       *
 10    10                     ;*                not search data                            *
 11    11                     ;*              cy = 0                                       *
 12    12                     ;*                DE-register <- search table address        *
 13    13                     ;*                                                           *
 14    14                     ;*                        RSS <- 0                           *
 15    15                     ;*************************************************************
 16    16                             PUBLIC  SEARCH
 17    17                     ;
 18    18 ----                        CSEG
 19    19                             RSS     0
 20    20 0000                SEARCH:
 21    21 0000  620000                MOVW    BC,#0
 22    22 0003  5C                    MOV     A,[DE]          ; load data-length
 23    23 0004  AF01                  CMP     A,#1            ; check / length = 0
 24    24 0006  8317                  BC      $SERCH2
 25    25                     ;
 26    26 0008  2421                  MOV     C,A             ; store length
 27    27 000A  C2                    INC     C
 28    28 000B  35C0                  PUSH    DE,HL
 29    29                     ;
 30    30 000D  1524                  CMPBKE  [DE+],[HL+]
 31    31 000F  8004                  BNZ     $SERCH1
 32    32 0011  34C0                  POP     DE,HL
 33    33 0013  40                    CLR1    CY              ; search success
 34    34 0014  56                    RET
 35    35                     ;
 36    36 0015                SERCH1:
 37    37 0015  05C8                  INCW    SP
 38    38 0017  05C8                  INCW    SP
 39    39 0019  3480                  POP     HL
 40    40 001B  88CA                  ADDW    DE,BC           ; DE <- next-table deta address
 41    41 001D  14E1                  BR      $SEARCH
 42    42                     ;
```

```
43    43 001F                SERCH2:
44    44 001F  41                    SET1    CY
45    45 0020  56                    RET
46    46
47    47                             ENDS
48    48
49    49                     $       EJECT
50    50
51    51                     ;*********************************************************
52    52                     ;*      data searching initilize                         *
53    53                     ;*********************************************************
54    54
55    55                     ;=========================================================
56    56                     ;               data table
57    57                     ;=========================================================
58    58 ----                        CSEG
59    59                             RSS     0
60    60 0021                STABLE:
61    61 0021  03123478              DB      03,12H,34H,78H
62    62 0025  04556677              DB      04,55H,66H,77H,88H
         0029  88
63    63 002A  05123456              DB      05,12H,34H,56H,78H,10H
         002E  7810
64    64 0030  03123456              DB      03,12H,34H,56H
65    65 0034  0412340A              DB      04,12H,34H,0AH,78H
         0038  78
66    66 0039  04123456              DB      04,12H,34H,56H,70H
         003D  70
67    67 003E  04123456              DB      04,12H,34H,56H,78H
         0042  78
68    68 0043  01AB                  DB      01,0ABH
69    69 0045  023478                DB      02,34H,78H
70    70 0048  00                    DB      00
71    71
72    72                     ;=========================================================
73    73                     ;               searching data
74    74                     ;=========================================================
75    75
76    76 0049                SDATA:
77    77 0049  04123456              DB      04,12H,34H,56H,78H
         004D  78
```

```
   78    78
   79    79                    ;=======================================================
   80    80                    ;               main routine
   81    81                    ;=======================================================
   82    82
   83    83 004E               SMAIN:
   84    84 004E R652100               MOVW    DE,#STABLE
   85    85 0051 R674900               MOVW    HL,#SDATA
   86    86
   87    87 0054 R280000               CALL    !SEARCH
   88    88 0057  8302                 BC      $SERR
   89    89 0059               SL1:
   90    90 0059  14FE                 BR      $SL1
   91    91
   92    92 005B               SERR:
   93    93 005B  14FE                 BR      $SERR
   94    94
   95    95                            ENDS
   96    96                            END

Segment informations:

ADRS  LEN    NAME

0000  005DH  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG             CSEG        ?CSEG       18#    58#
SDATA      49H    R ADDR       ?CSEG       76#    85
SEARC             MOD                      2#
SEARCH      0H    R ADDR  PUB  ?CSEG       16@    20#    41     87
SERCH1     15H    R ADDR       ?CSEG       31     36#
SERCH2     1FH    R ADDR       ?CSEG       24     43#
SERR       5BH    R ADDR       ?CSEG       88     92#    93
SL1        59H    R ADDR       ?CSEG       89#    90
SMAIN      4EH    R ADDR       ?CSEG       83#
STABLE     21H    R ADDR       ?CSEG       60#    84

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

保守／廃止

3.6.3　メモリ内のビット・テストとセット／リセット

(1) テストするビットが複数の場合

メモリとアキュームレータの内容とを比較（AND）し，Zフラグによってビット・テストを行います。

```
例1.  MOV R1, #00000011B   ; R1=A
      AND  A, saddr        ; Test A and Memory
      BNZ  $LABEL1
```

例1では，メモリのビット0，1の少なくとも一方が“1”（ANDの結果≠0）のとき，LABEL1番地へ分岐します。

(2) テストするビットが1個の場合

テストするビットが1個の場合，ビット判定命令（BT）を用いることができます。

```
例2.  BT saddr.1, $LABEL2
```

例2では，メモリのビット1が“1”のとき，LABEL2番地へ分岐します。

```
例3.  MOV A, mem
      BT   A.2, $LABEL3
```

例3では，メモリのビット2が“1”のとき，LABEL3番地へ分岐します。

(3) セット／リセットするビットが複数の場合

ショート・ダイレクト・メモリの内容とイミーディエト・データのOR，またはANDを取り，ビット単位にセット／リセットします。

```
例4.  OR saddr, #00000011B
  5.  AND saddr, #00001111B
```

例4では，saddrと“00000011”とのORを取りビット0，1をセットします。
例5では，saddrと“00001111”とのANDを取りビット4－7をリセットします。

次に，memとアキュームレータとのOR，またはANDを取り，ビット単位にセット／リセットします。

例6. MOV R1, #00000011B

OR mem, A（または AND mem, A）

メモリと“00000011”とのOR（またはAND）を取り，ビット０，１をセット（リセット）します。

(4) セット／リセットするビットが１個の場合

セット／リセットするビットが１個の場合，ビット操作命令を用いることができます。

例7. SET1　saddr.1

8. CLR1　saddr.1

例７では，saddrのビット１をセットします。

例８では，saddrのビット１をリセットします。

ステップ数　7Oバイト

Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                      $       TITLE   ('memory test')
   2     2                              NAME    BTESTR
   3     3                      ;***********************************************
   4     4                      ;*      memory bit test                        *
   5     5                      ;***********************************************
   6     6
   7     7                      ;===============================================
   8     8                      ;               test data
   9     9                      ;===============================================
  10    10                              EXTRN   SADD1, SADD2, SADD3
  11    11                              EXTRN   SADD4, SADD5, SADD6
  12    12                      ;
  13    13 ----                         CSEG
  14    14                              RSS     0
  15    15 0000                 BTEST1:
  16    16 0000  03                     DB      00000011B
  17    17 0001                 BTEST2:
  18    18 0001  02                     DB      00000010B
  19    19
  20    20                      ;===============================================
  21    21                      ;               plural bit test
  22    22                      ;===============================================
  23    23                      ;
  24    24                      ;       ****  mem  ****
  25    25                      ;
  26    26 0002 R670000                 MOVW    HL, #BTEST1
  27    27                      ;
  28    28 0005  B903                   MOV     A, #00000011B
  29    29 0007  165C                   AND     A, [HL]
  30    30 0009  800A                   BNZ     $LABEL1
  31    31 000B                 BTL1:
  32    32 000B  14FE                   BR      $BTL1
  33    33                      ;
  34    34                      ;       ****  saddr  ****
  35    35                      ;
  36    36 000D  B903                   MOV     A, #00000011B
  37    37 000F R9C00                   AND     A, SADD1
  38    38 0011  8002                   BNZ     $LABEL1
  39    39 0013                 BTL2:
  40    40 0013  14FE                   BR      $BTL2
  41    41
  42    42 0015                 LABEL1:
  43    43 0015  14FE                   BR      $LABEL1
  44    44
  45    45                      $       EJECT
```

```
46    46
47    47                    ;================================================
48    48                    ;               singular number
49    49                    ;================================================
50    50
51    51 0017 R71000B               BT      SADD2.1,$LABEL2
52    52
53    53 001A               SNL1:
54    54 001A  14FE                 BR      $SNL1
55    55
56    56 001C R670100               MOVW    HL,#BTEST2
57    57
58    58 001F  5D                   MOV     A,[HL]
59    59 0020  03BA02               BT      A.2,$LABEL2
60    60
61    61 0023               SNL2:
62    62 0023  14FE                 BR      $SNL2
63    63
64    64 0025               LABEL2:
65    65 0025  14FE                 BR      $LABEL2
66    66
67    67                    $       EJECT
68    68
69    69                    ;************************************************
70    70                    ;*      bit set/reset                           *
71    71                    ;************************************************
72    72
73    73                    ;================================================
74    74                    ;               bit set
75    75                    ;================================================
76    76
77    77 0027 R6E0003               OR      SADD3,#00000011B
78    78
79    79                    ;================================================
80    80                    ;               bit reset
81    81                    ;================================================
82    82
83    83 002A R6C000F               AND     SADD4,#00001111B
84    84
85    85                    ;================================================
86    86                    ;               set/reset data
87    87                    ;================================================
88    88
89    89 002D               SRDATA:
90    90 002D  02                   DB      00000010B
```

```
  91    91
  92    92                    ;================================================
  93    93                    ;               bit set
  94    94                    ;================================================
  95    95
  96    96 002E R672D00               MOVW    HL,#SRDATA
  97    97
  98    98 0031  B903                 MOV     A,#00000011B
  99    99
 100   100 0033  16DE                 OR      [HL],A
 101   101
 102   102 0035               SRL1:
 103   103 0035  14FE                 BR      $SRL1
 104   104
 105   105                    ;================================================
 106   106                    ;               bit reset
 107   107                    ;================================================
 108   108
 109   109 0037 R672D00               MOVW    HL,#SRDATA
 110   110
 111   111 003A  B903                 MOV     A,#00000011B
 112   112 003C  16DC                 AND     [HL],A
 113   113
 114   114 003E               SRL2:
 115   115 003E  14FE                 BR      $SRL2
 116   116
 117   117                    ;================================================
 118   118                    ;               singular number set/reset
 119   119                    ;================================================
 120   120
 121   121 0040 RB100                 SET1    SADD5.1
 122   122
 123   123 0042 RA100                 CLR1    SADD6.1
 124   124
 125   125 0044               SRL3:
 126   126 0044  14FE                 BR      $SRL3
 127   127
 128   128                            ENDS
 129   129                            END

Segment informations:

ADRS  LEN    NAME

0000  0046H  ?CSEG
```

```
          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG             CSEG       ?CSEG       13#
BTEST1      0H    R ADDR       ?CSEG       15#    26
BTEST2      1H    R ADDR       ?CSEG       17#    56
BTESTR            MOD                       2#
BTL1        BH    R ADDR       ?CSEG       31#    32
BTL2       13H    R ADDR       ?CSEG       39#    40
LABEL1     15H    R ADDR       ?CSEG       30     38     42#    43
LABEL2     25H    R ADDR       ?CSEG       51     59     64#    65
SADD1     ----H   E       EXT              10@    37
SADD2     ----H   E       EXT              10@    51
SADD3     ----H   E       EXT              10@    77
SADD4     ----H   E       EXT              11@    83
SADD5     ----H   E       EXT              11@   121
SADD6     ----H   E       EXT              11@   123
SNL1       1AH    R ADDR       ?CSEG       53#    54
SNL2       23H    R ADDR       ?CSEG       61#    62
SRDATA     2DH    R ADDR       ?CSEG       89#    96    109
SRL1       35H    R ADDR       ?CSEG      102#   103
SRL2       3EH    R ADDR       ?CSEG      114#   115
SRL3       44H    R ADDR       ?CSEG      125#   126

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

保守／廃止

3.7　テーブル参照処理

ここではテーブル参照の応用例として，補間計算法を紹介します。

図３－４のように関数f(X)と区間〔0，260〕上の26個の等間隔の点X

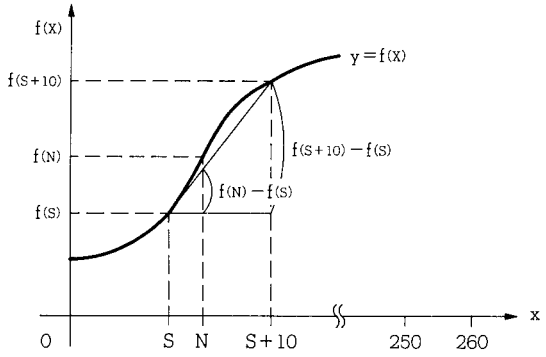
(X＝0，10，20，……，260)

におけるf(X)の値

(f(0)，f(10)，f(20)，……，f(260))

が与えられているとき，区間〔0，255〕での任意の点Nにおけるf(N)の値を補間計算法によって求めます。

図３－４　関数f(x)



まずNが26個の等間隔点で分けられた区間のどこにあるのかを次式で求めます。

$$S=INT(N/10)*10 \qquad \cdots\cdots(1)$$

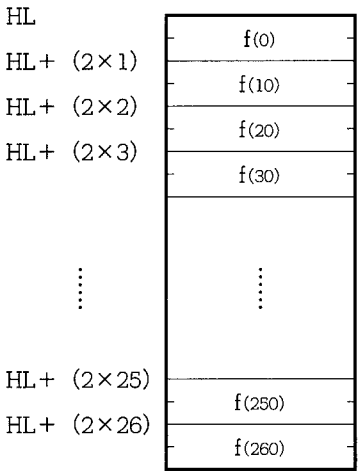
図３－４のような，相似関係により次式が成り立ちます。

$$f(N)-f(S) : f(S+10)-f(S)=N-S : S+10-S$$

$$\therefore f(N)=f(S)+\frac{(f(S+10)-f(S))\ (N-S)}{10} \qquad \cdots\cdots(2)$$

したがって，任意の点Nにおけるf(N)の値を求めることができます。

図3－5　データ・テーブル



次に実際の補間計算によるプログラム例について説明します。

図3－5のようにデータ・テーブルの先頭番地をHLレジスタで設定しておきます。

アキュムレータにより，点Nを与えて，$f_{(N)}$をメモリ上のデータ・テーブルにある$f_{(0)}$，$f_{(10)}$，$f_{(20)}$，……，$f_{(260)}$の値（2バイト・データ）から補間計算で求めます。

(1) 処理手順

① 式(1)よりSを求める

② データ・テーブル内の各データ（$f_{(0)}$，$f_{(10)}$，$f_{(20)}$，……，$f_{(260)}$）は2バイト単位のため求めたSを2倍する。

③ データ・テーブルの先頭番地に②で求めたSを加えることによって$f_{(S)}$が格納されるアドレスが得られる。

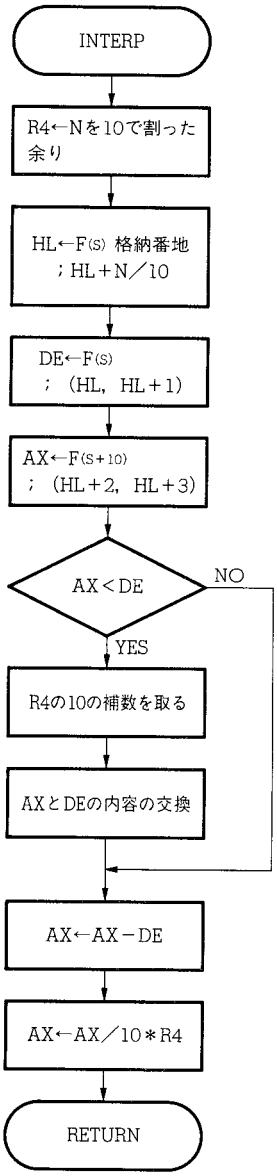
④ $f_{(S)}$，$f_{(S+10)}$を比較して$f_{(S)}$が大きければ$f_{(S)}$と$f_{(S+10)}$の内容を変換し，10－(N－S)を(N－S)とする。

⑤ 式(2)より$f_{(N)}$を求める。

(2) ステップ数

55バイト

フロー・チャート



Assemble list

```
ALNO  STNO ADRS  OBJECT   M I  SOURCE STATEMENT

   1     1                     $       TITLE   ('interpolation polynomial')
   2     2                             NAME    INTER
   3     3                     ;*************************************************
   4     4                     ;*      interpolation polynomial                 *
   5     5                     ;*              input  condition                 *
   6     6                     ;*                Acc <- N                       *
   7     7                     ;*                HL  <- data table top address *
   8     8                     ;*              output condition                 *
   9     9                     ;*                AX  <- result                  *
  10    10                     ;*   F(N) = F(S) + ( F(S+10)-F(S) ) * (N-S) /10 *
  11    11                     ;*                                               *
  12    12                     ;*              RSS <- 0                         *
  13    13                     ;*************************************************
  14    14                             PUBLIC  INTERP
  15    15                     ;
  16    16 ----                        CSEG
  17    17                             RSS     0
  18    18 0000                INTERP:
  19    19 0000  B800                  MOV     X,#0
  20    20 0002  BD00                  MOV     R5,#0
  21    21 0004  D8                    XCH     A,X             ; AX <- N
  22    22 0005  BC0A                  MOV     R4,#10
  23    23 0007  051C                  DIVUW   R4              ; AX <- AX / 10
  24    24 0009  31C8                  SHLW    AX,1            ; AX <- AX * 2
  25    25 000B  88E8                  ADDW    HL,AX           ; HL <- HL + AX
  26    26                     ;
  27    27 000D  1611                  MOVW    AX,[HL+]        ; DE <- (HL,HL+1)
  28    28 000F  24C8                  MOVW    DE,AX
  29    29                     ;
  30    30 0011  1651                  MOVW    AX,[HL]         ; AX <- (HL+2,HL+3)
  31    31 0013  4F                    DECW    HL
  32    32                     ;
  33    33 0014  8F0D                  CMPW    AX,DE
  34    34 0016  8209                  BNC     $INTER1
  35    35 0018  660A00                MOVW    RP3,#10         ; complement of RP2
  36    36 001B  8A6C                  SUBW    RP3,RP2
  37    37 001D  254E                  XCHW    RP2,RP3
  38    38                     ;
  39    39 001F  250D                  XCHW    AX,DE
```

```
   40    40 0021                INTER1:
   41    41 0021  8A0D                 SUBW    AX,DE           ; AX <- AX - DE
   42    42 0023  BE0A                 MOV     R6,#10
   43    43 0025  051E                 DIVUW   R6              ; AX,R6 <- AX / 10
   44    44                      ;
   45    45 0027  BF05                 MOV     R7,#5           ; getting the nearest
   46    46 0029  8F67                 CMP     R6,R7           ;       integral number
   47    47 002B  8303                 BC      $INTER2
   48    48 002D  2D0100               ADDW    AX,#1           ; increment AX
   49    49 0030                INTER2:
   50    50 0030  052C                 MULUW   RP2             ; AX,RP2 <- AX * RP2
   51    51 0032  884D                 ADDW    RP2,DE
   52    52 0034  240C                 MOVW    AX,RP2
   53    53                      ;
   54    54 0036  56                   RET
   55    55
   56    56                            ENDS
   57    57                            END

Segment informations:

ADRS  LEN    NAME

0000  0037H  ?CSEG

          Cross-Reference List

NAME      VALUE  R ATTR  RTYP SEGNAME  XREFS

?CSEG              CSEG       ?CSEG       16#
INTER              MOD                     2#
INTER1     21H   R ADDR       ?CSEG       34     40#
INTER2     30H   R ADDR       ?CSEG       47     49#
INTERP      0H   R ADDR  PUB  ?CSEG       14@    18#

Target chip:uPD78320
Assembly complete,     0 error(s) and     0 warning(s) found. (    0)
```

付録A　命令セット

ここでは，命令のオペレーションについてのみ説明しています。

命令の実行クロック数および，命令コードについては，各製品のユーザーズ・マニュアルを参照してください。

(1) オペランドの表現形式と記述方法

各命令のオペランド欄には，その命令のオペランド表現形式に対する記述方法に従ってオペランドを記述します(詳細は，アセンブラ仕様による)。記述方法の中で複数個あるものは，それらの要素の1つを選択します。大文字で書かれた英字および+，－，＃，＄，!，[ ]記号はキー・ワードであり，そのまま記述します。

イミーディエト・データの場合は，適当な数値またはレーベルを記述します。レーベルで記述する際も＃，＄，!，[ ]記号は必ず記述してください。

表A－1　オペランドの表現形式と記述方法

表現形式	記　　述　　方　　法
r	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15
r1	R0, R1, R2, R3, R4, R5, R6, R7
r2	C, B
rp	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp1	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp2	DE, HL, VP, UP
sfr	特殊機能レジスタ名
sfrp	特殊機能レジスタ名（16ビット操作可能レジスタ）
post	RP0, RP1, RP2, RP3, RP4, RP5/PSW, RP6, RP7 〔複数記述可能。ただし，RP5はPUSH，POP命令，PSWはPUSHU，POPU命令に限る。〕
mem	[DE], [HL], [DE+], [HL+], [DE−], [HL−], [VP], [UP]：レジスタ・インダイレクト・モード [DE+A], [HL+A], [DE+B], [HL+B], [VP+DE], [VP+HL]：ベースト・インデクスト・モード [DE+byte], [HL+byte], [VP+byte], [UP+byte], [SP+byte]：ベースト・モード word[A], word[B], word[DE], word[HL]：インデクスト・モード
saddr	FE20H-FF1FH　イミーディエト・データまたはレーベル
saddrp	FE20H-FF1EH　イミーディエト・データ（ただし，bit0＝0）またはレーベル（16ビット操作時）
$addr16	0000H-FDFFH　イミーディエト・データまたはレーベル：レラティブ・アドレシング
!addr16	0000H-FDFFH　イミーディエト・データまたはレーベル：イミーディエト・アドレシング （ただしMOV命令ではFFFFHまで記述可能）
addr11	800H-FFFH　イミーディエト・データまたはレーベル
addr5	40H-7EH　イミーディエト・データ（ただし，bit0＝0）[注]またはレーベル
word	16ビット・イミーディエト・データまたはレーベル
byte	8ビット・イミーディエト・データまたはレーベル
bit	3ビット・イミーディエト・データまたはレーベル
n	3ビット・イミーディエト・データ（0−7）

注　bit0＝1（奇数アドレス）へのワード・アクセスはしないでください。

備考1. rpとrp1は，記述できるレジスタ名は同じですが，発生するコードが異なります。

2. イミーディエト・アドレシングは全空間のアドレスが可能です。レラティブ・アドレシングは，次に続く命令の先頭アドレス－128から先頭アドレス＋127の範囲のみアドレスが可能です。

8ビット・レジスタの表現形式r，rlおよび16ビット・レジスタ・ペアの表現形式rp，rpl，postには，絶対名称（R0-R15，RP0-RP7）のほかに，機能名称で記述ができます。各絶対名称に対応する機能名称は表A－2，表A－3のようになります。

表A－2　8ビット・レジスタの絶対名称↔機能名称対応

絶対名称	機能名称		絶対名称	機能名称	
	RSS＝0	RSS＝1		RSS＝0	RSS＝1
R0	X		R8	VP_L	VP_L
R1	A		R9	VP_H	VP_H
R2	C		R10	UP_L	UP_L
R3	B		R11	UP_H	UP_H
R4		X	R12	E	E
R5		A	R13	D	D
R6		C	R14	L	L
R7		B	R15	H	H

表A－3　16ビット・レジスタ・ペアの絶対名称↔機能名称対応

絶対名称	機能名称	
	RSS＝0	RSS＝1
RP0	AX	
RP1	BC	
RP2		AX
RP3		BC
RP4	VP	VP
RP5	UP	UP
RP6	DE	DE
RP7	HL	HL

RSSはレジスタ・セット選択フラグ（PSWのビット5）で，そのセット/リセットにより絶対名称と機能名称の対応が切り替わります。

(2) オペレーション説明上の凡例

A	：Aレジスタ；8ビット・アキュームレータ
X	：Xレジスタ
B	：Bレジスタ
C	：Cレジスタ
D	：Dレジスタ
E	：Eレジスタ
H	：Hレジスタ
L	：Lレジスタ
R0-R15	：レジスタ0-レジスタ15(絶対名称)
AX	：レジスタ・ペア（AX)；16ビット・アキュームレータ
BC	：レジスタ・ペア（BC)
DE	：レジスタ・ペア（DE)
HL	：レジスタ・ペア（HL)
RP0-RP7	：レジスタ・ペア0-レジスタ・ペア7(絶対名称)
PC	：プログラム・カウンタ
SP	：スタック・ポインタ
UP	：ユーザ・スタック・ポインタ
PSW	：プログラム・ステータス・ワード
CY	：キャリー・フラグ
AC	：補助キャリー・フラグ
Z	：ゼロ・フラグ
P/V	：パリティ/オーバフロー・フラグ
S	：サイン・フラグ
TPF	：テーブル・ポジション・フラグ
RBS	：レジスタ・バンク・セレクト・フラグ
RSS	：レジスタ・セット・セレクト・フラグ
IE	：割り込み許可フラグ
STBC	：スタンバイ・コントロール・レジスタ
WDM	：ウォッチドッグ・タイマ・モード・レジスタ
jdisp8	：符号付き8ビット・データ（ディスプレースメント：−128〜+127)
(　)	：(　) 内のアドレスまたはレジスタの内容で示されるメモリの内容。 (　+), (　−)の場合は，命令実行後(　)内の内容が+1または−1される。
((　))	：((　)) 内のアドレスで示されるメモリの内容により示されるメモリの内容
××H	：16進数
$\times_H$, $\times_L$	：16ビット・レジスタの上位8ビット，下位8ビット

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
8ビット・データ転送	MOV	r1, #byte	2	r1←byte					
		saddr, #byte	3	(saddr) ←byte					
		sfr[注], #byte	3	sfr←byte					
		r, r1	2	r←r1					
		A, r1	1	A←r1					
		A, saddr	2	A← (saddr)					
		saddr, A	2	(saddr) ←A					
		saddr, saddr	3	(saddr) ← (saddr)					
		A, sfr	2	A←sfr					
		sfr, A	2	sfr←A					
		A, mem	1-4	A← (mem)					
		mem, A	1-4	(mem) ←A					
		A, [saddrp]	2	A← ((saddrp))					
		[saddrp], A	2	((saddrp)) ←A					
		A, !addr16	4	A← (addr16)					
		!addr16, A	4	(addr16)←A					
		PSWL, #byte	3	PSW_L←byte	×	×	×	×	×
		PSWH, #byte	3	PSW_H←byte					
		PSWL, A	2	PSW_L←A	×	×	×	×	×
		PSWH, A	2	PSW_H←A					
		A, PSWL	2	A←PSW_L					
		A, PSWH	2	A←PSW_H					
	XCH	A, r1	1	A↔r1					
		r, r1	2	r↔r1					
		A, mem	2-4	A↔ (mem)					
		A, saddr	2	A↔ (saddr)					
		A, sfr	3	A↔sfr					
		A, [saddrp]	2	A↔ ((saddrp))					
		saddr, saddr	3	(saddr)↔ (saddr)					

注　sfrにSTBC，WDMを記述した場合は別の専用命令となり，バイト数がこの命令とは異なります。

備考　フラグ動作欄の記号は次の表を参照してください。

記　号	説　明
（ブランク）	変化なし
0	0にクリアされる
1	1にセットされる
×	結果に従ってセット/クリアされる
P	P/Vフラグが，パリティ・フラグとして動作する
V	P/Vフラグが，オーバフロー・フラグとして動作する
R	以前に退避した値がリストアされる

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
16ビット・データ転送	MOVW	rp1, #word	3	rp1←word					
		saddrp, #word	4	(saddrp) ←word					
		sfrp, #word	4	sfrp←word					
		rp, rp1	2	rp←rp1					
		AX, saddrp	2	AX← (saddrp)					
		saddrp, AX	2	(saddrp) ←AX					
		saddrp, saddrp	3	(saddrp) ← (saddrp)					
		AX, sfrp	2	AX←sfrp					
		sfrp, AX	2	sfrp←AX					
		rp1, !addr16	4	rp1← (addr16)					
		!addr16, rp1	4	(addr16) ←rp1					
		AX, mem	2-4	AX←mem					
		mem, AX	2-4	mem←AX					
	XCHW	AX, saddrp	2	AX↔ (saddrp)					
		AX, sfrp	3	AX↔sfrp					
		saddrp, saddrp	3	(saddrp) ↔ (saddrp)					
		rp, rp1	2	rp↔rp1					
		AX, mem	2-4	AX↔mem					
8ビット演算	ADD	A, #byte	2	A, CY←A+byte	×	×	×	V	×
		saddr, #byte	3	(saddr) , CY← (saddr) +byte	×	×	×	V	×
		sfr, #byte	4	sfr, CY←sfr+byte	×	×	×	V	×
		r, rl	2	r, CY←r+rl	×	×	×	V	×
		A, saddr	2	A, CY←A+ (saddr)	×	×	×	V	×
		A, sfr	3	A, CY←A+sfr	×	×	×	V	×
		saddr, saddr	3	(saddr) , CY← (saddr) + (saddr)	×	×	×	V	×
		A, mem	2-4	A, CY←A+ (mem)	×	×	×	V	×
		mem, A	2-4	(mem) , CY← (mem) +A	×	×	×	V	×
	ADDC	A, #byte	2	A, CY←A+byte+CY	×	×	×	V	×
		saddr, #byte	3	(saddr) , CY← (saddr) +byte+CY	×	×	×	V	×
		sfr, #byte	4	sfr, CY←sfr+byte+CY	×	×	×	V	×
		r, rl	2	r, CY←r+rl+CY	×	×	×	V	×
		A, saddr	2	A, CY←A+ (saddr) +CY	×	×	×	V	×
		A, sfr	3	A, CY←A+sfr+CY	×	×	×	V	×
		saddr, saddr	3	(saddr), CY← (saddr)+ (saddr)+CY	×	×	×	V	×
		A, mem	2-4	A, CY←A+ (mem) +CY	×	×	×	V	×
		mem, A	2-4	(mem) , CY← (mem) +A+CY	×	×	×	V	×

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ				
					S	Z	AC	P/V	CY
8ビット演算	SUB	A, #byte	2	A, CY←A−byte	×	×	×	V	×
		saddr, #byte	3	(saddr), CY← (saddr) −byte	×	×	×	V	×
		sfr, #byte	4	sfr, CY←sfr−byte	×	×	×	V	×
		r, rl	2	r, CY←r−rl	×	×	×	V	×
		A, saddr	2	A, CY←A− (saddr)	×	×	×	V	×
		A, sfr	3	A, CY←A−sfr	×	×	×	V	×
		saddr, saddr	3	(saddr), CY← (saddr) − (saddr)	×	×	×	V	×
		A, mem	2-4	A, CY←A− (mem)	×	×	×	V	×
		mem, A	2-4	(mem), CY← (mem) −A	×	×	×	V	×
	SUBC	A, #byte	2	A, CY←A−byte−CY	×	×	×	V	×
		saddr, #byte	3	(saddr), CY← (saddr) −byte−CY	×	×	×	V	×
		sfr, #byte	4	sfr, CY←sfr−byte−CY	×	×	×	V	×
		r, rl	2	r, CY←r−rl−CY	×	×	×	V	×
		A, saddr	2	A, CY←A− (saddr) −CY	×	×	×	V	×
		A, sfr	3	A, CY←A−sfr−CY	×	×	×	V	×
		saddr, saddr	3	(saddr), CY← (saddr) − (saddr) −CY	×	×	×	V	×
		A, mem	2-4	A, CY←A− (mem) −CY	×	×	×	V	×
		mem, A	2-4	(mem), CY← (mem) −A−CY	×	×	×	V	×
	AND	A, #byte	2	A←A∧byte	×	×		P	
		saddr, #byte	3	(saddr) ← (saddr)∧byte	×	×		P	
		sfr, #byte	4	sfr←sfr∧byte	×	×		P	
		r, rl	2	r←r∧rl	×	×		P	
		A, saddr	2	A←A∧ (saddr)	×	×		P	
		A, sfr	3	A←A∧sfr	×	×		P	
		saddr, saddr	3	(saddr) ← (saddr)∧ (saddr)	×	×		P	
		A, mem	2-4	A←A∧ (mem)	×	×		P	
		mem, A	2-4	(mem) ← (mem) ∧A	×	×		P	
	OR	A, #byte	2	A←A∨byte	×	×		P	
		saddr, #byte	3	(saddr) ← (saddr) ∨byte	×	×		P	
		sfr, #byte	4	sfr←sfr∨byte	×	×		P	
		r, rl	2	r←r∨rl	×	×		P	
		A, saddr	2	A←A∨ (saddr)	×	×		P	
		A, sfr	3	A←A∨sfr	×	×		P	
		saddr, saddr	3	(saddr) ← (saddr) ∨ (saddr)	×	×		P	
		A, mem	2-4	A←A∨ (mem)	×	×		P	
		mem, A	2-4	(mem) ← (mem) ∨A	×	×		P	

保守／廃止

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ				
					S	Z	AC	P/V	CY
8ビット演算	**XOR**	A, #byte	2	A←A∀byte	×	×		P	
		saddr, #byte	3	(saddr) ← (saddr) ∀byte	×	×		P	
		sfr, #byte	4	sfr←sfr∀byte	×	×		P	
		r, r1	2	r←r∀r1	×	×		P	
		A, saddr	2	A←A∀ (saddr)	×	×		P	
		A, sfr	3	A←A∀sfr	×	×		P	
		saddr, saddr	3	(saddr) ← (saddr)∀ (saddr)	×	×		P	
		A, mem	2-4	A←A∀ (mem)	×	×		P	
		mem, A	2-4	(mem) ← (mem)∀A	×	×		P	
	CMP	A, #byte	2	A−byte	×	×	×	V	×
		saddr, #byte	3	(saddr) −byte	×	×	×	V	×
		sfr, #byte	4	sfr−byte	×	×	×	V	×
		r, r1	2	r−r1	×	×	×	V	×
		A, saddr	2	A− (saddr)	×	×	×	V	×
		A, sfr	3	A−sfr	×	×	×	V	×
		saddr, saddr	3	(saddr) − (saddr)	×	×	×	V	×
		A, mem	2-4	A− (mem)	×	×	×	V	×
		mem, A	2-4	(mem) −A	×	×	×	V	×

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
16ビット演算	ADDW	AX, #word	3	AX, CY←AX+word	×	×	×	V	×
		saddrp, #word	4	(saddrp), CY←(saddrp)+word	×	×	×	V	×
		sfrp, #word	5	sfrp, CY←sfrp+word	×	×	×	V	×
		rp, rp1	2	rp, CY←rp+rp1	×	×	×	V	×
		AX, saddrp	2	AX, CY←AX+(saddrp)	×	×	×	V	×
		AX, sfrp	3	AX, CY←AX+sfrp	×	×	×	V	×
		saddrp, saddrp	3	(saddrp), CY←(saddrp)+(saddrp)	×	×	×	V	×
	SUBW	AX, #word	3	AX, CY←AX−word	×	×	×	V	×
		saddrp, #word	4	(saddrp), CY←(saddrp)−word	×	×	×	V	×
		sfrp, #word	5	sfrp, CY←sfrp−word	×	×	×	V	×
		rp, rp1	2	rp, CY←rp−rp1	×	×	×	V	×
		AX, saddrp	2	AX, CY←AX−(saddrp)	×	×	×	V	×
		AX, sfrp	3	AX, CY←AX−sfrp	×	×	×	V	×
		saddrp, saddrp	3	(saddrp), CY←(saddrp)−(saddrp)	×	×	×	V	×
	CMPW	AX, #word	3	AX−word	×	×	×	V	×
		saddrp, #word	4	(saddrp)−word	×	×	×	V	×
		sfrp, #word	5	sfrp−word	×	×	×	V	×
		rp, rp1	2	rp−rp1	×	×	×	V	×
		AX, saddrp	2	AX−(saddrp)	×	×	×	V	×
		AX, sfrp	3	AX−sfrp	×	×	×	V	×
		saddrp, saddrp	3	(saddrp)−(saddrp)	×	×	×	V	×
乗除算	MULU	rl	2	AX←A×rl					
	DIVUW	rl	2	AX(商), rl(余り)←AX÷rl					
	MULUW	rp1	2	AX(上位16ビット), rp1(下位16ビット)←AX×rp1					
	DIVUX	rp1	2	AXDE(商), rp1(余り)←AXDE÷rp1					
乗算符号付き	MULW	rp1	2	AX(上位16ビット), rp1(下位16ビット)←AX×rp1					
積和演算	MACW[注1]	n	3	AXDE←(B)×(C)+AXDE B←B+2, C←C+2, n←n−1 End if n=0 or P/V=1	×	×	×	V	×

飽和付き積和演算	MACSW[注2]	n	3	AXDE←(B)×(C)+AXDE
				B←B+2, C←C+2, n←n−1
				if overflow (P/V=1) then
				AXDE←7FFFFFFFH
				if underflow (P/V=1) then
				AXDE←80000000H
				End if n=0 or P/V=1
					×	×	×	V	×

注1. μPD78352Aサブシリーズ，78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズのみ。

2. μPD78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズのみ。

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
相関演算	**SACW**注1	[DE+],[HL+]	4	AX←AX+|(DE)−(HL)| DE←DE+2, HL←HL+2, C←C−1 End if C=0 or CY=1	×	×	×	V	×
テーブル・シフト	**MOVTBLW**注2	!addr16, n	4	(addr16+2) ← (addr16), n←n−1 addr16←addr16−2, End if n=0					
増減	**INC**	r1	1	r1←r1+1	×	×	×	V	
		saddr	2	(saddr)← (saddr)+1	×	×	×	V	
	DEC	r1	1	r1←r1−1	×	×	×	V	
		saddr	2	(saddr) ← (saddr) −1	×	×	×	V	
	INCW	rp2	1	rp2←rp2+1					
		saddrp	3	(saddrp) ← (saddrp) +1					
	DECW	rp2	1	rp2←rp2−1					
		saddrp	3	(saddrp) ← (saddrp) −1					
シフト・ローテート	**ROR**	r1, n	2	(CY, $r1_7$←$r1_0$, $r1_{m-1}$←$r1_m$) ×n回				P	×
	ROL	r1, n	2	(CY, $r1_0$←$r1_7$, $r1_{m+1}$←$r1_m$) ×n回				P	×
	RORC	r1, n	2	(CY←$r1_0$, $r1_7$←CY, $r1_{m-1}$←$r1_m$) ×n回				P	×
	ROLC	r1, n	2	(CY←$r1_7$, $r1_0$←CY, $r1_{m+1}$←$r1_m$) ×n回				P	×
	SHR	r1, n	2	(CY←$r1_0$, $r1_7$←0, $r1_{m-1}$←$r1_m$) ×n回	×	×	0	P	×
	SHL	r1, n	2	(CY←$r1_7$, $r1_0$←0, $r1_{m+1}$←$r1_m$) ×n回	×	×	0	P	×
	SHRW	rp1, n	2	(CY←$rp1_0$, $rp1_{15}$←0, $rp1_{m-1}$←$rp1_m$) ×n回	×	×	0	P	×
	SHLW	rp1, n	2	(CY←$rp1_{15}$, $rp1_0$←0, $rp1_{m+1}$←$rp1_m$) ×n回	×	×	0	P	×
	ROR4	[rp1]	2	A_{3-0}← (rp1) $_{3-0}$, (rp1) $_{7-4}$←A_{3-0}, (rp1) $_{3-0}$← (rp1) $_{7-4}$					
	ROL4	[rp1]	2	A_{3-0}← (rp1) $_{7-4}$, (rp1) $_{3-0}$←A_{3-0}, (rp1) $_{7-4}$← (rp1) $_{3-0}$					
BCD補正	**ADJBA**		2	Decimal Adjust Accumulator	×	×	×	P	×
	ADJBS								
データ変換	**CVTBW**		1	A_7=0のとき X←A, A←00H A_7=1のとき X←A, A←FFH					

注1. μPD78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズのみ。

2. μPD78352Aサブシリーズ，78356サブシリーズ，78366Aサブシリーズ，78372サブシリーズのみ。

備考1. テーブル・シフト命令のアドレス範囲はFE00H-FEFFHです。

2. シフト・ローテート命令のnは，シフト・ローテート命令の回数を示します。

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
ビット操作	**XOR1**	CY, saddr. bit	3	CY←CY∀ (saddr. bit)					×
		CY, sfr. bit	3	CY←CY∀sfr. bit					×
		CY, A. bit	2	CY←CY∀A. bit					×
		CY, X. bit	2	CY←CY∀X. bit					×
		CY, PSWH. bit	2	CY←CY∀PSW_H. bit					×
		CY, PSWL. bit	2	CY←CY∀PSW_L. bit					×
	SET1	saddr. bit	2	(saddr. bit) ←1					
		sfr. bit	3	sfr. bit←1					
		A. bit	2	A. bit←1					
		X. bit	2	X. bit←1					
		PSWH. bit	2	PSW_H. bit←1					
		PSWL. bit	2	PSW_L. bit←1	×	×	×	×	×
	CLR1	saddr. bit	2	(saddr. bit) ←0					
		sfr. bit	3	sfr. bit←0					
		A. bit	2	A. bit←0					
		X. bit	2	X. bit←0					
		PSWH. bit	2	PSW_H. bit←0					
		PSWL. bit	2	PSW_L. bit←0	×	×	×	×	×
	NOT1	saddr. bit	3	(saddr. bit) ← $\overline{\text{(saddr. bit)}}$					
		sfr. bit	3	sfr. bit←$\overline{\text{sfr. bit}}$					
		A. bit	2	A. bit←$\overline{\text{A. bit}}$					
		X. bit	2	X. bit←$\overline{\text{X. bit}}$					
		PSWH. bit	2	PSW_H. bit←$\overline{PSW_H\text{. bit}}$					
		PSWL. bit	2	PSW_L. bit←$\overline{PSW_L\text{. bit}}$	×	×	×	×	×
	SET1	CY	1	CY←1					1
	CLR1	CY	1	CY←0					0
	NOT1	CY	1	CY←$\overline{CY}$					×

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
ビット操作	MOV1	CY, saddr. bit	3	CY←(saddr. bit)					×
		CY, sfr. bit	3	CY←sfr. bit					×
		CY, A. bit	2	CY←A. bit					×
		CY, X. bit	2	CY←X. bit					×
		CY, PSWH. bit	2	CY←PSW_H. bit					×
		CY, PSWL. bit	2	CY←PSW_L. bit					×
		saddr. bit, CY	3	(saddr. bit) ←CY					
		sfr. bit, CY	3	sfr. bit←CY					
		A. bit, CY	2	A. bit←CY					
		X. bit, CY	2	X. bit←CY					
		PSWH. bit, CY	2	PSW_H. bit←CY					
		PSWL. bit, CY	2	PSW_L. bit←CY					
	AND1	CY, saddr. bit	3	CY←CY∧(saddr. bit)					×
		CY, /saddr. bit	3	CY←CY∧$\overline{(saddr.\ bit)}$					×
		CY, sfr. bit	3	CY←CY∧sfr. bit					×
		CY, /sfr. bit	3	CY←CY∧$\overline{sfr.\ bit}$					×
		CY, A. bit	2	CY←CY∧A. bit					×
		CY, /A. bit	2	CY←CY∧$\overline{A.\ bit}$					×
		CY, X. bit	2	CY←CY∧X. bit					×
		CY, /X. bit	2	CY←CY∧$\overline{X.\ bit}$					×
		CY, PSWH. bit	2	CY←CY∧PSW_H. bit					×
		CY, /PSWH. bit	2	CY←CY∧$\overline{PSW_H.\ bit}$					×
		CY, PSWL. bit	2	CY←CY∧PSW_L. bit					×
		CY, /PSWL. bit	2	CY←CY∧$\overline{PSW_L.\ bit}$					×
	OR1	CY, saddr. bit	3	CY←CY∨(saddr. bit)					×
		CY, /saddr. bit	3	CY←CY∨$\overline{(saddr.\ bit)}$					×
		CY, sfr. bit	3	CY←CY∨sfr. bit					×
		CY, /sfr. bit	3	CY←CY∨$\overline{sfr.\ bit}$					×
		CY, A. bit	2	CY←CY∨A. bit					×
		CY, /A. bit	2	CY←CY∨$\overline{A.\ bit}$					×
		CY, X. bit	2	CY←CY∨X. bit					×
		CY, /X. bit	2	CY←CY∨$\overline{X.\ bit}$					×
		CY, PSWH. bit	2	CY←CY∨PSW_H. bit					×
		CY, /PSWH. bit	2	CY←CY∨$\overline{PSW_H.\ bit}$					×
		CY, PSWL. bit	2	CY←CY∨PSW_L. bit					×
		CY, /PSWL. bit	2	CY←CY∨$\overline{PSW_L.\ bit}$					×

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
コール・リターン	**CALL**	!addr16	3	(SP-1) ← $(PC+3)_H$, (SP−2) ← $(PC+3)_L$, PC←addr16, SP←SP−2					
	CALLF	!addr11	2	(SP−1) ← $(PC+2)_H$, (SP−2) ←$(PC+2)_L$, PC_{15-11}←00001, PC_{10-0}←addr11, SP←SP−2					
	CALLT	[addr5]	1	(SP−1)← $(PC+1)_H$, (SP−2)←$(PC+1)_L$, PC_H← (TPF, 000000001, addr5+1), PC_L← (TPF, 000000001, addr5), SP←SP−2					
	CALL	rp1	2	(SP−1) ← $(PC+2)_H$, (SP−2) ← $(PC+2)_L$, PC_H←$rp1_H$, PC_L←$rp1_L$, SP←SP−2					
		[rp1]	2	(SP−1) ← $(PC+2)_H$, (SP−2) ← $(PC+2)_L$, PC_H← (rp1+1), PC_L← (rp1), SP←SP−2					
	BRK		1	(SP−1) ←PSW_H, (SP−2) ←PSW_L (SP−3) ← $(PC+1)_H$, (SP−4) ←$(PC+1)_L$, PC_L← (003EH), PC_H← (003FH), SP←SP−4, IE←0					
	RET		1	PC_L← (SP), PC_H← (SP+1), SP←SP+2					
	RETB		1	PC_L← (SP), PC_H← (SP+1), PSW_L← (SP+2), PSW_H← (SP+3), SP←SP+4	R	R	R	R	R
	RETI		1	PC_L← (SP), PC_H← (SP+1), PSW_L← (SP+2), PSW_H← (SP+3), SP←SP+4, EOS←0	R	R	R	R	R
スタック操作	**PUSH**	sfrp	3	(SP−1) ←sfr_H (SP−2) ←sfr_L SP←SP−2					
		post	2	{(SP−1) ←$post_H$, (SP−2) ←$post_L$, SP←SP−2}×n回[注]					
		PSW	1	(SP−1) ←PSW_H, (SP−2) ←PSW_L, SP←SP−2					
	PUSHU	post	2	{(UP−1) ←$post_H$, (UP−2) ←$post_L$, UP←UP−2}×n回[注]					
	POP	sfrp	3	sfr_L← (SP) sfr_H← (SP+1) SP←SP+2					
		post	2	{$post_L$← (SP), $post_H$← (SP+1), SP←SP+2}×n回[注]					
		PSW	1	PSW_L← (SP), PSW_H← (SP+1), SP←SP+2	R	R	R	R	R
	POPU	post	2	{$post_L$← (UP), $post_H$← (UP+1), UP←UP+2}×n回[注]					
	MOVW	SP, #word	4	SP←word					
		SP, AX	2	SP←AX					
		AX, SP	2	AX←SP					
	INCW	SP	2	SP←SP+1					
	DECW	SP	2	SP←SP−1					
特殊	**CHKL**	sfr	3	(端子レベル) ⊻ (出力バッファの前段の信号レベル)	×	×		P	
	CHKLA	sfr	3	A←{(端子レベル) ∨ (出力バッファの前段の信号レベル)}	×	×		P	

注　nは，postとして記述したレジスタの数です。

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S Z AC P/V CY
無条件分岐	**BR**	! addr16	3	PC←addr16	
		rp1	2	PC_H←$rp1_H$, PC_L←$rp1_L$	
		[rp1]	2	PC_H← (rp1+1), PC_L← (rp1)	
		$ addr16	2	PC←PC+2+jdisp8	
条件付き分岐	**BC**	$ addr16	2	PC←PC+2+jdisp8 if CY=1	
	BL				
	BNC	$ addr16	2	PC←PC+2+jdisp8 if CY=0	
	BNL				
	BZ	$ addr16	2	PC←PC+2+jdisp8 if Z=1	
	BE				
	BNZ	$ addr16	2	PC←PC+2+jdisp8 if Z=0	
	BNE				
	BV	$ addr16	2	PC←PC+2+jdisp8 if P/V=1	
	BPE				
	BNV	$ addr16	2	PC←PC+2+jdisp8 if P/V=0	
	BPO				
	BN	$ addr16	2	PC←PC+2+jdisp8 if S=1	
	BP	$ addr16	2	PC←PC+2+jdisp8 if S=0	
	BGT	$ addr16	3	PC←PC+3+jdisp8 if (P/V⊻S) ∨Z=0	
	BGE	$ addr16	3	PC←PC+3+jdisp8 if P/V⊻S=0	
	BLT	$ addr16	3	PC←PC+3+jdisp8 if P/V⊻S=1	
	BLE	$ addr16	3	PC←PC+3+jdisp8 if (P/V⊻S) ∨Z=1	
	BH	$ addr16	3	PC←PC+3+jdisp8 if Z∨CY=0	
	BNH	$ addr16	3	PC←PC+3+jdisp8 if Z∨CY=1	
	BT	saddr. bit, $ addr16	3	PC←PC+3+jdisp8 if (saddr. bit)=1	
		sfr. bit, $ addr16	4	PC←PC+4+jdisp8 if sfr. bit=1	
		A. bit, $ addr16	3	PC←PC+3+jdisp8 if A. bit=1	
		X. bit, $ addr16	3	PC←PC+3+jdisp8 if X. bit=1	
		PSWH. bit, $ addr16	3	PC←PC+3+jdisp8 if PSW_H. bit=1	
		PSWL. bit, $ addr16	3	PC←PC+3+jdisp8 if PSW_L. bit=1	
	BF	saddr. bit, $ addr16	4	PC←PC+4+jdisp8 if (saddr. bit)=0	
		sfr. bit, $ addr16	4	PC←PC+4+jdisp8 if sfr. bit=0	
		A. bit, $ addr16	3	PC←PC+3+jdisp8 if A. bit=0	
		X. bit, $ addr16	3	PC←PC+3+jdisp8 if X. bit=0	
		PSWH. bit, $ addr16	3	PC←PC+3+jdisp8 if PSW_H. bit=0	
		PSWL. bit, $ addr16	3	PC←PC+3+jdisp8 if PSW_L. bit=0	

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
条件付き分岐	**BTCLR**	saddr. bit, $ addr16	4	PC←PC+4+jdisp8 if (saddr. bit)=1 then reset (saddr. bit)					
		sfr. bit, $ addr16	4	PC←PC+4+jdisp8 if sfr. bit=1 then reset sfr. bit					
		A. bit, $ addr16	3	PC←PC+3+jdisp8 if A. bit=1 then reset A. bit					
		X. bit, $ addr16	3	PC←PC+3+jdisp8 if X. bit=1 then reset X. bit					
		PSWH. bit, $ addr16	3	PC←PC+3+jdisp8 if PSW_H. bit=1 then reset PSW_H. bit					
		PSWL. bit, $ addr16	3	PC←PC+3+jdisp8 if PSW_L. bit=1 then reset PSW_L. bit	×	×	×	×	×
	BFSET	saddr. bit, $ addr16	4	PC←PC+4+jdisp8 if (saddr. bit)=0 then set (saddr. bit)					
		sfr. bit, $ addr16	4	PC←PC+4+jdisp8 if sfr. bit=0 then set sfr. bit					
		A. bit, $ addr16	3	PC←PC+3+jdisp8 if A. bit=0 then set A. bit					
		X. bit, $ addr16	3	PC←PC+3+jdisp8 if X. bit=0 then set X. bit					
		PSWH. bit, $ addr16	3	PC←PC+3+jdisp8 if PSW_H. bit=0 then set PSW_H. bit					
		PSWL. bit, $ addr16	3	PC←PC+3+jdisp8 if PSW_L. bit=0 then set PSW_L. bit	×	×	×	×	×
	DBNZ	r2, $ addr16	2	r2←r2−1, then PC←PC+2+jdisp8 if r2≠0					
		saddr, $ addr16	3	(saddr)←(saddr)−1, then PC←PC+3+jdisp8 if (saddr)≠0					
コンテキスト・スイッチング	**BRKCS**	RBn	2	PC_H↔R5, PC_L↔R4, R7←PSW_H R6←PSW_L, RBS2−0←n, RSS←0, IE←0					
	RETCS	! addr16	3	PC_H←R5, PC_L←R4, R5, R4←addr 16, PSW_H←R7, PSW_L←R6	R	R	R	R	R
	RETCSB	! addr16	4	PC_H←R5, PC_L←R4, R5←addr 16_H, R4←addr 16_L, PSW_H←R7, PSW_L←R6	R	R	R	R	R

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
ストリング	**MOVM**	[DE+], A	2	(DE+) ←A, C←C−1 End if C=0					
		[DE−], A	2	(DE−)←A, C←C−1 End if C=0					
	MOVBK	[DE+], [HL+]	2	(DE+)←(HL+), C←C−1 End if C=0					
		[DE−], [HL−]	2	(DE−)←(HL−), C←C−1 End if C=0					
	XCHM	[DE+], A	2	(DE+)↔A, C←C−1 End if C=0					
		[DE−], A	2	(DE−)↔A, C←C−1 End if C=0					
	XCHBK	[DE+], [HL+]	2	(DE+)↔(HL+), C←C−1 End if C=0					
		[DE−], [HL−]	2	(DE−)↔(HL−), C←C−1 End if C=0					
	CMPME	[DE+], A	2	(DE+)−A, C←C−1 End if C=0 or Z=0	×	×	×	V	×
		[DE−], A	2	(DE−)−A, C←C−1 End if C=0 or Z=0	×	×	×	V	×
	CMPBKE	[DE+], [HL+]	2	(DE+)−(HL+), C←C−1 End if C=0 or Z=0	×	×	×	V	×
		[DE−], [HL−]	2	(DE−)−(HL−), C−C−1 End if C=0 or Z=0	×	×	×	V	×
	CMPMNE	[DE+], A	2	(DE+)−A, C←C−1 End if C=0 or Z=1	×	×	×	V	×
		[DE−], A	2	(DE−)−A, C←C−1 End if C=0 or Z=1	×	×	×	V	×
	CMPBKNE	[DE+], [HL+]	2	(DE+)−(HL+), C←C−1 End if C=0 or Z=1	×	×	×	V	×
		[DE−], [HL−]	2	(DE−)−(HL−), C←C−1 End if C=0 or Z=1	×	×	×	V	×
	CMPMC	[DE+], A	2	(DE+)−A, C←C−1 End if C=0 or CY=0	×	×	×	V	×
		[DE−], A	2	(DE−)−A, C←C−1 End if C=0 or CY=0	×	×	×	V	×

命令群	ニモニック	オペランド	バイト	オペレーション	フラグ S	Z	AC	P/V	CY
ストリング	**CMPBKC**	[DE+], [HL+]	2	(DE+)－(HL+), C←C－1 End if C=0 or CY=0	×	×	×	V	×
		[DE－], [HL－]	2	(DE－)－(HL－), C←C－1 End if C=0 or CY=0	×	×	×	V	×
	CMPMNC	[DE+], A	2	(DE+)－A, C←C－1 End if C=0 or CY=1	×	×	×	V	×
		[DE－], A	2	(DE－)－A, C←C－1 End if C=0 or CY=1	×	×	×	V	×
	CMPBKNC	[DE+], [HL+]	2	(DE+)－(HL+), C←C－1 End if C=0 or CY=1	×	×	×	V	×
		[DE－], [HL－]	2	(DE－)－(HL－), C←C－1 End if C=0 or CY=1	×	×	×	V	×
CPU制御	**MOV**	STBC, #byte	4	STBC←byte注					
		WDM, #byte	4	WDM←byte注					
	SWRS		1	RSS←$\overline{\text{RSS}}$					
	SEL	RBn	2	RBS2－0←n, RSS←0					
		RBn, ALT	2	RBS2－0←n, RSS←1					
	NOP		1	No Operation					
	EI		1	IE←1 (Enable Interrupt)					
	DI		1	IE←0 (Disable Interrupt)					

注　オペコード・トラップ割り込み発生時のオペレーション

$(SP-1)\leftarrow PSW_H$, $(SP-2)\leftarrow PSW_L$

$(SP-3)\leftarrow (PC-4)_H$, $(SP-4)\leftarrow (PC-4)_L$

$PC_L\leftarrow (003CH)$, $PC_H\leftarrow (003DH)$

$SP\leftarrow SP-4$, $IE\leftarrow 0$

〔メ　モ〕

保守／廃止

★

付録B　改版履歴

これまでの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

版　数	前版からの主な改版内容	適用箇所
第3版	次の製品を追加 μPD78320(A1)，78320(A2)，78322(A1)，78322(A2)，78323(A)，78323(A1)，78323(A2)，78324(A)，78324(A1)，78324(A2)，78P324(A)，78P324(A1)，78P324(A2)，78330(A)，78330(A1)，78330(A2)，78334(A)，78334(A1)，78334(A2)，78P334(A)，78P334(A1)，78P334(A2)，78350，78350A，78352A，78P352，78355，78356，78P356，78355A，78356A，78P356A，78362，78P364，78365，78366，78P368，78370，78372，78P372	全　般
	次の製品を開発中→開発済みに変更 μPD78323，78324，78P334	
第4版	次の製品を追加 μPD78356(A)，78P356(A)，78361A，78362A，78P364A，78363A，78365A，78366A，78368A，78P368A，78372(A)，78372(A1)，78372(A2)，78P372(A)，78P372(A1)，78P372(A2)	全　般
	次の製品を削除 μPD78355A，78356A，78P356A，78362，78P364，78365，78366，78P368，78370，78372，78P372	
	次の製品を開発中→開発済みに変更 μPD78P324，78P324(A)，78P324(A1)，78P324(A2)，78350A，78355，78356，78P356	

保守／廃止

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)

保守／廃止

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/Ⅲシリーズ　アプリケーション・ノート　ソフトウエア基礎編
（U12118JJ4V1AN00（第4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

キリトリ