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。

User's Manual

16

RENESAS

改訂一覧は表紙をクリックして直接ご覧になれます。
改訂一覧は改訂箇所をまとめたものであり、詳細については、
必ず本文の内容をご確認ください。

H8S/2600シリーズ、H8S/2000シリーズ

ソフトウェアマニュアル

ルネサス16ビットシングルチップマイクロコンピュータ
H8Sファミリ

ルネサス エレクトロニクス
www.renesas.com

Rev.5.00　2006.02

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (http://www.renesas.com) などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

はじめに

H8S/2600 シリーズおよび H8S/2000 シリーズは、それぞれ H8S/2600 CPU、H8S/2000 CPU をコアとしています。H8S/2600 および H8S/2000 CPU は、内部 32 ビット構成の共通のアーキテクチャを持っています。いずれの CPU も基本命令を 1 ステートで実行でき、16 ビット×16 本の汎用レジスタと簡潔で最適化された命令セットを備えています。また、16M／4G バイトのリニアなアドレス空間を扱うことができます。高級言語 C をベースとしたプログラムも効率的に実行できます。

命令は、H8/300H、H8/300、H8/300L シリーズとオブジェクトレベルで上位互換を保っており、容易に移行することができます。

H8S/2600 CPU は、H8S/2000 CPU に対しオブジェクトレベルで上位互換であり、積和演算命令をサポートしています。

本マニュアルは、H8S/2600 および H8S/2000 CPU の命令の詳細について記載しており、H8S/2600 シリーズおよび H8S/2000 シリーズ共通に使用することができます。

なお、ハードウェアの詳細については、当該 LSI のハードウェアマニュアルをご覧ください。

本版で改訂された箇所

<table>
<tr><th>修正項目</th><th>ページ</th><th>修正内容（詳細はマニュアル参照）</th></tr>
<tr><td rowspan="2">1.1.1　特長</td><td>1-1</td><td>注を追加
• 高速動作
– 頻出命令をすべて１～２ステートで実行
– 最高動作周波数：20MHz*</td></tr>
<tr><td>1-2</td><td>【注】* 最高動作周波数および命令実行時間は製品によって異なります。</td></tr>
<tr><td>2.2.27　CLRMAC</td><td>2-49</td><td>注に記述を追加
例えば、MAC 命令と CLRMAC 命令の間に 1 ステート命令（NOP 等）が 1 つある場合、CLRMAC 命令は 2 ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。</td></tr>
<tr><td>2.2.31　DAA</td><td>2-53</td><td>表を修正
<table><tr><th>補正前の
C フラグ</th><th>補正前の
上位 4 ビット</th></tr><tr><td>1</td><td>0～2</td></tr><tr><td>1</td><td>0～2</td></tr><tr><td>1</td><td>0～3</td></tr></table></td></tr>
<tr><td>2.2.56　LDMAC</td><td>2-82</td><td>注に記述を追加
例えば、MAC 命令と LDMAC 命令の間に 1 ステート命令（NOP 等）が 1 つある場合、LDMAC 命令は 2 ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。</td></tr>
<tr><td>2.2.69　MULXS (B)</td><td>2-103</td><td>注に記述を追加
例えば、MAC 命令と MULXS.B 命令の間に 1 ステート命令（NOP 等）が 1 つある場合、MULXS.B 命令は 1 ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。</td></tr>
<tr><td>2.2.70　MULXS (W)</td><td>2-104</td><td>注に記述を追加
例えば、MAC 命令と MULXS.W 命令の間に 1 ステート命令（NOP 等）が 1 つある場合、MULXS.W 命令は 1 ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。</td></tr>
<tr><td>2.2.71　MULXU (B)</td><td>2-105</td><td>注に記述を追加
例えば、MAC 命令と MULXU.B 命令の間に 1 ステート命令（NOP 等）が 1 つある場合、MULXU.B 命令は 2 ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。</td></tr>
</table>

<table>
<tr><th>修正項目</th><th>ページ</th><th>修正内容（詳細はマニュアル参照）</th></tr>
<tr><td>2.2.72　MULXU (W)</td><td>2-106</td><td>注に記述を追加
例えば、MAC 命令と MULXU.W 命令の間に 1 ステート命令（NOP 等）が 1 つある場合、MULXU.W 命令は 2 ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。</td></tr>
<tr><td>2.2.145　STMAC</td><td>2-182</td><td>表を修正
<table><tr><th rowspan="2">アドレッシングモード</th><th rowspan="2">ニーモニック</th><th rowspan="2">オペランド形式</th><th colspan="6">インストラクションフォーマット</th><th rowspan="2">実行ステート数</th></tr><tr><th colspan="2">第1バイト</th><th colspan="2">第2バイト</th><th>第3バイト</th><th>第4バイト</th></tr><tr><td>レジスタ直接</td><td>STMAC</td><td>MACH, ERd</td><td>0</td><td>2</td><td>2</td><td>0</td><td>erd</td><td></td><td></td><td>1*</td></tr><tr><td>レジスタ直接</td><td>STMAC</td><td>MACL, ERd</td><td>0</td><td>2</td><td>3</td><td>0</td><td>erd</td><td></td><td></td><td>1*</td></tr></table>
注に記述を追加
例えば、MAC 命令と STMAC 命令の間に 1 ステート命令（NOP 等）が 1 つある場合、STMAC 命令は 2 ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。</td></tr>
<tr><td>2.3　命令セット一覧
表 2.1　命令セット一覧
（2）算術演算命令</td><td>2-197</td><td>注を追加
<table><tr><th rowspan="3" colspan="2">ニーモニック</th><th rowspan="3">サイズ</th><th colspan="8">アドレッシングモード／命令長（バイト）</th><th rowspan="3">オペレーション</th><th colspan="6">コンディションコード</th><th colspan="2">実行ステート数*1</th></tr><tr><th rowspan="2">#xx</th><th rowspan="2">Rn</th><th rowspan="2">@ERn</th><th rowspan="2">@(d,ERn)</th><th rowspan="2">@-ERn/@ERn+</th><th rowspan="2">@aa</th><th rowspan="2">@(d,PC)</th><th rowspan="2">@@aa</th><th rowspan="2">|</th><th rowspan="2">I</th><th rowspan="2">H</th><th rowspan="2">N</th><th rowspan="2">Z</th><th rowspan="2">V</th><th rowspan="2">C</th><th>ノーマル</th><th>アドバンスト</th></tr><tr><th>—</th><th>—</th></tr><tr><td>MULXU</td><td>MULXU.B Rs,Rd</td><td>B</td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Rd8×Rs8→Rd16（符号なし乗算）</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>3(12*2)*6倍</td><td></td></tr><tr><td></td><td>MULXU.W Rs,ERd</td><td>W</td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Rd16×Rs16→ERd32（符号なし乗算）</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>4(20*2)*6倍</td><td></td></tr><tr><td>MULXS</td><td>MULXS.B Rs,Rd</td><td>B</td><td></td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Rd8×Rs8→Rd16（符号付き乗算）</td><td>—</td><td>—</td><td>↕</td><td>↕</td><td>—</td><td>—</td><td>4(13*2)*6倍</td><td></td></tr><tr><td></td><td>MULXS.W Rs,ERd</td><td>W</td><td></td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Rd16×Rs16→ERd32（符号付き乗算）</td><td>—</td><td>—</td><td>↕</td><td>↕</td><td>—</td><td>—</td><td>5(21*2)*6倍</td><td></td></tr><tr><td>CLRMAC*1</td><td>CLRMAC</td><td>—</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2</td><td>0→MACH, MACL</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>2*2*6倍</td><td></td></tr><tr><td>LDMAC*1</td><td>LDMAC ERs,MACH</td><td>L</td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>ERs→MACH</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>2*2*6倍</td><td></td></tr><tr><td></td><td>LDMAC ERs,MACL</td><td>L</td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>ERs→MACL</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>2*2*6倍</td><td></td></tr><tr><td>STMAC*1</td><td>STMAC MACH,ERd</td><td>L</td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>MACH→ERd</td><td>—</td><td>—</td><td>↕</td><td>↕</td><td>↕</td><td>—</td><td>1*2*6倍</td><td></td></tr><tr><td></td><td>STMAC MACL,ERd</td><td>L</td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>MACL→ERd</td><td>—</td><td>—</td><td>↕</td><td>↕</td><td>↕</td><td>—</td><td>1*2*6倍</td><td></td></tr></table></td></tr>
<tr><td>（7）システム制御命令</td><td>2-203</td><td>注を追加
<table><tr><th rowspan="3" colspan="2">ニーモニック</th><th rowspan="3">サイズ</th><th colspan="8">アドレッシングモード／命令長（バイト）</th><th rowspan="3">オペレーション</th><th colspan="6">コンディションコード</th><th colspan="2">実行ステート数*1</th></tr><tr><th rowspan="2">#xx</th><th rowspan="2">Rn</th><th rowspan="2">@ERn</th><th rowspan="2">@(d,ERn)</th><th rowspan="2">@-ERn/@ERn+</th><th rowspan="2">@aa</th><th rowspan="2">@(d,PC)</th><th rowspan="2">@@aa</th><th rowspan="2">|</th><th rowspan="2">I</th><th rowspan="2">H</th><th rowspan="2">N</th><th rowspan="2">Z</th><th rowspan="2">V</th><th rowspan="2">C</th><th>ノーマル</th><th>アドバンスト</th></tr><tr><th>—</th><th>—</th></tr><tr><td>TRAPA</td><td>TRAPA #xx:2</td><td>—</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2</td><td>PC→@-SP,CCR→@-SP,
EXR→@-SP,<ベクタ>→PC</td><td>1</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>7 [9*3]</td><td>8 [9*3]</td></tr><tr><td>RTE</td><td>RTE</td><td>—</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>EXR←@SP+,CCR←@SP+,
PC←@SP+</td><td>↕</td><td>↕</td><td>↕</td><td>↕</td><td>↕</td><td>↕</td><td>5 [9*3]</td><td></td></tr></table></td></tr>
</table>

修正項目	ページ	修正内容（詳細はマニュアル参照）
2.3　命令セット一覧 表 2.1　命令セット一覧 （8）ブロック転送命令	2-204	注を修正、追加 【注】*　H8S/2600 CPU でのみサポートしています。 *1　実行ステート数は、命令コードおよびオペランドが内蔵メモリに存在する場合の値です。 *2　（ ）内は H8S/2000 CPU の値です。［ ］内は割り込み制御モード 2、3 の値です。 *3　n は R4L または R4 の初期設定値です。 *4　復帰／退避レジスタ数が 2 本のとき 7 ステート、3 本のとき 9 ステート、4 本のとき 11 ステートになります。 *5　MULXU、MULXS、STMAC 命令の直後は 1 ステート多くなります。また、MAC 命令実行後 3 ステート以内に MULXU 命令を実行しようとした場合、最大で 3 ステート多くなります。例えば、MAC 命令と MULXU 命令の間に 1 ステート命令(NOP 等)が 1 つある場合、MULXU 命令は 2 ステート多くなります。 *6　MAC 命令実行後 2 ステート以内に MULXS 命令を実行しようとした場合、最大で 2 ステート多くなります。例えば、MAC 命令と MULXS 命令の間に 1 ステート命令（NOP 等）が 1 つある場合、MULXS 命令は 1 ステート多くなります。 *7　MAC 命令実行後 3 ステート以内にこれらの命令を実行しようとした場合、最大で 3 ステート多くなります。例えば、MAC 命令とこれらの命令の間に 1 ステート命令（NOP 等）が 1 つある場合、これらの命令は 2 ステート多くなります。 *8　TAS 命令を使用する場合は、レジスタ ER0、ER1、ER4、ER5 を使用してください。 *9　製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

<table>
<tr><th>修正項目</th><th>ページ</th><th>修正内容（詳細はマニュアル参照）</th></tr>
<tr><td rowspan="4">2.6　命令実行ステート数
表 2.5　命令実行状態（サイクル数）</td><td>2-224</td><td>注を追加
<table>
<tr><th>命令</th><th>ニーモニック</th><th>命令フェッチ</th><th>分岐アドレスリード</th><th>スタック操作</th><th>バイトデータアクセス</th><th>ワードデータアクセス</th><th>内部動作</th></tr>
<tr><td></td><td></td><td>I</td><td>J</td><td>K</td><td>L</td><td>M</td><td>N</td></tr>
<tr><td>CLRMAC *3</td><td>CLRMAC</td><td>1</td><td></td><td></td><td></td><td></td><td>1*6</td></tr>
</table>
</td></tr>
<tr><td>2-225</td><td>
<table>
<tr><th>命令</th><th>ニーモニック</th><th>命令フェッチ</th><th>分岐アドレスリード</th><th>スタック操作</th><th>バイトデータアクセス</th><th>ワードデータアクセス</th><th>内部動作</th></tr>
<tr><td></td><td></td><td>I</td><td>J</td><td>K</td><td>L</td><td>M</td><td>N</td></tr>
<tr><td rowspan="2">LDMAC *3</td><td>LDMAC ERs, MACH</td><td>1</td><td></td><td></td><td></td><td></td><td>1*6</td></tr>
<tr><td>LDMAC ERs, MACL</td><td>1</td><td></td><td></td><td></td><td></td><td>1*6</td></tr>
</table>
</td></tr>
<tr><td>2-227</td><td>
<table>
<tr><th rowspan="2">命令</th><th rowspan="2">ニーモニック</th><th rowspan="2"></th><th>命令フェッチ</th><th>分岐アドレスリード</th><th>スタック操作</th><th>バイトデータアクセス</th><th>ワードデータアクセス</th><th>内部動作</th></tr>
<tr><th>I</th><th>J</th><th>K</th><th>L</th><th>M</th><th>N</th></tr>
<tr><td rowspan="8">MULXS</td><td rowspan="2">MULXS.B Rs, Rd</td><td>H8S/2600</td><td>2</td><td></td><td></td><td></td><td></td><td>2*6</td></tr>
<tr><td>H8S/2000</td><td>2</td><td></td><td></td><td></td><td></td><td>11</td></tr>
<tr><td rowspan="2">MULXS.W Rs, ERd</td><td>H8S/2600</td><td>2</td><td></td><td></td><td></td><td></td><td>3*6</td></tr>
<tr><td>H8S/2000</td><td>2</td><td></td><td></td><td></td><td></td><td>19</td></tr>
<tr><td rowspan="4">MULXU</td><td rowspan="2">MULXU.B Rs, Rd</td><td>H8S/2600</td><td>1</td><td></td><td></td><td></td><td></td><td>2*6</td></tr>
<tr><td>H8S/2000</td><td>1</td><td></td><td></td><td></td><td></td><td>11</td></tr>
<tr><td rowspan="2">MULXU.W Rs, ERd</td><td>H8S/2600</td><td>1</td><td></td><td></td><td></td><td></td><td>3*6</td></tr>
<tr><td>H8S/2000</td><td>1</td><td></td><td></td><td></td><td></td><td>19</td></tr>
</table>
</td></tr>
<tr><td>2-229</td><td>
<table>
<tr><th>命令</th><th>ニーモニック</th><th>命令フェッチ</th><th>分岐アドレスリード</th><th>スタック操作</th><th>バイトデータアクセス</th><th>ワードデータアクセス</th><th>内部動作</th></tr>
<tr><td></td><td></td><td>I</td><td>J</td><td>K</td><td>L</td><td>M</td><td>N</td></tr>
<tr><td rowspan="2">STMAC *3</td><td>STMAC MACH, ERd</td><td>1</td><td></td><td></td><td></td><td></td><td>0*6</td></tr>
<tr><td>STMAC MACL, ERd</td><td>1</td><td></td><td></td><td></td><td></td><td>0*6</td></tr>
</table>
</td></tr>
<tr><td>2-230</td><td>*6　製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。</td></tr>
<tr><td>2.7　命令実行中のバス状態
表 2.6　命令の実行状態</td><td>2-231、
2-233～
2-242</td><td>記号説明から「：M」を削除。</td></tr>
<tr><td>3.3.5　使用上の注意事項</td><td>3-4</td><td>新規追加</td></tr>
</table>

目次

1. CPU

1.1 概要

H8S/2600 CPU および H8S/2000 CPU は、共通のアーキテクチャを持つ内部 32 ビット構成の高速 CPU です。いずれも、H8/300 CPU および H8/300H CPU の上位互換です。

H8S/2600 CPU および H8S/2000 CPU は、16 ビット×16 本の汎用レジスタを持ち、4G バイトのリニアなアドレス空間を扱うことができ、リアルタイム制御に最適です。

1.1.1 特長

H8S/2600 CPU および H8S/2000 CPU には、次の特長があります。

- H8/300 CPU および H8/300H CPU の上位互換
  - H8/300 および H8/300H CPU オブジェクトプログラムを実行可能
- 汎用レジスタ方式
  - 16 ビット×16 本（8 ビット×16 本、32 ビット×8 本としても使用可能）
- 69 種類（H8S/2000 CPU は 65 種類）の基本命令
  - 8／16／32 ビット演算命令
  - 乗除算命令
  - 強力なビット操作命令
  - 積和演算命令（H8S/2600 CPU のみ）
- 8 種類のアドレッシングモード
  - レジスタ直接（Rn）
  - レジスタ間接（@ERn）
  - ディスプレースメント付レジスタ間接（@(d:16,ERn)／@(d:32,ERn)）
  - ポストインクリメント／プリデクリメントレジスタ間接（@ERn＋／@－ERn）
  - 絶対アドレス（@aa:8／@aa:16／@aa:24／@aa:32）
  - イミディエイト（#xx:8／#xx:16／#xx:32）
  - プログラムカウンタ相対（@(d:8,PC)／@(d:16,PC)）
  - メモリ間接（@@aa:8）
- 4G バイトのアドレス空間
  - プログラム　16M バイト
  - データ　　4G バイト
- 高速動作
  - 頻出命令をすべて１～２ステートで実行
  - 最高動作周波数：20MHz*
  - 8／16／32 ビットレジスタ間加減算　50ns
  - 8×8 ビットレジスタ間乗算　150ns（H8S/2000 CPU は600ns）
  - 16÷8 ビットレジスタ間除算　600ns
  - 16×16 ビットレジスタ間乗算　200ns（H8S/2000 CPU は1000ns）
  - 32÷16 ビットレジスタ間除算　1000ns

- 2 種類の CPU 動作モード
  – ノーマルモード／アドバンストモード
- 低消費電力状態
  – SLEEP 命令により低消費電力状態に遷移
  – CPU 動作クロックを選択可能

【注】＊　最高動作周波数および命令実行時間は製品によって異なります。

1.1.2 H8S/2600 CPU と H8S/2000 CPU の相違点

H8S/2600 CPU および H8S/2000 CPU の相違点は、以下の通りです。

- レジスタ構成
  – MAC レジスタは、H8S/2600 CPU のみサポートしています。
    詳細は「1.4　レジスタ構成」を参照してください。
- 基本命令
  – MAC、CLRMAC、LDMAC、STMAC の 4 命令は、H8S/2600 CPU のみサポートしています。
    詳細は「1.6　命令セット」、「第2章　各命令の説明」を参照してください。
- 実行ステート数
  – MULXU、MULXS 命令の実行ステート数
    詳細は「2.6　命令実行ステート数」を参照してください。

そのほか、製品によって、アドレス空間や EXR レジスタの機能、低消費電力状態などが異なる場合があります。詳細は、当該製品のハードウェアマニュアルを参照してください。

1.1.3 H8/300 CPU との相違点

H8S/2600 CPU および H8S/2000 CPU は、H8/300 CPU に対して、次の点が追加、拡張されています。

- 汎用レジスタ、コントロールレジスタを拡張
  – 16 ビット×8 本の拡張レジスタ、および 8 ビット×1 本、32 ビット×2 本のコントロールレジスタを追加
- アドレス空間を拡張
  – ノーマルモードのとき、H8/300 CPU と同一の 64k バイトのアドレス空間を使用可能
  – アドバンストモードのとき、最大 4G バイトのアドレス空間を使用可能
- アドレッシングモードを強化
  – 4G バイトのアドレス空間を有効に使用可能
- 命令強化
  – ビット操作命令のアドレッシングモードを強化
  – 符号付き乗除算命令などを追加
  – 積和演算命令を追加（H8S/2600 CPU のみ）
  – 2 ビットシフト、2 ビットローテート命令を追加
  – 複数レジスタの退避／復帰命令を追加
  – テストアンドセット命令を追加
- 高速化
  – 基本的な命令を 2 倍に高速化

1.1.4 H8/300H CPU との相違点

H8S/2600 CPU および H8S/2000 CPU は、H8/300H CPU に対して、次の点が追加、拡張されています。

- コントロールレジスタを拡張
  - 8 ビット×1 本、32 ビット×2 本のコントロールレジスタを追加
- アドレス空間を拡張
  - アドバンストモードのとき、データのみ最大 4G バイトのアドレス空間を使用可能
- 命令強化
  - ビット操作命令のアドレッシングモードを強化
  - 積和演算命令を追加（H8S/2600 CPU のみ）
  - 2 ビットシフト、2 ビットローテート命令を追加
  - 複数レジスタの退避／復帰命令を追加
  - テストアンドセット命令を追加
- 高速化
  - 基本的な命令を 2 倍に高速化

1.2 CPU 動作モード

本 CPU は、H8/300H CPU と同様に、ノーマルモードおよびアドバンストモードの 2 つの CPU 動作モードを持っています。サポートするアドレス空間は、ノーマルモードの場合最大 64k バイト、アドバンストモードの場合、プログラム領域最大 16M バイト、データ領域最大 4G バイト、合計で最大 4G バイトとなります。

各モードは LSI のモード端子によって選択されます。詳細は当該製品のハードウェアマニュアルを参照してください。

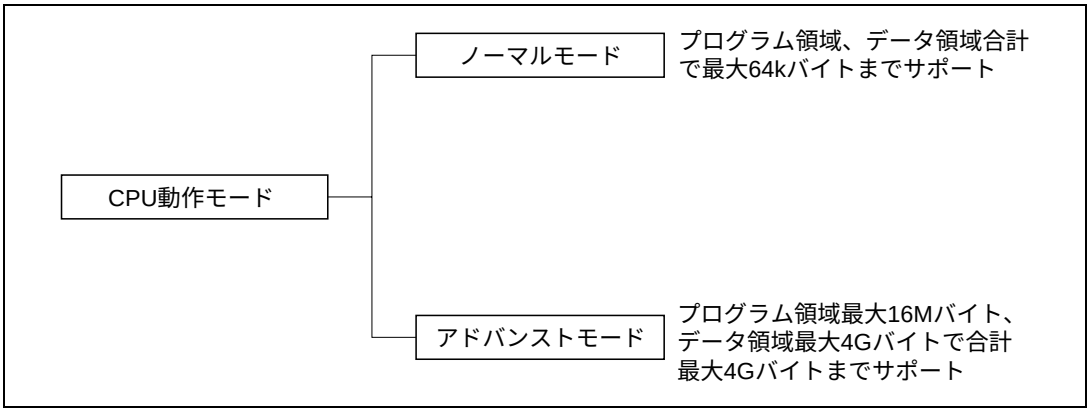


図 1.1 CPU 動作モード

(1) ノーマルモード

ノーマルモードでは例外処理ベクタ、スタックの構造が H8/300 CPU と同一になります。

(a) アドレス空間

H8/300 CPU と同様、最大 64k バイトをアクセス可能です。

(b) 拡張レジスタ（En）

拡張レジスタ（E0～E7）は、16 ビットレジスタとして、または 32 ビットレジスタの上位 16 ビットとして使用できます。

拡張レジスタ En は、対応する汎用レジスタ Rn をアドレスレジスタとして使用している場合でも、16 ビットレジスタとして任意の値を設定することができます（ただし、プリデクリメントレジスタ間接（@－Rn）、ポストインクリメントレジスタ間接（@Rn＋）により汎用レジスタ Rn が参照された場合、キャリ／ボローが発生すると、対応する拡張レジスタ En の内容に伝播しますので注意してください）。

(c) 命令セット

H8/300 CPU に対して追加された命令およびアドレッシングモードはすべて使用できます。実効アドレス（EA）の下位 16 ビットのみが有効となります。

(d) 例外処理ベクタテーブルおよびメモリ間接の分岐アドレス

ノーマルモードでは、H'0000 から始まる先頭領域に例外処理ベクタテーブル領域が割り当てられており、各 16 ビットの分岐先アドレスを格納します。ノーマルモードの例外処理ベクタテーブルの構造を図 1.2 に示します。例外処理ベクタテーブルは各製品ごとに異なりますので、詳細は当該製品

のハードウェアマニュアルを参照してください。

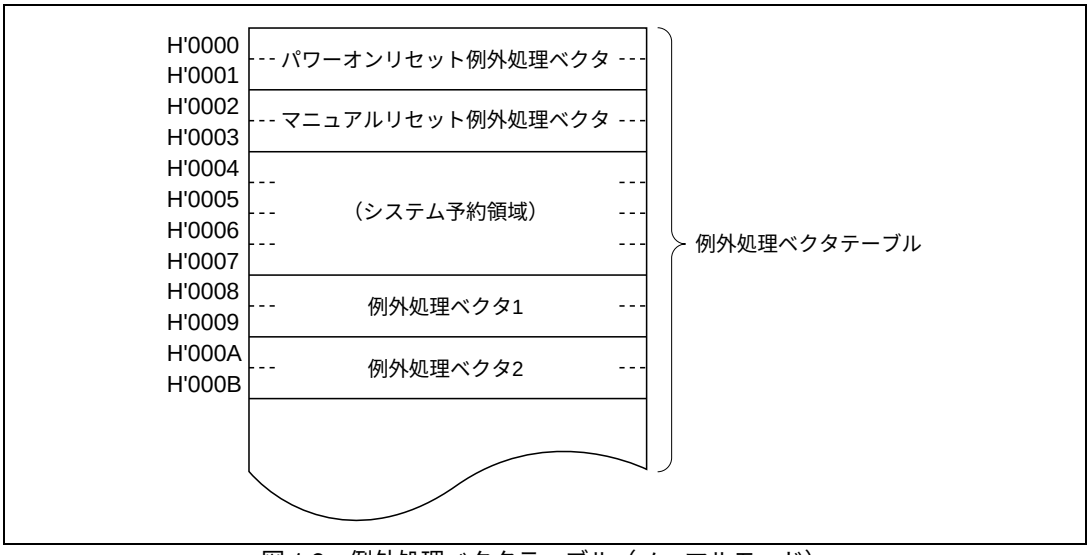


図 1.2　例外処理ベクタテーブル（ノーマルモード）

メモリ間接（@@aa:8）は、JMP および JSR 命令で使用されます。命令コードに含まれる 8 ビット絶対アドレスによりメモリ上のオペランドを指定し、この内容が分岐先アドレスとなります。

ノーマルモードでは、オペランドは 16 ビット（ワード）となり、この 16 ビットが分岐先アドレスとなります。なお、分岐先アドレスを格納できるのは、H'0000～H'00FF の領域であり、この範囲の先頭領域は例外処理ベクタテーブルと共通となっていますので注意してください。

(e)　スタック構造

サブルーチン分岐時の PC のスタック構造と、例外処理時の PC と CCR、EXR のスタックの構造を図 1.3 に示します。EXR は EXR が無効のときはスタックされません。詳細は当該製品のハードウェアマニュアルを参照してください。

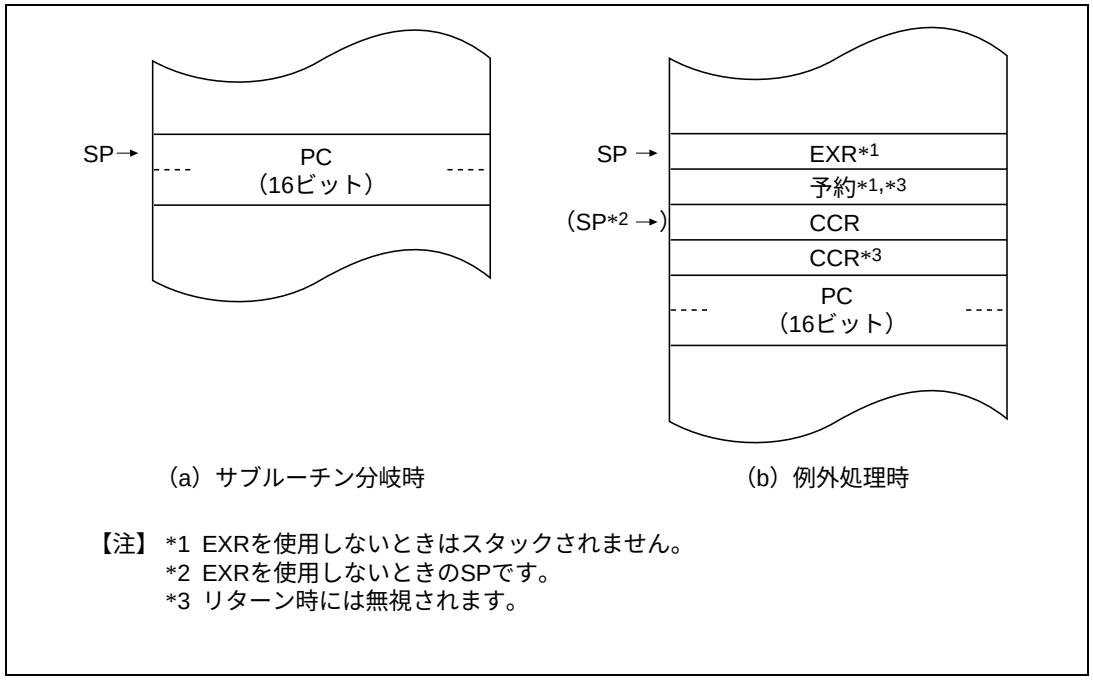


図 1.3　ノーマルモードのスタック構造

(2)　アドバンストモード

アドバンストモードでは、H8/300H CPU に対して、アドレス空間（データ領域）が拡張されています。

(a)　アドレス空間

プログラム領域最大 16M バイト、データ領域最大 4G バイト、合計最大 4G バイトをリニアにアクセス可能です。

(b)　拡張レジスタ（En）

拡張レジスタ（E0～E7）は、16 ビットレジスタとして、または 32 ビットレジスタ・アドレスレジスタの上位 16 ビットとして使用できます。

(c)　命令セット

命令およびアドレッシングモードはすべて使用できます。

(d)　例外処理ベクタテーブル、メモリ間接の分岐アドレス

アドバンストモードでは、H'00000000 から始まる先頭領域に 32 ビット単位で例外処理ベクタテーブル領域が割り当てられており、上位 8 ビットは無視され 24 ビットの分岐先アドレスを格納します（図 1.4 参照）。例外処理ベクタテーブルは各製品ごとに異なりますので、詳細は当該製品のハードウェアマニュアルを参照してください。

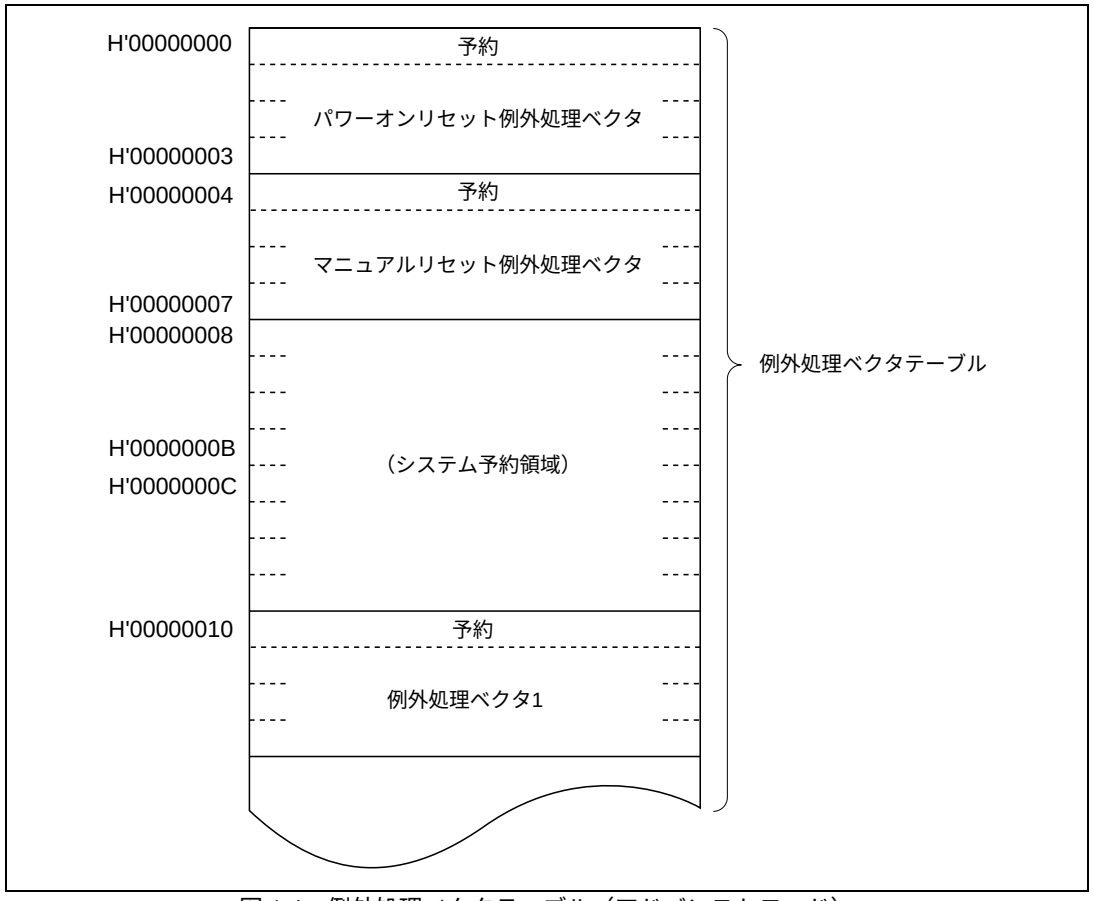


図 1.4　例外処理ベクタテーブル（アドバンストモード）

メモリ間接（@@aa:8）は、JMP および JSR 命令で使用されます。命令コードに含まれる 8 ビット絶対アドレスによりメモリ上のオペランドを指定し、この内容が分岐先アドレスとなります。

アドバンストモードでは、オペランドは 32 ビット（ロングワード）となり、この 32 ビットが分岐先アドレスとなります。この内、上位 8 ビットは予約領域となっており H'00 と見なされます。なお、分岐先アドレスを格納できるのは、H'00000000～H'000000FF の領域であり、この範囲の先頭領域は例外処理ベクタテーブルと共通となっていますので注意してください。

(e)　スタック構造

アドバンストモード時のサブルーチン分岐時の PC のスタック構造と、例外処理時の PC と CCR、EXR のスタックの構造を図 1.5 に示します。EXR は EXR が無効のときはスタックされません。詳細は当該製品のハードウェアマニュアルを参照してください。

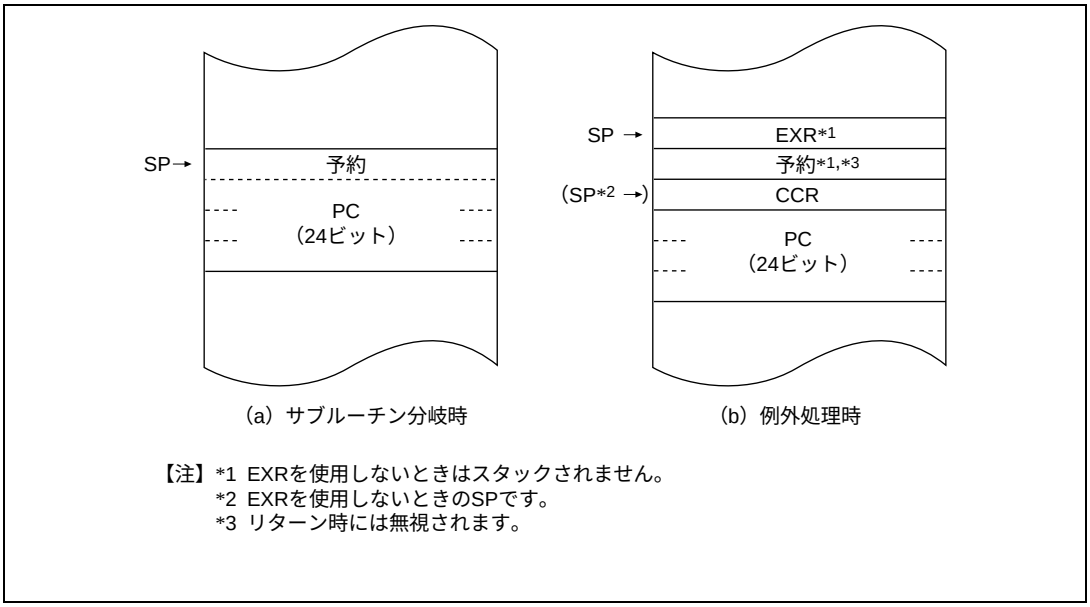


図 1.5　アドバンストモードのスタック構造

1.3 アドレス空間

本 CPU のメモリマップを図 1.6 に示します。本 CPU は、ノーマルモードのとき最大 64k バイト、またアドバンストモードのとき最大 4G バイトのアドレス空間をリニアに使用することができます。

アドレス空間は動作モードなどによって異なります。詳細は当該製品のハードウェアマニュアルを参照してください。

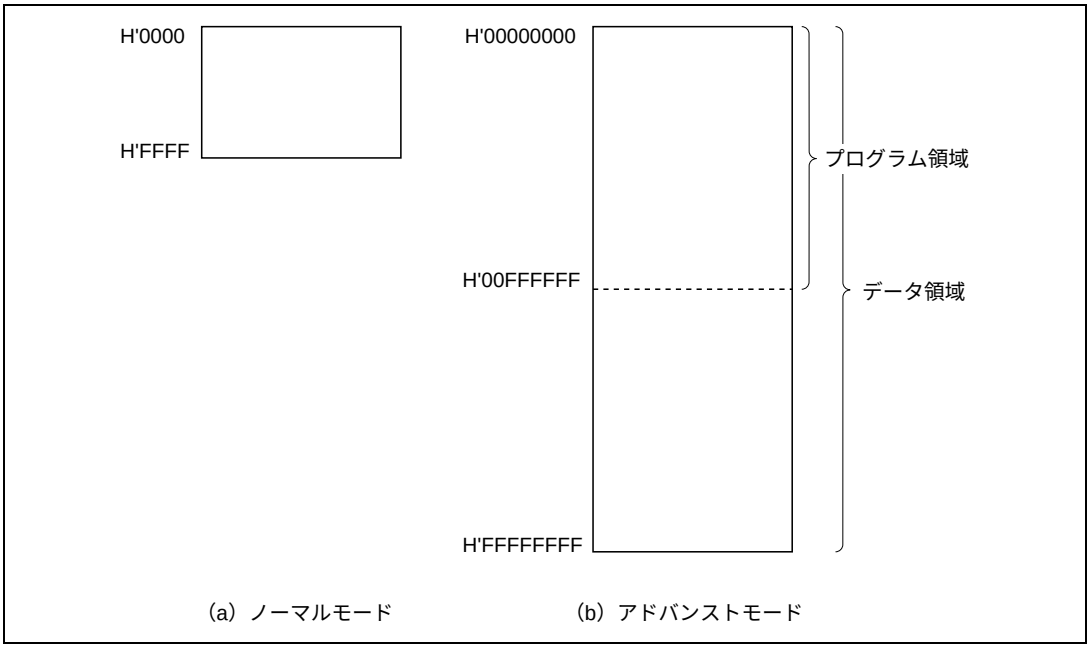


図 1.6 メモリマップ

1.4 レジスタ構成

1.4.1 概要

本 CPU の内部レジスタ構成を図 1.7 に示します。これらのレジスタは、汎用レジスタとコントロールレジスタの 2 つに分類することができます。

H8S/2000 CPU では、MAC レジスタをサポートしていません。

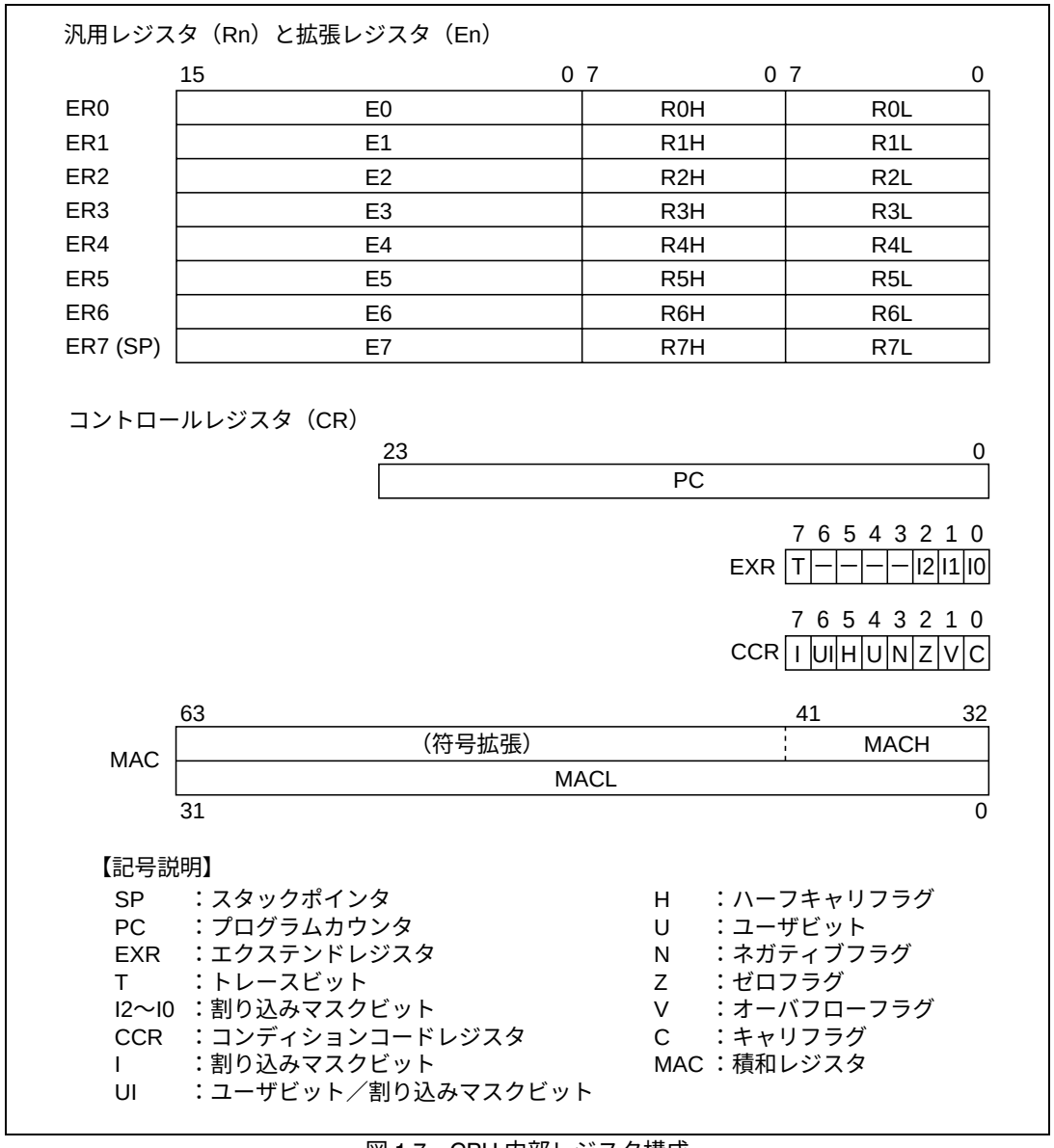


図 1.7 CPU 内部レジスタ構成

1.4.2 汎用レジスタ

本 CPU は、32 ビット長の汎用レジスタを 8 本持っています。汎用レジスタは、すべて同じ機能を持っており、アドレスレジスタとしてもデータレジスタとしても使用することができます。データレジスタとしては 32 ビット、16 ビットおよび 8 ビットレジスタとして使用できます。

アドレスレジスタおよび 32 ビットレジスタとしては、一括して汎用レジスタ ER（ER0～ER7）として使用します。

16 ビットレジスタとしては、汎用レジスタ ER を分割して汎用レジスタ E（E0～E7）、汎用レジスタ R（R0～R7）として使用します。これらは同等の機能を持っており、16 ビットレジスタを最大 16 本まで使用することができます。なお、汎用レジスタ E（E0～E7）を、特に拡張レジスタと呼ぶ場合があります。

8 ビットレジスタとしては、汎用レジスタ R を分割して汎用レジスタ RH（R0H～R7H）、汎用レジスタ RL（R0L～R7L）として使用します。これらは同等の機能を持っており、8 ビットレジスタを最大 16 本まで使用することができます。

汎用レジスタの使用方法を図 1.8 に示します。各レジスタ独立に使用方法を選択することができます。

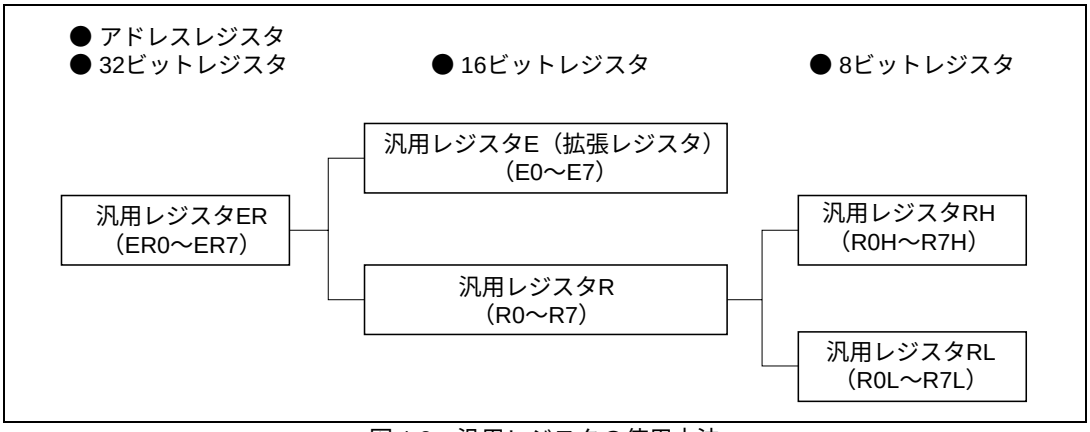


図 1.8　汎用レジスタの使用方法

汎用レジスタ ER7 には、汎用レジスタとしての機能に加えて、スタックポインタ（SP）としての機能が割り当てられており、例外処理やサブルーチン分岐などで暗黙的に使用されます。スタックの状態を図 1.9 に示します。

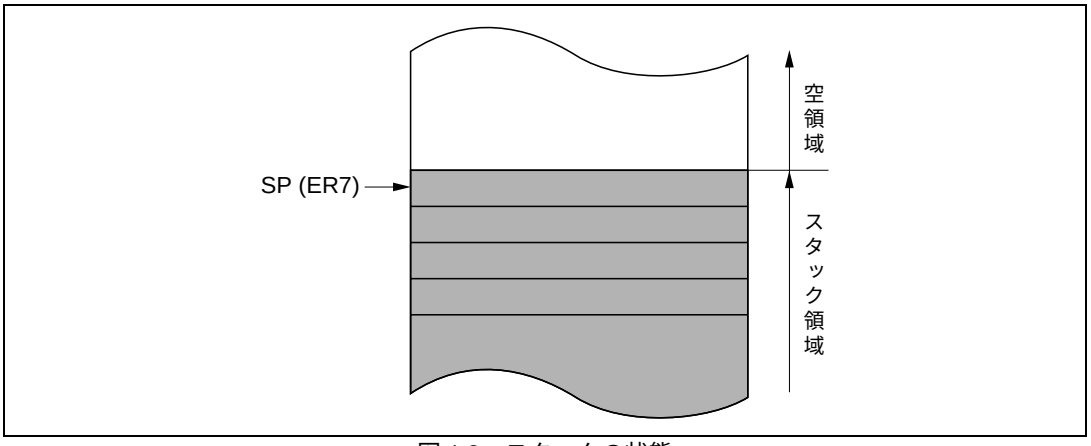


図 1.9　スタックの状態

1.4.3　コントロールレジスタ

コントロールレジスタには、24 ビットのプログラムカウンタ（PC）、8 ビットのエクステンドレジスタ（EXR）、8 ビットのコンディションコードレジスタ（CCR）、および 64 ビットの積和レジスタ（MAC：H8S/2600 CPU のみ）があります。

(1)　プログラムカウンタ（PC）

24 ビットのカウンタで、CPU が次に実行する命令のアドレスを示しています。CPU の命令は、すべて 2 バイト（ワード）を単位としているため、最下位ビットは無効です（命令コードのリード時には最下位ビットは 0 とみなされます）。

(2)　エクステンドレジスタ（EXR）

8 ビットのレジスタです。トレースビット（T）、割り込みマスクビット（I）を含む 8 ビットで構成されています。

ビット 7：トレースビット（T）

トレースモードか否かを指定します。本ビットが 0 にクリアされているときは命令を順次実行します。1 にセットされているときは 1 命令実行する毎にトレース例外処理を開始します。

ビット 6～4：リザーブビット

リザーブビットです。リードすると常に 1 が読み出されます。

ビット 2～0：割り込みマスクビット（I2～I0）

割り込み要求マスクレベル（0～7）を指定します。詳細は当該製品のハードウェアマニュアルを参照してください。

EXR は、LDC、STC、ANDC、ORC、XORC 命令で操作することができます。このうち STC を除く命令を実行した場合、実行終了後 3 ステートの間は、NMI を含めてすべての割り込みは受け付けられません。

(3) コンディションコードレジスタ（CCR）

8 ビットのレジスタで、CPU の内部状態を示しています。割り込みマスクビット（I）とハーフキャリ（H）、ネガティブ（N）、ゼロ（Z）、オーバフロー（V）、キャリ（C）の各フラグを含む 8 ビットで構成されています。

ビット 7：割り込みマスクビット（I）

本ビットが 1 にセットされると、割り込みがマスクされます。ただし、NMI は I ビットに関係なく受け付けられます。例外処理の実行が開始されたときに 1 にセットされます。

ビット 6：ユーザビット／割り込みマスクビット（UI）

ソフトウェア（LDC、STC、ANDC、ORC、XORC 命令）でリード／ライトできます。割り込みマスクビットとしても使用可能です。詳細は当該製品のハードウェアマニュアルを参照してください。

ビット 5：ハーフキャリフラグ（H）

ADD.B、ADDX.B、SUB.B、SUBX.B、CMP.B、NEG.B 命令の実行により、ビット 3 にキャリまたはボローが生じたとき 1 にセットされ、生じなかったとき 0 にクリアされます。また、ADD.W、SUB.W、CMP.W、NEG.W 命令の実行により、ビット 11 にキャリまたはボローが生じたとき、もしくは ADD.L、SUB.L、CMP.L、NEG.L 命令の実行により、ビット 27 にキャリまたはボローが生じたとき 1 にセットされ、生じなかったとき 0 にクリアされます。

ビット 4：ユーザビット（U）

ソフトウェア（LDC、STC、ANDC、ORC、XORC 命令）でリード／ライトできます。

ビット 3：ネガティブフラグ（N）

データの最上位ビットを符号ビットとみなし、最上位ビットの値を格納します。

ビット 2：ゼロフラグ（Z）

データがゼロのとき 1 にセットされ、ゼロ以外のとき 0 にクリアされます。

ビット 1：オーバフローフラグ（V）

算術演算命令の実行により、オーバフローが生じたとき 1 にセットされます。それ以外のとき 0 にクリアされます。

ビット 0：キャリフラグ（C）

演算の実行により、キャリが生じたとき 1 にセットされ、生じなかったとき 0 にクリアされます。キャリには次の種類があります。

（a）加算結果のキャリ

（b）減算結果のボロー

（c）シフト／ローテートのキャリ

また、キャリフラグには、ビットアキュムレータ機能があり、ビット操作命令で使用されます。

なお、命令によってはフラグが変化しない場合があります。

各命令ごとのフラグの変化については、2.2.1 以降の各命令の説明を参照してください。

CCR は、LDC、STC、ANDC、ORC、XORC 命令で操作することができます。また、N、Z、V、C の各フラグは、条件分岐命令（Bcc）で使用されます。

(4) 積和レジスタ（MAC）

H8S/2600 CPU のみでサポートしています。

64 ビットのレジスタで、積和演算結果を格納します。32 ビットの MACH、MACL から構成されます。MACH は下位 10 ビットが有効であり、上位は符号拡張されています。

1.4.4 CPU 内部レジスタの初期値

リセット例外処理によって、CPU 内部レジスタのうち、PC はベクタからロードすることにより初期化されます。また EXR の T ビットは 0 にクリアされ、EXR、CCR の I ビットは 1 にセットされますが、汎用レジスタと CCR の他のビットは初期化されません。SP（ER7）の初期値も不定です。したがって、リセット直後に、MOV.L 命令を使用して SP の初期化を行ってください。

1.5 データ構成

本 CPU は、1 ビット、4 ビット BCD、8 ビット（バイト）、16 ビット（ワード）、および 32 ビット（ロングワード）のデータを扱うことができます。

1 ビットデータはビット操作命令で扱われ、オペランドデータ（バイト）の第 n ビット（n＝0, 1, 2, ……7）という形式でアクセスされます。

なお、DAA および DAS の 10 進補正命令では、バイトデータは 2 桁の 4 ビット BCD データとなります。

1.5.1 汎用レジスタのデータ構成

汎用レジスタのデータ構成を図 1.10 に示します。

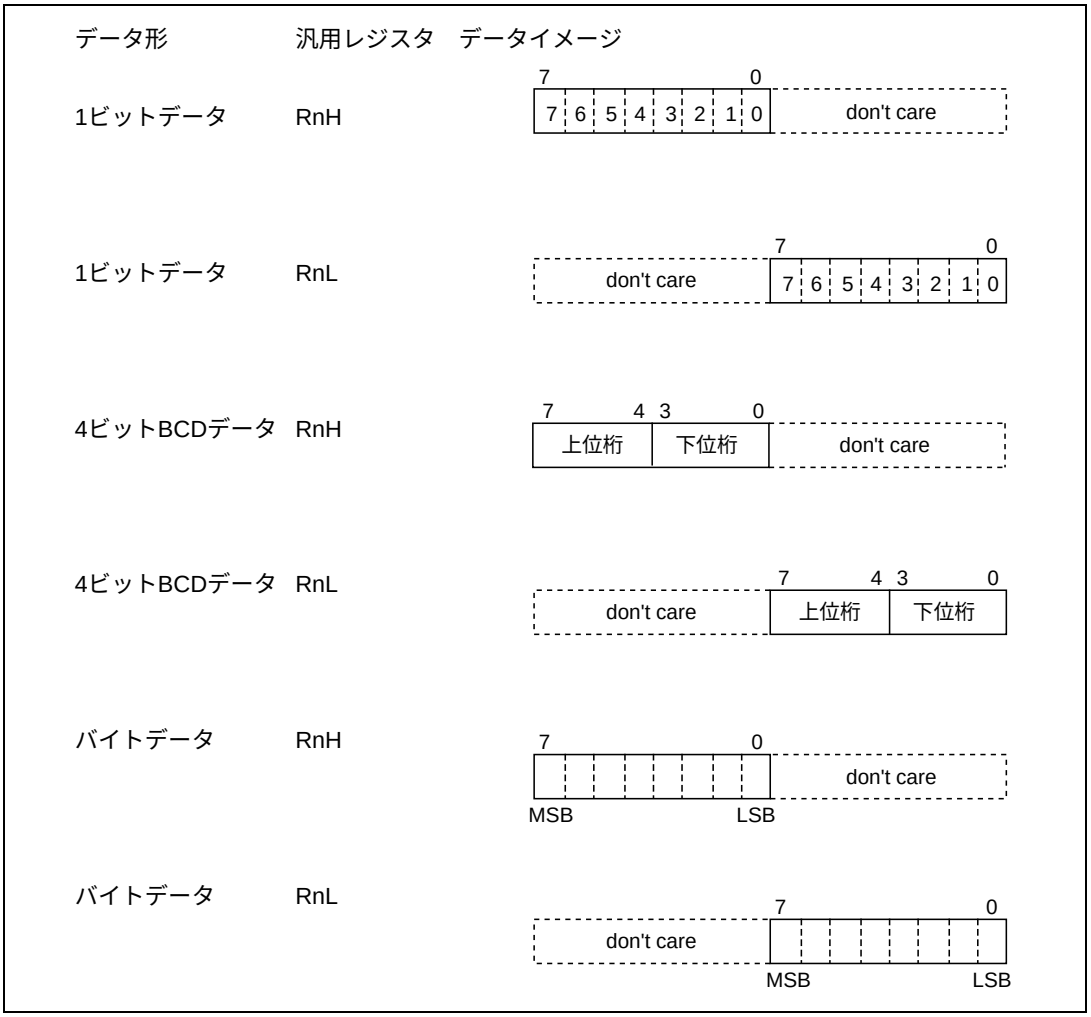


図 1.10 汎用レジスタのデータ構成（1）

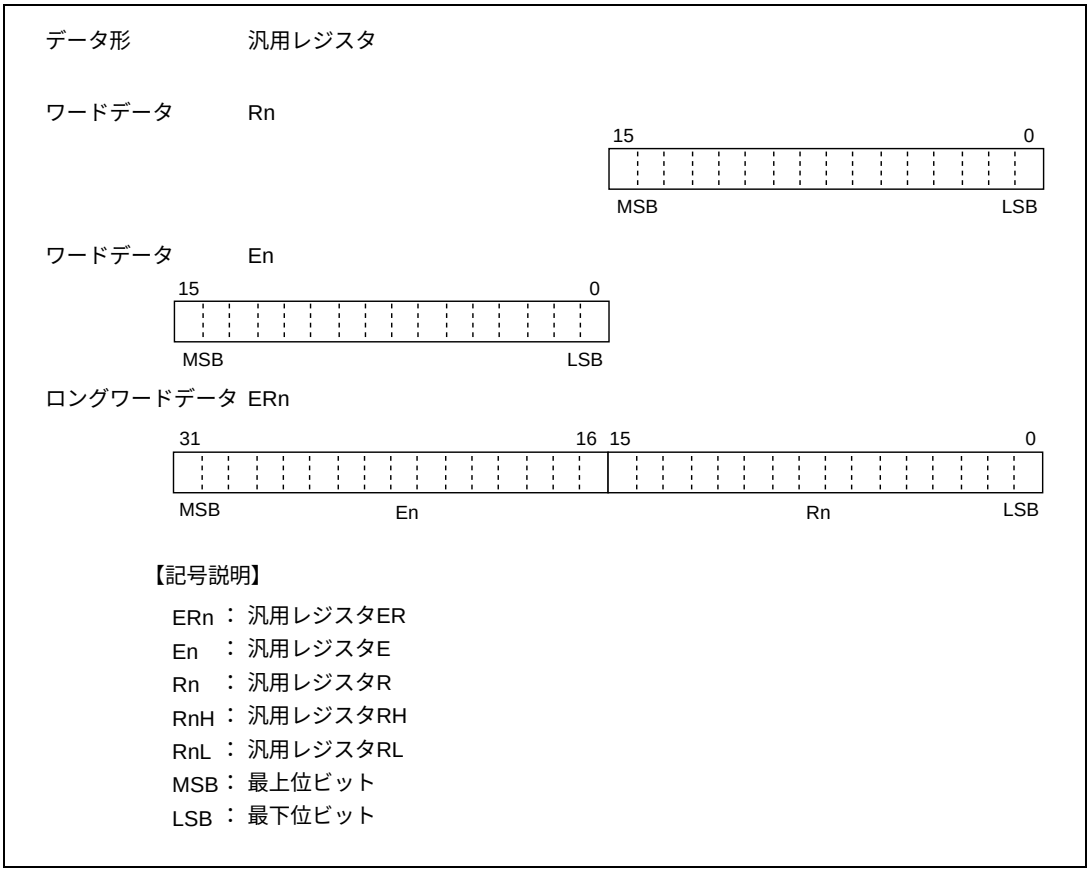


図 1.10　汎用レジスタのデータ構成（2）

1.5.2 メモリ上でのデータ構成

メモリ上でのデータ構成を図 1.11 に示します。

本 CPU は、メモリ上のワードデータ／ロングワードデータをアクセスすることができます。これらは、偶数番地から始まるデータに限定されます。奇数番地から始まるワードデータ／ロングワードデータをアクセスした場合、アドレスの最下位ビットは 0 とみなされ、1 番地前から始まるデータをアクセスします。この場合、アドレスエラーは発生しません。命令コードについても同様です。

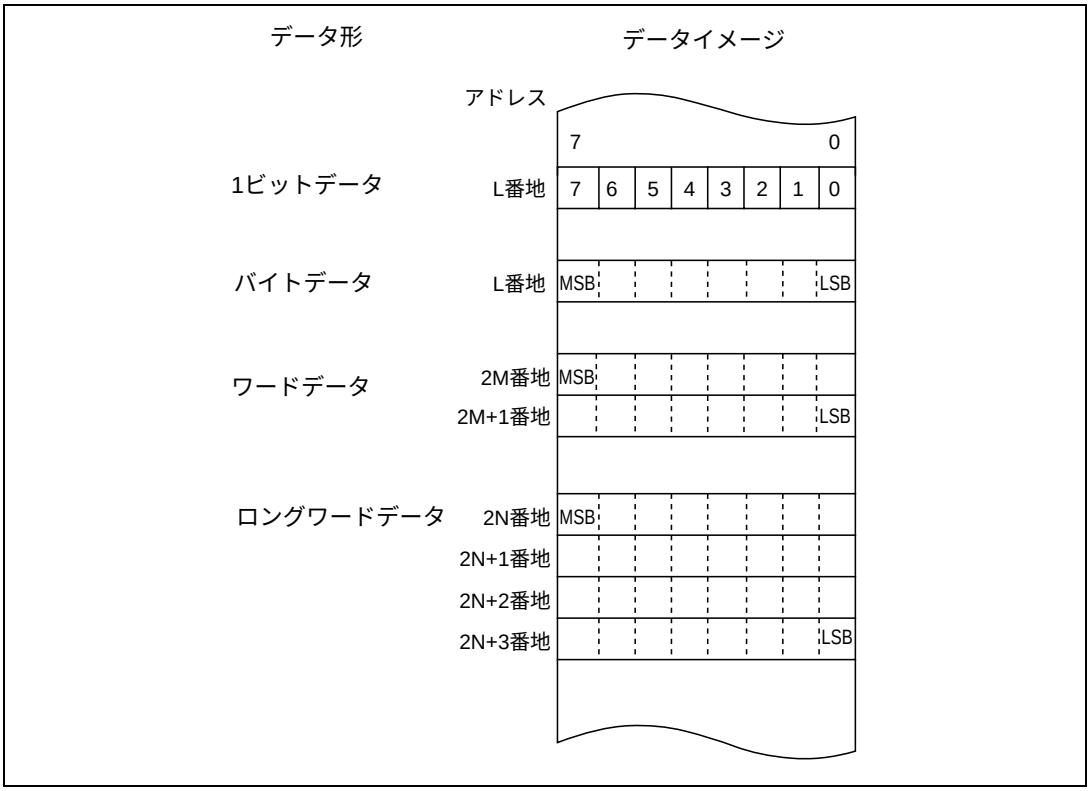


図 1.11　メモリ上でのデータ構成

なお、SP（ER7）をアドレスレジスタとしてスタックをアクセスするときは、必ずワードサイズまたはロングワードサイズでアクセスしてください。

1.6 命令セット

1.6.1 概要

H8S/2600 CPU の命令は合計 69 種類、H8S/2000 CPU の命令は合計 65 種類あります。命令セットは、各命令の持つ機能によって表 1.1 に示すように分類されます。各命令についての詳細は「2.2 各命令の説明」を参照してください。

表 1.1 命令の分類

分 類	命 令	サイズ	種類
転送命令	MOV	BWL	5
	POP*2, PUSH*2	WL	
	LDM, STM	L	
	MOVFPE, MOVTPE	B	
算術演算命令	ADD, SUB, CMP, NEG	BWL	19
	ADDX, SUBX, DAA, DAS	B	
	INC, DEC	BWL	
	ADDS, SUBS	L	
	MULXU, DIVXU, MULXS, DIVXS	BW	
	EXTU, EXTS	WL	
	TAS*4	B	
	MAC, LDMAC, STMAC, CLRMAC*1	−	4*1
論理演算命令	AND, OR, XOR, NOT	BWL	4
シフト命令	SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR	BWL	8
ビット操作命令	BSET, BCLR, BNOT, BTST, BLD, BILD, BST, BIST, BAND, BIAND, BOR, BIOR, BXOR, BIXOR	B	14
分岐命令	Bcc*3, JMP, BSR, JSR, RTS	−	5
システム制御命令	TRAPA, RTE, SLEEP, LDC, STC, ANDC, ORC, XORC, NOP	−	9
ブロック転送命令	EEPMOV	−	1

H8S/2600 CPU 合計 69 種類
H8S/2000 CPU 合計 65 種類

【注】 B：バイトサイズ W：ワードサイズ L：ロングワードサイズ

*1 MAC、LDMAC、STMAC、CLRMAC 命令は、H8S/2600 CPU のみでサポートしています。

*2 POP.W Rn、PUSH.W Rn は、それぞれ MOV.W @SP+,Rn、MOV.W Rn,@−SP と同一です。
また、POP.L ERn、PUSH.L ERn は、それぞれ MOV.L @SP+,ERn、MOV.L ERn,@−SP と同一です。

*3 Bcc は条件分岐命令の総称です。

*4 TAS 命令を使用する場合は、レジスタ ER0、ER1、ER4、ER5 を使用してください。

1.6.2 命令とアドレッシングモードの組み合わせ

本 CPU で使用できる命令とアドレッシングモードの組み合わせを表 1.2 に示します。

表 1.2 命令とアドレッシングモードの組み合わせ

機能	命令	アドレッシングモード													
		#xx	Rn	@ERn	@(d:16, ERn)	@(d:32, ERn)	@-ERn/@ERn+	@aa:8	@aa:16	@aa:24	@aa:32	@(d:8, PC)	@(d:16, PC)	@@aa:8	–
データ転送命令	MOV	BWL	BWL	BWL	BWL	BWL	BWL	B	BWL	–	BWL	–	–	–	–
	POP, PUSH	–	–	–	–	–	–	–	–	–	–	–	–	–	WL
	LDM, STM	–	–	–	–	–	–	–	–	–	–	–	–	–	L
	MOVFPE, MOVTPE	–	–	–	–	–	–	–	B	–	–	–	–	–	–
算術演算命令	ADD, CMP	BWL	BWL	–	–	–	–	–	–	–	–	–	–	–	–
	SUB	WL	BWL	–	–	–	–	–	–	–	–	–	–	–	–
	ADDX, SUBX	B	B	–	–	–	–	–	–	–	–	–	–	–	–
	ADDS, SUBS	–	L	–	–	–	–	–	–	–	–	–	–	–	–
	INC, DEC	–	BWL	–	–	–	–	–	–	–	–	–	–	–	–
	DAA, DAS	–	B	–	–	–	–	–	–	–	–	–	–	–	–
	MULXU, DIVXU	–	BW	–	–	–	–	–	–	–	–	–	–	–	–
	MULXS, DIVXS	–	BW	–	–	–	–	–	–	–	–	–	–	–	–
	NEG	–	BWL	–	–	–	–	–	–	–	–	–	–	–	–
	EXTU, EXTS	–	WL	–	–	–	–	–	–	–	–	–	–	–	–
	TAS*2	–	–	B	–	–	–	–	–	–	–	–	–	–	–
	MAC*1	–	–	–	–	–	○	–	–	–	–	–	–	–	–
	CLRMAC*1	–	–	–	–	–	–	–	–	–	–	–	–	–	○
	LDMAC*1, STMAC*1	–	L	–	–	–	–	–	–	–	–	–	–	–	–
論理演算命令	AND, OR, XOR	BWL	BWL	–	–	–	–	–	–	–	–	–	–	–	–
	NOT	–	BWL	–	–	–	–	–	–	–	–	–	–	–	–
シフト命令		–	BWL	–	–	–	–	–	–	–	–	–	–	–	–
ビット操作命令		–	B	B	–	–	–	B	B	–	B	–	–	–	–
分岐命令	Bcc, BSR	–	–	–	–	–	–	–	–	–	–	○	○	–	–
	JMP, JSR	–	–	–	–	–	–	–	–	○	–	–	–	○	–
	RTS	–	–	–	–	–	–	–	–	–	–	–	–	–	○
システム制御命令	TRAPA	–	–	–	–	–	–	–	–	–	–	–	–	–	○
	RTE	–	–	–	–	–	–	–	–	–	–	–	–	–	○
	SLEEP	–	–	–	–	–	–	–	–	–	–	–	–	–	○
	LDC	B	B	W	W	W	W	–	W	–	W	–	–	–	–
	STC	–	B	W	W	W	W	–	W	–	W	–	–	–	–
	ANDC, ORC, XORC	B	–	–	–	–	–	–	–	–	–	–	–	–	–
	NOP	–	–	–	–	–	–	–	–	–	–	–	–	–	○
ブロック転送命令		–	–	–	–	–	–	–	–	–	–	–	–	–	BW

【記号説明】
B ：バイト
W ：ワード
L ：ロングワード

【注】 *1 H8S/2600 CPUのみでサポートしています。
*2 TAS命令を使用する場合は、レジスタER0、ER1、ER4、ER5を使用してください。

1.6.3 命令の機能別一覧

表 1.3 に命令の機能別一覧を示します。また、以下に表 1.3 で使用される記号の意味を示します。

《オペレーションの記号》

Rd	汎用レジスタ（デスティネーション側）*
Rs	汎用レジスタ（ソース側）*
Rn	汎用レジスタ*
ERn	汎用レジスタ（32 ビットレジスタ）
MAC	積和レジスタ（32 ビットレジスタ）
(EAd)	デスティネーションオペランド
(EAs)	ソースオペランド
EXR	エクステンドレジスタ
CCR	コンディションコードレジスタ
N	CCR の N（ネガティブ）フラグ
Z	CCR の Z（ゼロ）フラグ
V	CCR の V（オーバフロー）フラグ
C	CCR の C（キャリ）フラグ
PC	プログラムカウンタ
SP	スタックポインタ
#IMM	イミディエイトデータ
disp	ディスプレースメント
＋	加算
－	減算
×	乗算
÷	除算
∧	論理積
∨	論理和
⊕	排他的論理和
→	転送
～	反転論理（論理的補数）
:8／:16／:24／:32	8／16／24／32 ビット長

【注】 * 汎用レジスタは、8 ビット（R0H～R7H、R0L～R7L）、16 ビット（R0～R7、E0～E7）、または 32 ビットレジスタ（ER0～ER7）です。

表 1.3 命令の機能別一覧

分類	命令	サイズ*1	機　能
データ転送命令	MOV	B／W／L	(EAs)→Rd、Rs→(EAd) 汎用レジスタと汎用レジスタ、または汎用レジスタとメモリ間でデータ転送します。また、イミディエイトデータを汎用レジスタに転送します。
	MOVFPE	B	(EAs)→Rd 外部メモリの内容（@aa:16 で指定）を E クロックに同期したタイミングで汎用レジスタに転送します。
	MOVTPE	B	Rs→(EAs) 汎用レジスタの内容を E クロックに同期したタイミングで外部メモリ（@aa:16 で指定）に転送します。
	POP	W／L	@SP+→Rn スタックから汎用レジスタへデータを復帰します。 POP.W Rn は MOV.W @SP+, Rn と、また、POP.L ERn は MOV.L @SP+, ERn と同一です。
	PUSH	W／L	Rn→@－SP 汎用レジスタの内容をスタックに退避します。 PUSH.W Rn は MOV.W Rn, @－SP と、また、PUSH.L ERn は MOV.L ERn, @－SP と同一です。
	LDM	L	@SP+→Rn（レジスタ群） スタックから複数の汎用レジスタへデータを復帰します。
	STM	L	Rn（レジスタ群）→@－SP 複数の汎用レジスタの内容をスタックに退避します。
算術演算命令	ADD SUB	B／W／L	Rd±Rs→Rd、Rd±#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の加減算を行います（バイトサイズでの汎用レジスタとイミディエイトデータ間の減算はできません。SUBX 命令または ADD 命令を使用してください）。
	ADDX SUBX	B	Rd±Rs±C→Rd、Rd±#IMM±C→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間のキャリ付きの加減算を行います。
	INC DEC	B／W／L	Rd±1→Rd、Rd±2→Rd 汎用レジスタに 1 または 2 を加減算します（バイトサイズで 1 の加減算のみ可能です）。
	ADDS SUBS	L	Rd±1→Rd、Rd±2→Rd、Rd±4→Rd 32 ビットレジスタに 1、2、または 4 を加減算します。
	DAA DAS	B	Rd(10 進補正) →Rd 汎用レジスタ上の加減算結果を CCR を参照して 4 ビット BCD データに補正します。

分類	命令	サイズ*1	機　能
算術演算命令	MULXU	B／W	Rd×Rs→Rd 汎用レジスタと汎用レジスタ間の符号なし乗算を行います。 8 ビット×8 ビット→16 ビット、16 ビット×16 ビット→32 ビットの乗算が可能です。
	MULXS	B／W	Rd×Rs→Rd 汎用レジスタと汎用レジスタ間の符号付き乗算を行います。 8 ビット×8 ビット→16 ビット、16 ビット×16 ビット→32 ビットの乗算が可能です。
	DIVXU	B／W	Rd÷Rs→Rd 汎用レジスタと汎用レジスタ間の符号なし除算を行います。 16 ビット÷8 ビット→商 8 ビット余り 8 ビット、 32 ビット÷16 ビット→商 16 ビット余り 16 ビットの除算が可能です。
	DIVXS	B／W	Rd÷Rs→Rd 汎用レジスタと汎用レジスタ間の符号付き除算を行います。 16 ビット÷8 ビット→商 8 ビット余り 8 ビット、 32 ビット÷16 ビット→商 16 ビット余り 16 ビットの除算が可能です。
	CMP	B／W／L	Rd－Rs、Rd－#IMM 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の比較を行い、その結果を CCR に反映します。
	NEG	B／W／L	0－Rd→Rd 汎用レジスタの内容の 2 の補数（算術的補数）をとります。
	EXTU	W／L	Rd(ゼロ拡張) →Rd 16 ビットレジスタの下位 8 ビットをワードサイズにゼロ拡張します。または、 32 ビットレジスタの下位 16 ビットをロングワードサイズにゼロ拡張します。
	EXTS	W／L	Rd(符号拡張) →Rd 16 ビットレジスタの下位 8 ビットをワードサイズに符号拡張します。または、 32 ビットレジスタの下位 16 ビットをロングワードサイズに符号拡張します。
	TAS	B	@ERd－0、1 → (<ビット 7>of @ERd)*2 メモリの内容をテストした後、最上位ビット（ビット 7）を 1 にセットします。
	MAC	－	(EAs)×(EAd)＋MAC→MAC メモリとメモリ間の符号付き乗算を行い、結果を積和レジスタに加算します。 16 ビット×16 ビット＋32 ビット→32 ビットの飽和演算、 16 ビット×16 ビット＋42 ビット→42 ビットの非飽和演算が可能です。 H8S/2600 CPU のみでサポートしています。
	CLRMAC	－	0→MAC 積和レジスタをゼロクリアします。 H8S/2600 CPU のみでサポートしています。
	LDMAC STMAC	L	Rs→MAC、MAC→Rd 汎用レジスタと積和レジスタ間でデータ転送します。 H8S/2600 CPU のみでサポートしています。

分類	命令	サイズ*1	機　能
論理演算命令	AND	B／W／L	Rd∧Rs→Rd、Rd∧#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理積をとります。
	OR	B／W／L	Rd∨Rs→Rd、Rd∨#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理和をとります。
	XOR	B／W／L	Rd⊕Rs→Rd、Rd⊕#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の排他的論理和をとります。
	NOT	B／W／L	～Rd→Rd 汎用レジスタの内容の 1 の補数（論理的補数）をとります。
シフト命令	SHAL SHAR	B／W／L	Rd(シフト処理) →Rd 汎用レジスタの内容を算術的にシフトします。 1 ビットまたは 2 ビットのシフトが可能です。
	SHLL SHLR	B／W／L	Rd(シフト処理) →Rd 汎用レジスタの内容を論理的にシフトします。 1 ビットまたは 2 ビットのシフトが可能です。
	ROTL ROTR	B／W／L	Rd(ローテート処理) →Rd 汎用レジスタの内容をローテートします。 1 ビットまたは 2 ビットのローテートが可能です。
	ROTXL ROTXR	B／W／L	Rd(ローテート処理) →Rd 汎用レジスタの内容をキャリフラグを含めてローテートします。 1 ビットまたは 2 ビットのローテートが可能です。
ビット操作命令	BSET	B	1→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを 1 にセットします。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容下位 3 ビットで指定します。
	BCLR	B	0→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを 0 にクリアします。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容下位 3 ビットで指定します。
	BNOT	B	～(<ビット番号>of<EAd>)→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを反転します。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容下位 3 ビットで指定されます。
	BTST	B	～(<ビット番号>of<EAd>)→Z 汎用レジスタまたはメモリのオペランドの指定された 1 ビットをテストし、ゼロフラグに反映します。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容下位 3 ビットで指定されます。

分類	命令	サイズ*1	機　能
ビット操作命令	BAND	B	C∧(<ビット番号>of<EAd>)→C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットとキャリフラグとの論理積をとり、結果をキャリフラグに格納します。
	BIAND	B	C∧〔～(<ビット番号>of<EAd>)〕→C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを反転し、キャリフラグとの論理積をとり、結果をキャリフラグに格納します。 ビット番号は、3 ビットのイミディエイトデータで指定されます。
	BOR	B	C∨(<ビット番号>of<EAd>) →C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットとキャリフラグとの論理和をとり、結果をキャリフラグに格納します。
	BIOR	B	C∨〔～(<ビット番号>of<EAd>)〕→C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを反転し、キャリフラグとの論理和をとり、結果をキャリフラグに格納します。 ビット番号は、3 ビットのイミディエイトデータで指定されます。
	BXOR	B	C⊕(<ビット番号>of<EAd>) →C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。
	BIXOR	B	C⊕〔～(<ビット番号>of<EAd>)〕→C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを反転し、キャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。 ビット番号は、3 ビットのイミディエイトデータで指定されます。
	BLD	B	(<ビット番号>of<EAd>) →C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットをキャリフラグに転送します。
	BILD	B	～(<ビット番号>of<EAd>) →C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを反転し、キャリフラグに転送します。 ビット番号は、3 ビットのイミディエイトデータで指定されます。
	BST	B	C→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された 1 ビットに、キャリフラグの内容を転送します。
	BIST	B	～C→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された 1 ビットに、キャリフラグを反転して転送します。 ビット番号は、3 ビットのイミディエイトデータで指定されます。

<table>
<tr><th>分類</th><th>命令</th><th>サイズ*1</th><th>機　能</th></tr>
<tr><td rowspan="5">分岐命令</td><td>Bcc</td><td>－</td><td>指定した条件が成立しているとき、指定されたアドレスへ分岐します。分岐条件を下表に示します。
<table>
<tr><th>ニーモニック</th><th>説　明</th><th>分岐条件</th></tr>
<tr><td>BRA (BT)</td><td>Always (True)</td><td>Always</td></tr>
<tr><td>BRN (BF)</td><td>Never (False)</td><td>Never</td></tr>
<tr><td>BHI</td><td>HIgh</td><td>C∨Z＝0</td></tr>
<tr><td>BLS</td><td>Low or Same</td><td>C∨Z＝1</td></tr>
<tr><td>BCC (BHS)</td><td>Carry Clear (High or Same)</td><td>C＝0</td></tr>
<tr><td>BCS (BLO)</td><td>Carry Set (LOw)</td><td>C＝1</td></tr>
<tr><td>BNE</td><td>Not Equal</td><td>Z＝0</td></tr>
<tr><td>BEQ</td><td>EQual</td><td>Z＝1</td></tr>
<tr><td>BVC</td><td>oVerflow Clear</td><td>V＝0</td></tr>
<tr><td>BVS</td><td>oVerflow Set</td><td>V＝1</td></tr>
<tr><td>BPL</td><td>PLus</td><td>N＝0</td></tr>
<tr><td>BMI</td><td>MInus</td><td>N＝1</td></tr>
<tr><td>BGE</td><td>Greater or Equal</td><td>N⊕V＝0</td></tr>
<tr><td>BLT</td><td>Less Than</td><td>N⊕V＝1</td></tr>
<tr><td>BGT</td><td>Greater Than</td><td>Z∨(N⊕V)＝0</td></tr>
<tr><td>BLE</td><td>Less or Equal</td><td>Z∨(N⊕V)＝1</td></tr>
</table></td></tr>
<tr><td>JMP</td><td>－</td><td>指定されたアドレスへ無条件に分岐します。</td></tr>
<tr><td>BSR</td><td>－</td><td>指定されたアドレスへサブルーチン分岐します。</td></tr>
<tr><td>JSR</td><td>－</td><td>指定されたアドレスへサブルーチン分岐します。</td></tr>
<tr><td>RTS</td><td>－</td><td>サブルーチンから復帰します。</td></tr>
</table>

分類	命令	サイズ*1	機　能
システム制御命令	TRAPA	－	命令トラップ例外処理を行います。
	RTE	－	例外処理ルーチンから復帰します。
	SLEEP	－	低消費電力状態に遷移します。
	LDC	B／W	(EAs)→CCR、(EAs)→EXR 汎用レジスタまたはメモリの内容を CCR、EXR に転送します。 また、イミディエイトデータを CCR、EXR に転送します。CCR、EXR は 8 ビットですが、メモリと CCR、EXR 間の転送はワードサイズで行われ、上位 8 ビットが有効になります。
	STC	B／W	CCR→(EAd)、EXR→(EAd) CCR、EXR の内容を汎用レジスタまたはメモリに転送します。CCR、EXR は 8 ビットですが、CCR、EXR とメモリ間の転送はワードサイズで行われ、上位 8 ビットが有効になります。
	ANDC	B	CCR∧#IMM→CCR、EXR∧#IMM→EXR CCR、EXR とイミディエイトデータの論理積をとります。
	ORC	B	CCR∨#IMM→CCR、EXR∨#IMM→EXR CCR、EXR とイミディエイトデータの論理和をとります。
	XORC	B	CCR⊕#IMM→CCR、EXR⊕#IMM→EXR CCR、EXR とイミディエイトデータの排他的論理和をとります。
	NOP	－	PC+2→PC PC のインクリメントだけを行います。
ブロック転送命令	EEPMOV.B	－	if R4L≠0 then Repeat @ER5+→@ER6+ R4L－1→R4L Until R4L＝0 else next;
	EEPMOV.W	－	if R4≠0 then Repeat @ER5+→@ER6+ R4－1→R4 Until R4＝0 else next; ブロック転送命令です。ER5 で示されるアドレスから始まり、R4L または R4 で指定されるバイト数のデータを、ER6 で示されるアドレスのロケーションへ転送します。転送終了後、次の命令を実行します。

【注】 *1　サイズはオペランドサイズを示します。
B　：バイト
W　：ワード
L　：ロングワード
*2　TAS 命令を使用する場合は、レジスタ ER0、ER1、ER4、ER5 を使用してください。

1.6.4 命令の基本フォーマット

本 CPU の命令は、2 バイト（ワード）を単位にしています。各命令はオペレーションフィールド（op）、レジスタフィールド（r）、EA 拡張部（EA）、およびコンディションフィールド（cc）から構成されています。

(1) オペレーションフィールド

命令の機能を表し、アドレッシングモードの指定、オペランドの処理内容を指定します。命令の先頭 4 ビットを必ず含みます。2 つのオペレーションフィールドを持つ場合もあります。

(2) レジスタフィールド

汎用レジスタを指定します。アドレスレジスタのとき 3 ビット、データレジスタのとき 3 ビットまたは 4 ビットです。2 つのレジスタフィールドを持つ場合、またはレジスタフィールドを持たない場合もあります。

(3) EA 拡張部

イミディエイトデータ、絶対アドレスまたはディスプレースメントを指定します。8 ビット、16 ビット、または 32 ビットです。

(4) コンディションフィールド

Bcc 命令の分岐条件を指定します。

図 1.12 に命令フォーマットの例を示します。

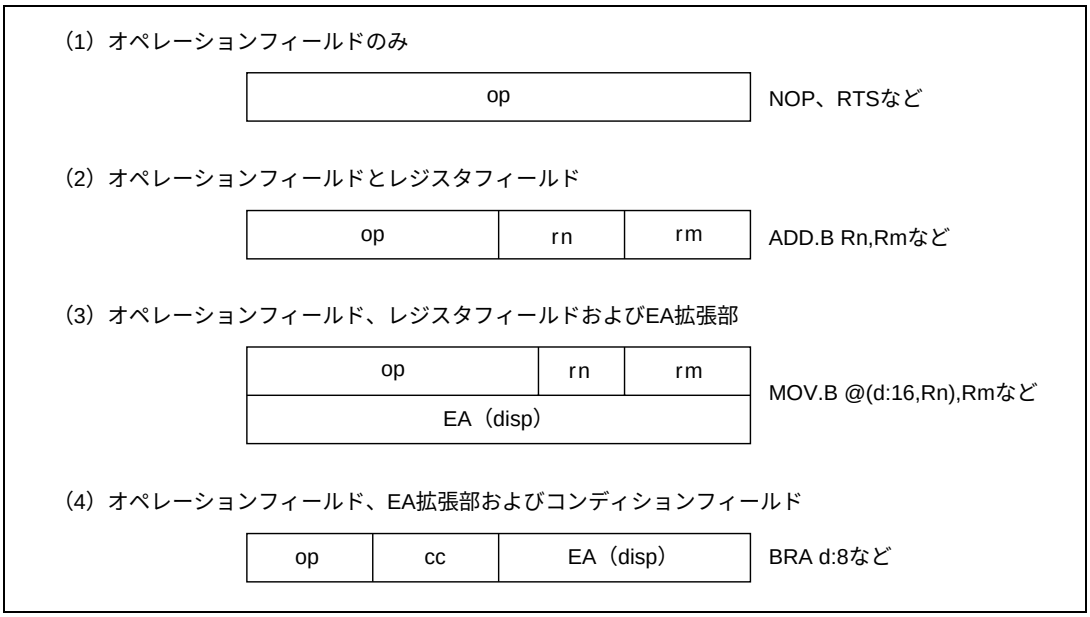


図 1.12　命令フォーマットの例

1.7 アドレッシングモードと実効アドレスの計算方法

1.7.1 アドレッシングモード

本CPUは表1.4に示すように、8種類のアドレッシングモードをサポートしています。命令ごとに、使用できるアドレッシングモードは異なります。

演算命令では、レジスタ直接、およびイミディエイトが使用できます。

転送命令では、プログラムカウンタ相対とメモリ間接を除くすべてのアドレッシングモードが使用できます。

また、ビット操作命令では、オペランドの指定にレジスタ直接、レジスタ間接、および絶対アドレスが使用できます。さらに、オペランド中のビット番号を指定するためにレジスタ直接（BSET、BCLR、BNOT、BTST の各命令）、およびイミディエイト（3 ビット）が独立して使用できます。

表 1.4 アドレッシングモード一覧表

No.	アドレッシングモード	記　号
1	レジスタ直接	Rn
2	レジスタ間接	@ERn
3	ディスプレースメント付きレジスタ間接	@(d:16,ERn)／@(d:32,ERn)
4	ポストインクリメントレジスタ間接 プリデクリメントレジスタ間接	@ERn＋ @－ERn
5	絶対アドレス	@aa:8／@aa:16／@aa:24／@aa:32
6	イミディエイト	#xx:8／#xx:16／#xx:32
7	プログラムカウンタ相対	@(d:8,PC)／@(d:16,PC)
8	メモリ間接	@@aa:8

(1) レジスタ直接　Rn

命令コードのレジスタフィールドで指定されるレジスタ（8 ビット、16 ビットまたは 32 ビット）がオペランドとなります。

8 ビットレジスタとしては R0H～R7H、R0L～R7L を指定可能です。

16 ビットレジスタとしては R0～R7、E0～E7 を指定可能です。

32 ビットレジスタとしては ER0～ER7 を指定可能です。

(2) レジスタ間接　@ERn

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ERn）の内容をアドレスとしてメモリ上のオペランドを指定します。

プログラム領域としては、下位 24 ビットが有効になり、上位 8 ビットはすべて 0（H'00）とみなされます。

(3) ディスプレースメント付きレジスタ間接　@（d:16,ERn）／@（d:32,ERn）

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ERn）の内容に命令コード中に含まれる 16 ビットディスプレースメントまたは 32 ビットディスプレースメントを加算した内容をアドレスとしてメモリ上のオペランドを指定します。加算に際して、16 ビットディスプレースメントは符号拡張されます。

(4) ポストインクリメントレジスタ間接 @ERn＋／プリデクリメントレジスタ間接 @－ERn

(a) ポストインクリメントレジスタ間接 @ERn＋

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ERn）の内容をアドレスとしてメモリ上のオペランドを指定します。その後、アドレスレジスタの内容に 1、2 または 4 が加算され、加算結果がアドレスレジスタに格納されます。バイトサイズでは 1、ワードサイズでは 2、ロングワードサイズでは 4 がそれぞれ加算されます。ワードサイズまたはロングワードサイズのとき、アドレスレジスタの内容が偶数となるようにしてください。

(b) プリデクリメントレジスタ間接 @－ERn

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ERn）の内容から 1、2 または 4 を減算した内容をアドレスとしてメモリ上のオペランドを指定します。その後、減算結果がアドレスレジスタに格納されます。バイトサイズでは 1、ワードサイズでは 2、ロングワードサイズでは 4 がそれぞれ減算されます。ワードサイズまたはロングワードサイズのとき、アドレスレジスタの内容が偶数になるようにしてください。

(5) 絶対アドレス @aa:8／@aa:16／@aa:24／@aa:32

命令コード中に含まれる絶対アドレスで、メモリ上のオペランドを指定します。

絶対アドレスは 8 ビット（@aa:8）、16 ビット（@aa:16）、24 ビット（@aa:24）、または 32 ビット（@aa:32）です。

データ領域としては、8 ビット（@aa:8）、16 ビット（@aa:16）、または 32 ビット（@aa:32）を使用します。8 ビット絶対アドレスの場合、上位 24 ビットはすべて 1（H'FFFF）となります。16 ビット絶対アドレスの場合、上位 16 ビットは符号拡張されます。32 ビット絶対アドレスの場合、全アドレス空間をアクセスできます。

プログラム領域としては 24 ビット（@aa:24）を使用します。上位 8 ビットはすべて 0（H'00）となります。

絶対アドレスのアクセス範囲を表 1.5 に示します。

表 1.5 絶対アドレスのアクセス範囲

絶対アドレス		ノーマルモード	アドバンストモード
データ領域	8 ビット（@aa:8）	H'FF00～H'FFFF	H'FFFFFF00～H'FFFFFFFF
	16 ビット（@aa:16）	H'0000～H'FFFF	H'00000000～H'00007FFF、H'FFFF8000～H'FFFFFFFF
	32 ビット（@aa:32）		H'00000000～H'FFFFFFFF
プログラム領域	24 ビット（@aa:24）		H'00000000～H'00FFFFFF

アクセス範囲の詳細については当該製品のハードウェアマニュアルを参照してください。

(6) イミディエイト #xx:8／#xx:16／#xx:32

命令コード中に含まれる 8 ビット（#xx:8）、16 ビット（#xx:16）、または 32 ビット（#xx:32）のデータを直接オペランドとして使用します。

なお、ADDS、SUBS、INC、DEC 命令では、イミディエイトデータが命令コード中に暗黙的に含まれます。ビット操作命令では、ビット番号を指定するための 3 ビットのイミディエイトデータが、命令コード中に含まれる場合があります。また、TRAPA 命令ではベクタアドレスを指定するための 2 ビットのイミディエイトデータが、命令コードの中に含まれます。

(7) プログラムカウンタ相対 @（d:8,PC）／@（d:16,PC）

Bcc、BSR 命令で使用されます。PC の内容で指定される 24 ビットのアドレスに、命令コード中に含まれる 8 ビット、または 16 ビットディスプレースメントを加算して 24 ビットの分岐アドレスを生成します。加算に際して、ディスプレースメントは 24 ビットに符号拡張されます。加算結果は下位 24 ビットが有効になり、上位 8 ビットはすべて 0（H'00）とみなされます。また加算される PC の内容は次の命令の先頭アドレスとなっていますので、分岐可能範囲は分岐命令に対して－126～＋128 バイト（－63～＋64 ワード）または－32766～＋32768 バイト（－16383～＋16384 ワード）です。このとき、加算結果が偶数となるようにしてください。

(8) メモリ間接 @@aa:8

JMP、JSR 命令で使用されます。命令コード中に含まれる 8 ビット絶対アドレスでメモリ上のオペランドを指定し、この内容を分岐アドレスとして分岐します。

8 ビット絶対アドレスの上位のビットはすべて 0 となりますので、分岐アドレスを格納できるのは 0～255（ノーマルモードのとき H'0000～H'00FF、アドバンストモードのとき H'00000000～H'000000FF）番地です。

ノーマルモードの場合は、メモリ上のオペランドはワードサイズで指定し、16 ビットの分岐アドレスを生成します。

また、アドバンストモードの場合は、メモリ上のオペランドはロングワードサイズで指定します。このうち先頭の 1 バイトはすべて 0（H'00）とみなされます。

ただし、分岐アドレスを格納可能なアドレスの先頭領域は例外処理ベクタ領域と共通になっていますから注意してください。詳細は当該製品のハードウェアマニュアルを参照してください。

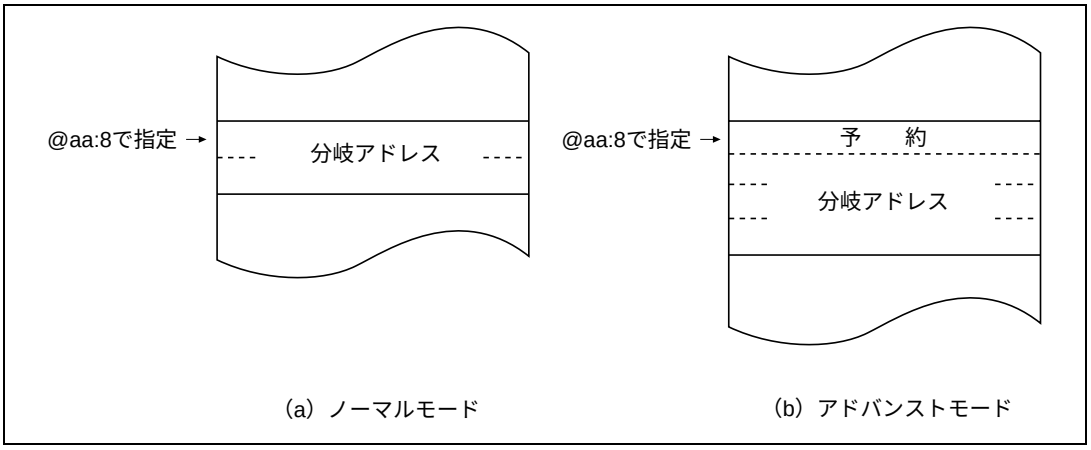


図 1.13　メモリ間接による分岐アドレスの指定

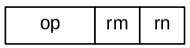
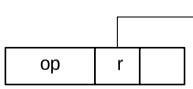
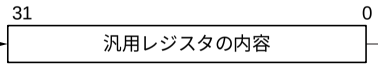
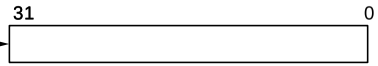
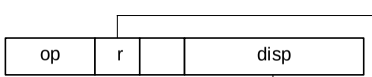
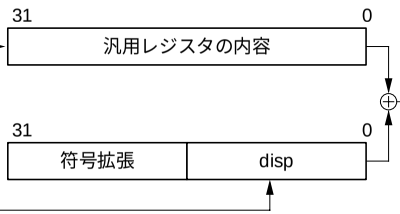
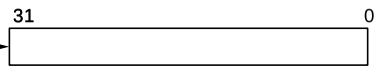
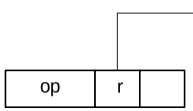
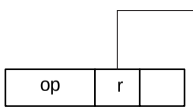
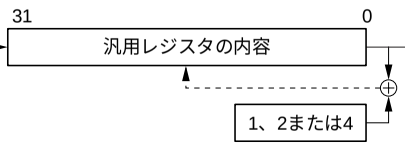
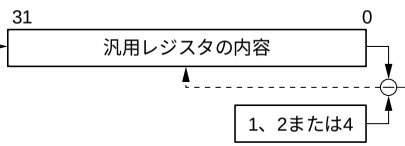
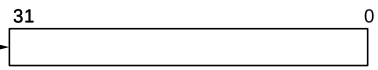
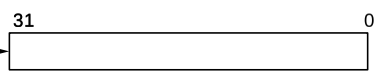
ワードサイズ、ロングワードサイズでメモリを指定する場合、および分岐アドレスを指定する場合に奇数アドレスを指定すると、最下位ビットは 0 とみなされ、1 番地前から始まるデータまたは命令コードをアクセスします（「1.5.2　メモリ上でのデータ構成」を参照してください）。

1.7.2　実効アドレスの計算方法

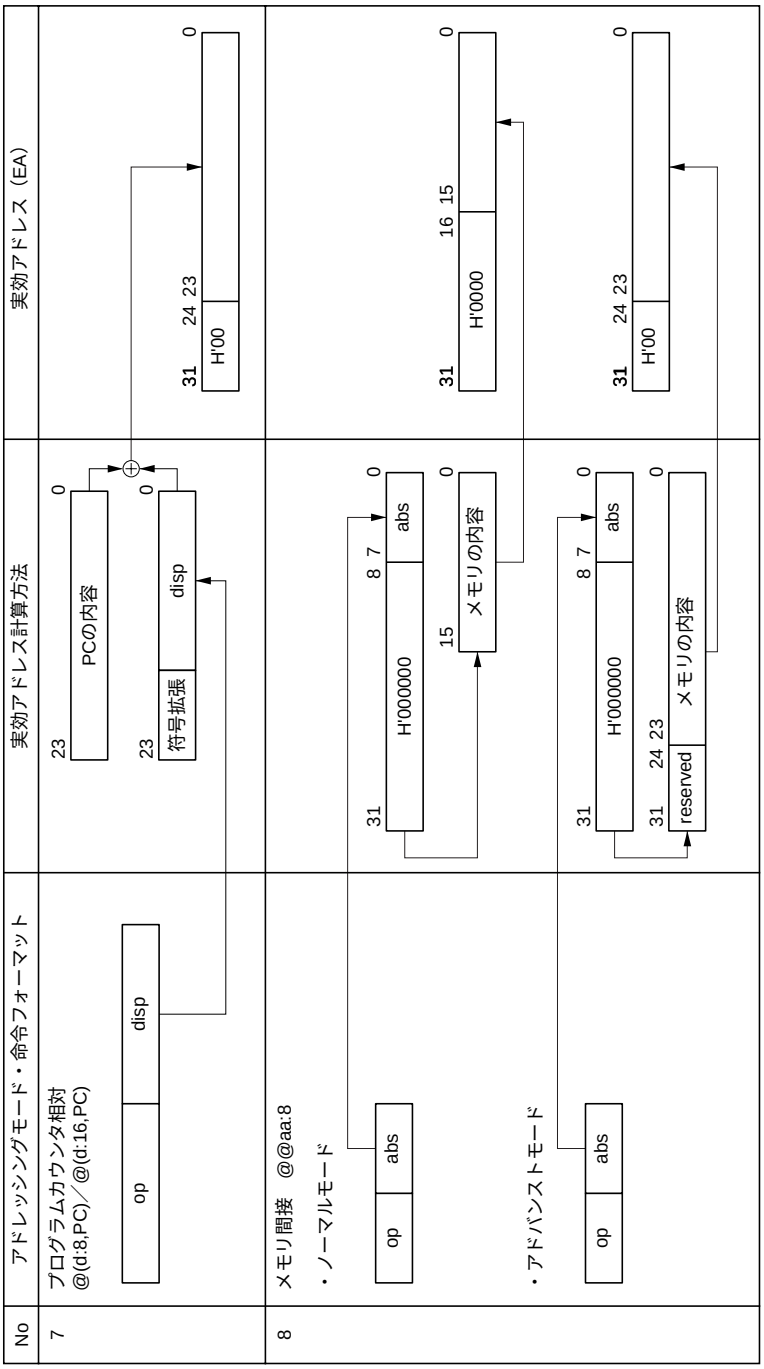
各アドレッシングモードにおける実効アドレス（EA：Effective Address）の計算法を表 1.6 に示します。

ノーマルモードの場合、実効アドレスの上位 8 ビットは無視され、16 ビットのアドレスとなります。

表1.6　実効アドレスの計算方法

No	アドレッシングモード・命令フォーマット	実効アドレス計算方法	実効アドレス（EA）
1	レジスタ直接（Rn） 		オペランドは汎用レジスタの内容です。
2	レジスタ間接（@ERn） 		
3	ディスプレースメント付きレジスタ間接 @(d:16,ERn) ／ @(d:32,ERn) 		
4	ポストインクリメントレジスタ間接／プリデクリメントレジスタ間接 ・ポストインクリメントレジスタ間接　@ERn＋  ・プリデクリメントレジスタ間接　@－ERn 	  オペランドサイズ／加減算される値 バイト：1 ワード：2 ロングワード：4	 

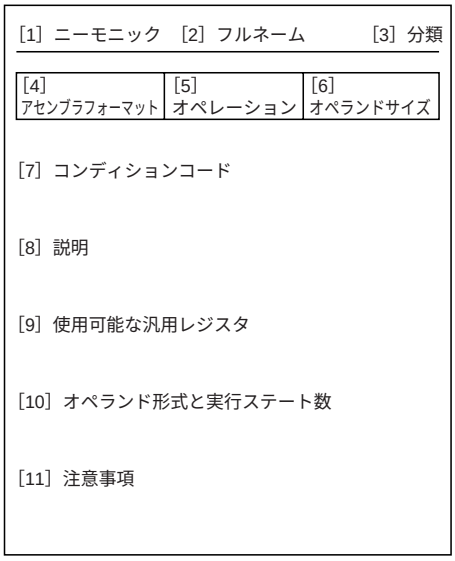
No	アドレッシングモード・命令フォーマット	実効アドレス計算方法	実効アドレス（EA）
5	絶対アドレス @aa:8 op abs @aa:16 op abs @aa:24 op abs @aa:32 op abs		31 8 7 0 H'FFFFFF 31 16 15 0 符号拡張 31 24 23 0 H'00 31 0
6	イミディエイト　#xx:8／#xx:16／#xx:32 op IMM		オペランドはイミディエイトデータです。


No
アドレッシングモード・命令フォーマット
実効アドレス計算方法
実効アドレス（EA）
7
プログラムカウンタ相対
@(d:8,PC)／@(d:16,PC)
op
disp
23
0
PCの内容
符号拡張
disp
31
24 23
0
H'00
8
メモリ間接 @@aa:8
・ノーマルモード
op
abs
H'000000
メモリの内容
16 15
H'0000
・アドバンストモード
reserved

2. 各命令の説明

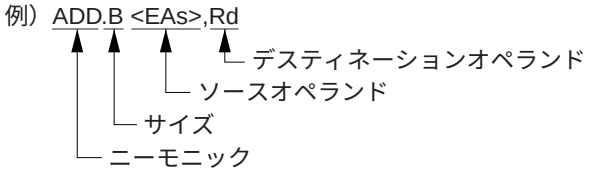
2.1 表と記号の説明

「2.2 各命令の説明」の表の見方について説明します。なお、同一の命令についての説明でも、複数ページにわたっているものがありますから注意してください。



[1] ニーモニック（フルネーム）	：命令のニーモニックを示します。
[2] フルネーム	：命令のフルネームを示します。
[3] 分類	：命令の機能を示します。
[4] アセンブラフォーマット	：命令のアセンブラフォーマットを示します。（2.1.1を参照）
[5] オペレーション	：命令の操作を簡潔に示します。（2.1.2を参照）
[6] オペランドサイズ	：使用できるオペランドのサイズを示します。
[7] コンディションコード	：命令実行後のコンディションコードレジスタ（CCR）の各ビットの変化を示します。（2.1.3を参照）
[8] 説明	：命令の動作について詳細に説明します。
[9] 使用可能な汎用レジスタ	：命令コードのレジスタフィールドで指定できるレジスタを示します。
[10] オペランド形式と実行ステート数	：命令のアドレッシングモード、インストラクションフォーマット、ならびに実行ステート数を示します。
[11] 注意事項	：命令を実行する上での注意事項などを示します。

2.1.1 アセンブラフォーマット



オペランドサイズは、バイト（B）、ワード（W）、ロングワード（L）があります。命令によって、使用できるオペランドサイズは異なります。

<EA>は、複数のアドレッシングモードが使用できることを示します。本 CPU がサポートするアドレッシングモードは、次の 8 種類です。実効アドレスの計算方法については「1.7　アドレッシングモードと実効アドレスの計算方法」を参照してください。

記　号	アドレッシングモード
Rn	レジスタ直接
@ERn	レジスタ間接
@(d:16,ERn)／@(d:32,ERn)	ディスプレースメント（16／32 ビット）付レジスタ間接
@ERn+／@-ERn	ポストインクリメントレジスタ間接／プリデクリメントレジスタ間接
@aa:8／@aa:16／@aa:24／@aa:32	絶対アドレス（8／16／24／32 ビット）
#xx:8／#xx:16／#xx:32	イミディエイト（8／16／32 ビット）
@(d:8,PC)／@(d:16,PC)	プログラムカウンタ相対（8／16 ビット）
@@aa:8	メモリ間接

なお、:8／:16／:24／:32 は省略することができます。特に絶対アドレス、およびディスプレースメントについては:8／:16／:24／:32 を省略すると、値の範囲に応じてアセンブラが最適化を行います。

詳細は「H8S, H8/300 シリーズ　クロスアセンブラ　ユーザーズマニュアル」を参照してください。

【注】 #xx (:2)、#xx (:3)の:2、:3 は指定可能なビット長を表しています。
アセンブラ表記上は (:2)、(:3)を付けないでください。
（例）　TRAPA　#3

2.1.2 オペレーション

オペレーションの欄で使用されている記号と動作記号を以下に示します。

Rd	汎用レジスタ（デスティネーション側）*
Rs	汎用レジスタ（ソース側）*
Rn	汎用レジスタ*
ERn	汎用レジスタ（32 ビットレジスタ）
MAC	積和レジスタ（32 ビットレジスタ）
(EAd)	デスティネーションオペランド
(EAs)	ソースオペランド
EXR	エクステンドレジスタ
CCR	コンディションコードレジスタ
N	CCR の N（ネガティブ）フラグ
Z	CCR の Z（ゼロ）フラグ
V	CCR の V（オーバフロー）フラグ
C	CCR の C（キャリ）フラグ
PC	プログラムカウンタ
SP	スタックポインタ
#IMM	イミディエイトデータ
disp	ディスプレースメント
＋	加算
－	減算
×	乗算
÷	除算
∧	論理積
∨	論理和
⊕	排他的論理和
→	左辺のオペランドから右辺のオペランドへの転送、 または左辺の状態から右辺の状態への遷移
～	反転論理（論理的補数）
( ) <>	オペランドの内容
:8／:16／:24／:32	8／16／24／32 ビット長

【注】 * 汎用レジスタは、8 ビット（R0H～R7H、R0L～R7L）、16 ビット（R0～R7、E0～E7）または 32 ビット（ER0～ER7）です。

2.1.3 コンディションコード

コンディションコードの欄で使用されている記号を以下に示します。

記　号	内　　容
↕	実行結果にしたがって変化することを表します。
＊	不確定であることを表します（値を保証しません）。
0	常に 0 にクリアされることを表します。
1	常に 1 にセットされることを表します。
−	実行結果に影響を受けないことを表します。
△	条件によって異なります。注意事項を参照してください。

コンディションコードの変化の詳細については「2.8　コンディションコードの変化」を参照してください。

2.1.4 インストラクションフォーマット

インストラクションフォーマットの欄で使用されている記号を以下に示します。

記　号	内　　容
IMM	イミディエイトデータ（2、3、8、16、32 ビット）
abs	絶対アドレス（8、16、24、32 ビット）
disp	ディスプレースメント（8、16、32 ビット）
rs、rd、rn	レジスタフィールド（4 ビット） rs、rd、rn はそれぞれオペランドの形式の Rs、Rd、Rn に対応
ers、erd、ern	レジスタフィールド（3 ビット） ers、erd、ern はオペランドの形式の ERs、ERd、ERn に対応

2.1.5 レジスタの指定方法

(1) アドレスレジスタの指定

汎用レジスタをアドレスレジスタとして使用するとき（@ERn、@(d:16,ERn)、@(d:32,ERn)、@ERn＋、@－ERn）は 3 ビットのレジスタフィールド（ers、erd）で指定されます。

(2) データレジスタの指定

汎用レジスタは、データレジスタとして使用するとき、32 ビット、16 ビットまたは 8 ビットレジスタです。

32 ビットレジスタとして使用するとき、3 ビットレジスタフィールド（ers、erd、ern）で指定されます。

16 ビットレジスタとして使用するとき、4 ビットのレジスタフィールド（rs、rd、rn）で指定されます。このときレジスタフィールドの下位 3 ビットがレジスタ番号を示し、上位 1 ビットが 1 のとき汎用レジスタ En が指定され、0 のとき汎用レジスタ Rn が指定されます。

また、8 ビットレジスタとして使用するとき、4 ビットのレジスタフィールド（rs、rd、rn）で指定されます。また、このときレジスタフィールドの下位 3 ビットがレジスタ番号を示し、上位 1 ビットが 1 のとき汎用レジスタ RnL が指定され、0 のとき汎用レジスタ RnH が指定されます。

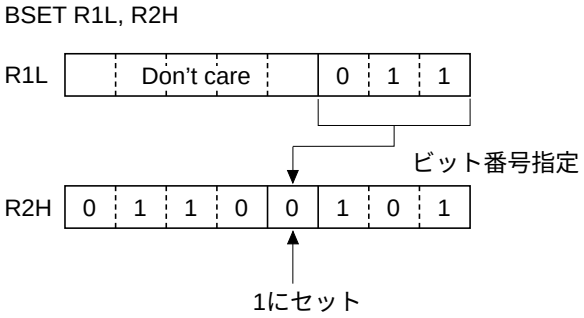
この対応を以下に示します。

アドレスレジスタ 32 ビットレジスタ		16 ビットレジスタ		8 ビットレジスタ	
レジスタ フィールド	汎用レジスタ	レジスタ フィールド	汎用レジスタ	レジスタ フィールド	汎用レジスタ
000	ER0	0000	R0	0000	R0H
001	ER1	0001	R1	0001	R1H
⋮	⋮	⋮	⋮	⋮	⋮
111	ER7	0111	R7	0111	R7H
		1000	E0	1000	R0L
		1001	E1	1001	R1L
		⋮	⋮	⋮	⋮
		1111	E7	1111	R7L

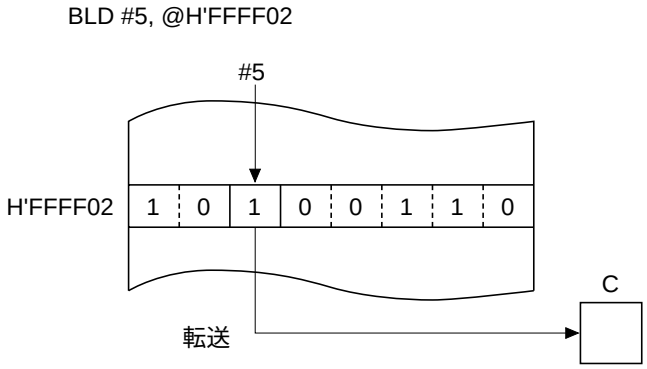
2.1.6 ビット操作命令におけるビットデータのアクセス方法

ビットデータは、レジスタまたはメモリ上のオペランドデータ（バイト）の第 n ビット（n＝0, 1, 2, 3, …7）という形でアクセスされます。このとき、ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容（下位 3 ビットのみ有効）によって指定されます。

（例 1）R2H のビット 3 を 1 にセットする場合



（例 2）H'FFFF02 番地のビット 5 をビットアキュムレータに転送する場合



なお、ビット操作命令のオペランドサイズおよびアドレス形式は、レジスタまたはメモリ上のオペランドデータについて示しています。

2.2 各命令の説明

2.2.1 以降に各命令について説明します。

2.2.1 ADD (B) ADD binary 2 進加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADD.B ＜EAs＞, Rd	Rd+（EAs）→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	↕	−	↕	↕	↕	↕

H：ビット 3 にキャリが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 7 にキャリが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）とソースオペランドを加算し、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
Rs ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	ADD.B	#xx:8,Rd	8	rd	IMM				1
レジスタ直接	ADD.B	Rs,Rd	0	8	rs	rd			1

2.2.2 ADD (W) ADD binary 2 進加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADD.W ＜EAs＞, Rd	Rd＋（EAs）→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	↕	−	↕	↕	↕	↕

H：ビット 11 にキャリが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 15 にキャリが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）とソースオペランドを加算し、結果を 16 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd ：R0～R7、E0～E7
Rs ：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	ADD.W	#xx:16,Rd	7	9	1	rd	IMM		2
レジスタ直接	ADD.W	Rs,Rd	0	9	rs	rd			1

2.2.3 ADD (L) ADD binary 2 進加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADD.L ＜EAs＞, ERd	ERd＋（EAs）→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	↕	−	↕	↕	↕	↕

H：ビット 27 にキャリが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 31 にキャリが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）とソースオペランドを加算し、結果を 32 ビットレジスタ ERd に格納します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	第5バイト	第6バイト	
イミディエイト	ADD.L	#xx:32,ERd	7	A	1	0 erd	IMM				3
レジスタ直接	ADD.L	ERs,ERd	0	A	1 ers	0 erd					1

2.2.4 ADDS ADD with Sign extention アドレスデータ 2 進加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADDS #1, ERd	ERd＋1→ERd	ロングワード
ADDS #2, ERd	ERd＋2→ERd	
ADDS #4, ERd	ERd＋4→ERd	

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）に 1、2 または 4 を加算します。ADD 命令とは異なり、コンディションコードは実行前の値を保持します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット							実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	ADDS	#1,ERd	0	B	0	0	erd			1
レジスタ直接	ADDS	#2,ERd	0	B	8	0	erd			1
レジスタ直接	ADDS	#4,ERd	0	B	9	0	erd			1

2.2.5 ADDX ADD with eXtend carry キャリ付加算

アセンブラフォーマット	オペレーション	オペランドサイズ
ADDX ＜EAs＞, Rd	Rd＋（EAs）＋C→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	↕	−	↕	↕	↕	↕

H：ビット 3 にキャリが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき実行前の値が保持され、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 7 にキャリが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）とソースオペランドとキャリフラグの値を加算し、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
Rs ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	ADDX	#xx:8,Rd	9	rd	IMM				1
レジスタ直接	ADDX	Rs,Rd	0	E	rs	rd			1

2.2.6 AND (B) ADD logical 論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
AND.B ＜EAs＞, Rd	Rd∧（EAs）→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）とソースオペランドの論理積をとり、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H
Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	AND.B	#xx:8,Rd	E	rd	IMM				1
レジスタ直接	AND.B	Rs,Rd	1	6	rs	rd			1

2.2.7 AND (W) AND logical 論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
AND.W ＜EAs＞, Rd	Rd∧（EAs）→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）とソースオペランドの論理積をとり、結果を 16 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7
Rs：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	AND.W	#xx:16,Rd	7	9	6	rd	IMM		2
レジスタ直接	AND.W	Rs,Rd	6	6	rs	rd			1

2.2.8 AND (L) AND logical 論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
AND.L ＜EAs＞, ERd	ERd∧（EAs）→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）とソースオペランドの論理積をとり、結果を 32 ビットレジスタ ERd に格納します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット												実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト			第5バイト	第6バイト		
イミディエイト	AND.L	#xx:32,ERd	7	A	6	0 erd	IMM								3
レジスタ直接	AND.L	ERs,ERd	0	1	F	0	6	6	0 ers	0 erd					2

2.2.9 ANDC AND Control register CCR との論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
ANDC #xx：8, CCR	CCR∧＃IMM→CCR	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

I ：実行結果の対応するビットの値が格納されます。
UI：実行結果の対応するビットの値が格納されます。
H ：実行結果の対応するビットの値が格納されます。
U ：実行結果の対応するビットの値が格納されます。
N ：実行結果の対応するビットの値が格納されます。
Z ：実行結果の対応するビットの値が格納されます。
V ：実行結果の対応するビットの値が格納されます。
C ：実行結果の対応するビットの値が格納されます。

(2) 説明

CCR の内容とイミディエイトデータの論理積をとり、結果を CCR に格納します。
なお、本命令の実行終了時点では、NMI を含めてすべての割り込みは受け付けられません。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット					実行ステート数
			第1バイト		第2バイト	第3バイト	第4バイト	
イミディエイト	ANDC	#xx:8,CCR	0	6	IMM			1

2.2.10 ANDC AND Control register EXR との論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
ANDC #xx：8, EXR	EXR∧＃IMM→EXR	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H ：実行前の値が保持されます。
N ：実行前の値が保持されます。
Z ：実行前の値が保持されます。
V ：実行前の値が保持されます。
C ：実行前の値が保持されます。

(2) 説明

EXR の内容とイミディエイトデータの論理積をとり、結果を EXR に格納します。

なお、本命令の実行終了後３ステートの間は、NMI を含めてすべての割り込みは受け付けられません。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット							実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト	
イミディエイト	ANDC	#xx:8,EXR	0	1	4	1	0	6	IMM	2

2.2.11 BAND Bit AND ビット論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
BAND #xx：3, <EAd>	C∧（<ビット番号>of<EAd>）→C	バイト

(1) コンディションコード

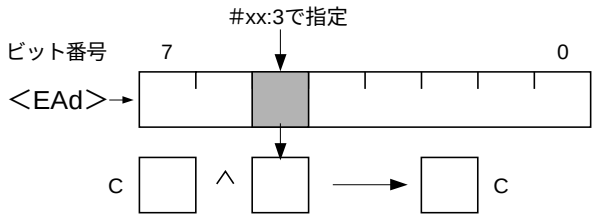
I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	↕

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行結果が格納されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットとキャリフラグとの論理積をとり、結果をキャリフラグに格納します。

ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット												実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト	第7バイト	第8バイト	
レジスタ直接	BAND	#xx:3,Rd	7	6	0	IMM	rd									1
レジスタ間接	BAND	#xx:3,@ERd	7	C	0	erd	0	7	6	0	IMM	0				3
絶対アドレス	BAND	#xx:3,@aa:8	7	E	abs			7	6	0	IMM	0				3
絶対アドレス	BAND	#xx:3,@aa:16	6	A	1	0	abs				7	6	0	IMM	0	4
絶対アドレス	BAND	#xx:3,@aa:32	6	A	3	0	abs							7	6	0	IMM	0	5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.12 Bcc Branch conditionary 条件付分岐

アセンブラフォーマット	オペレーション	オペランドサイズ
Bcc disp └→コンディションフィールド	If condition is true, then PC＋disp→PC else next ;	—

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：演算前の値が保持されます。
N：演算前の値が保持されます。
Z：演算前の値が保持されます。
V：演算前の値が保持されます。
C：演算前の値が保持されます。

(2) 説明

コンディションフィールド（cc）で指定された条件が成立していると、PC にディスプレースメントを加えたアドレスに分岐し、条件が不成立の場合は次の命令を実行します。アドレス計算に用いられる PC の値は本命令の直後の命令の先頭アドレスです。ディスプレースメントは符号付き 8 ビットまたは 16 ビットデータで、分岐できる範囲は本命令に対して－126～＋128、－32766～＋32768 バイトです。

ニーモニック	説明	cc	条件	符号と条件の対応*
BRA(BT)	Always(True)	0000	True	
BRN(BF)	Never(False)	0001	False	
BHI	HIgh	0010	C∨Z=0	X＞Y　符号なし
BLS	Low or Same	0011	C∨Z=1	X≦Y　符号なし
BCC(BHS)	Carry Clear(High or Same)	0100	C=0	X≧Y　符号なし
BCS(BLO)	Carry Set(LOw)	0101	C=1	X＜Y　符号なし
BNE	Not Equal	0110	Z=0	X≠Y　符号なし・あり
BEQ	EQual	0111	Z=1	X＝Y　符号なし・あり
BVC	oVerflow Clear	1000	V=0	
BVS	oVerflow Set	1001	V=1	
BPL	PLus	1010	N=0	
BMI	MInus	1011	N=1	
BGE	Greater or Equal	1100	N⊕V=0	X≧Y　符号あり
BLT	Less Than	1101	N⊕V=1	X＜Y　符号あり
BGT	Greater Than	1110	Z∨(N⊕V)=0	X＞Y　符号あり
BLE	Less or Equal	1111	Z∨(N⊕V)=1	X≦Y　符号あり

【注】 * 直前の命令が CMP 命令のとき、X は汎用レジスタの内容（デスティネーションオペランド）、Y はソースオペランドです。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		
プログラムカウンタ相対	BRA(BT)	d:8	4	0	disp						2
		d:16	5	8	0	0	disp				3
プログラムカウンタ相対	BRN(BF)	d:8	4	1	disp						2
		d:16	5	8	1	0	disp				3
プログラムカウンタ相対	BHI	d:8	4	2	disp						2
		d:16	5	8	2	0	disp				3
プログラムカウンタ相対	BLS	d:8	4	3	disp						2
		d:16	5	8	3	0	disp				3
プログラムカウンタ相対	BCC(BHS)	d:8	4	4	disp						2
		d:16	5	8	4	0	disp				3
プログラムカウンタ相対	BCS(BLO)	d:8	4	5	disp						2
		d:16	5	8	5	0	disp				3
プログラムカウンタ相対	BNE	d:8	4	6	disp						2
		d:16	5	8	6	0	disp				3
プログラムカウンタ相対	BEQ	d:8	4	7	disp						2
		d:16	5	8	7	0	disp				3
プログラムカウンタ相対	BVC	d:8	4	8	disp						2
		d:16	5	8	8	0	disp				3
プログラムカウンタ相対	BVS	d:8	4	9	disp						2
		d:16	5	8	9	0	disp				3
プログラムカウンタ相対	BPL	d:8	4	A	disp						2
		d:16	5	8	A	0	disp				3
プログラムカウンタ相対	BMI	d:8	4	B	disp						2
		d:16	5	8	B	0	disp				3
プログラムカウンタ相対	BGE	d:8	4	C	disp						2
		d:16	5	8	C	0	disp				3
プログラムカウンタ相対	BLT	d:8	4	D	disp						2
		d:16	5	8	D	0	disp				3
プログラムカウンタ相対	BGT	d:8	4	E	disp						2
		d:16	5	8	E	0	disp				3
プログラムカウンタ相対	BLE	d:8	4	F	disp						2
		d:16	5	8	F	0	disp				3

(4) 注意事項

1. 分岐先アドレスは、必ず偶数になるようにしてください。
2. BRA、BRN、BCC、BCSの機械語はそれぞれBT、BF、BHS、BLOと同一です。

2.2.13 BCLR Bit CLeaR ビットクリア

アセンブラフォーマット	オペレーション	オペランドサイズ
BCLR #xx：3, <EAd> BCLR Rn, <EAd>	0→（<ビット番号>of<EAd>）	バイト

(1) コンディションコード

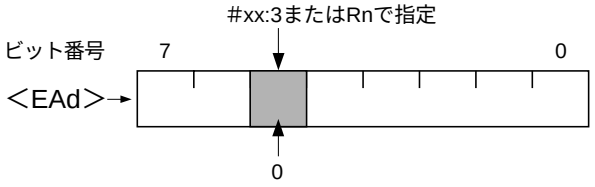
I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットを 0 にクリアします。ビット番号は、3 ビットのイミディエイトデータまたは 8 ビットレジスタ Rn の内容の下位 3 ビットで指定されます。

指定された 1 ビットのテストは行いません（コンディションコードは変化しません）。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7
Rn ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット																実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		
レジスタ直接	BCLR	#xx:3,Rd	7	2	0	IMM	rd												1
レジスタ間接	BCLR	#xx:3,@ERd	7	D	0	erd	0	7	2	0	IMM	0							4
絶対アドレス	BCLR	#xx:3,@aa:8	7	F	abs			7	2	0	IMM	0							4
絶対アドレス	BCLR	#xx:3,@aa:16	6	A	1		8	abs				7	2	0	IMM	0			5
絶対アドレス	BCLR	#xx:3,@aa:32	6	A	3		8	abs							7	2	0	IMM	6
レジスタ直接	BCLR	Rn,Rd	6	2	m		rd												1
レジスタ間接	BCLR	Rn,@ERd	7	D	0	erd	0	6	2	rn	0								4
絶対アドレス	BCLR	Rn,@aa:8	7	F	abs			6	2	rn	0								4
絶対アドレス	BCLR	Rn,@aa:16	6	A	1		8	abs				6	2	rn	0				5
絶対アドレス	BCLR	Rn,@aa:32	6	A	3		8	abs							6	2	rn	0	6

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.14 BIAND Bit Invert AND ビット論理積

アセンブラフォーマット	オペレーション	オペランドサイズ
BIAND #xx：3, <EAd>	C∧〔～（<ビット番号>of<EAd>）〕→C	バイト

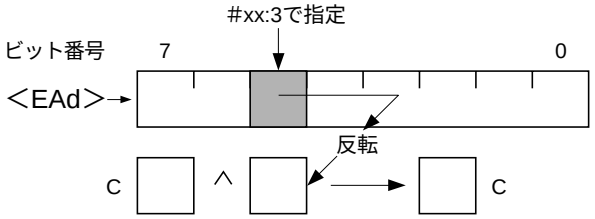
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	↕

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行結果が格納されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットを反転し、これとキャリフラグとの論理積をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット														実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		
レジスタ直接	BIAND	#xx:3,Rd	7	6	1	IMM	rd												1
レジスタ間接	BIAND	#xx:3,@ERd	7	C	0	erd	0	7	6	1	IMM	0							3
絶対アドレス	BIAND	#xx:3,@aa:8	7	E	abs			7	6	1	IMM	0							3
絶対アドレス	BIAND	#xx:3,@aa:16	6	A	1	0	abs				7	6	1	IMM	0				4
絶対アドレス	BIAND	#xx:3,@aa:32	6	A	3	0	abs								7	6	1	IMM	0	5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.15 BILD Bit Invert LoaD ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BILD #xx：3, <EAd>	～（<ビット番号>of<EAd>）→C	バイト

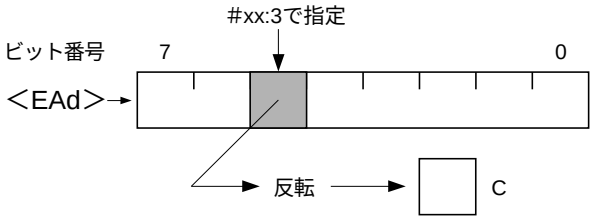
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	↕

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：指定ビットの内容が反転されて格納されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットを反転し、これをキャリフラグに転送します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット																実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		
レジスタ直接	BILD	#xx:3,Rd	7	7	1	IMM	rd												1
レジスタ間接	BILD	#xx:3,@ERd	7	C	0	erd	0	7	7	1	IMM	0							3
絶対アドレス	BILD	#xx:3,@aa:8	7	E	abs			7	7	1	IMM	0							3
絶対アドレス	BILD	#xx:3,@aa:16	6	A	1	0	abs					7	7	1	IMM	0			4
絶対アドレス	BILD	#xx:3,@aa:32	6	A	3	0	abs								7	7	1	IMM	5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.16 BIOR Bit Invert inclusive OR ビット論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
BIOR #xx：3, <EAd>	C∨〔～（<ビット番号>of<EAd>）〕→C	バイト

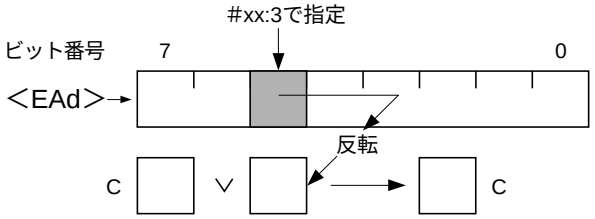
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	↕

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行結果が格納されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットを反転し、これとキャリフラグとの論理和をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット												実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト	第7バイト	第8バイト	
レジスタ直接	BIOR	#xx:3,Rd	7	4	1:IMM	rd										1
レジスタ間接	BIOR	#xx:3,@ERd	7	C	0:erd	0	7	4	1:IMM	0						3
絶対アドレス	BIOR	#xx:3,@aa:8	7	E	abs		7	4	1:IMM	0						3
絶対アドレス	BIOR	#xx:3,@aa:16	6	A	1	0	abs				7	4	1:IMM	0		4
絶対アドレス	BIOR	#xx:3,@aa:32	6	A	3	0	abs							7	4	1:IMM	0	5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.17 BIST Bit Invert STore ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BIST #xx：3, <EAd>	～C→（<ビット番号>of<EAd>）	バイト

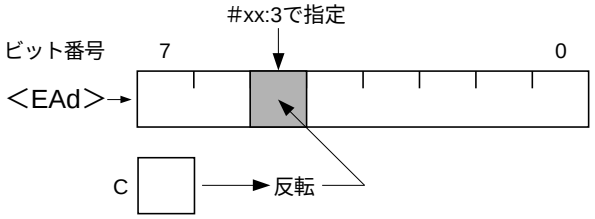
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	–

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットのロケーションに、キャリフラグの内容を反転して転送します。ビット番号は、3 ビットのイミディエイトデータで指定されます。なお、デスティネーションオペランドの指定されない他のビットの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット																実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		
レジスタ直接	BIST	#xx:3,Rd	6	7	1:IMM	rd													1
レジスタ間接	BIST	#xx:3,@ERd	7	D	0:erd	0	6	7	1:IMM	0									4
絶対アドレス	BIST	#xx:3,@aa:8	7	F	abs		6	7	1:IMM	0									4
絶対アドレス	BIST	#xx:3,@aa:16	6	A	1	8	abs				6	7	1:IMM	0					5
絶対アドレス	BIST	#xx:3,@aa:32	6	A	3	8	abs								6	7	1:IMM	0	6

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.18 BIXOR Bit Invert eXclusive OR ビット排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
BIXOR #xx：3, <EAd>	C ⊕〔～（<ビット番号>of<EAd>）〕→C	バイト

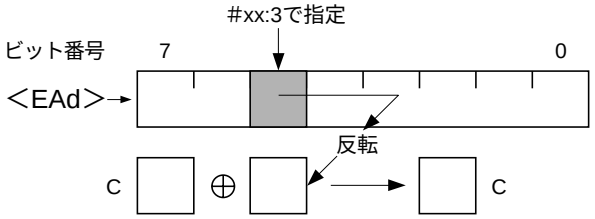
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	↕

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行結果が格納されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットを反転し、これとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット														実行ステート数
			第1バイト			第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト	
レジスタ直接	BIXOR	#xx:3,Rd	7	5	1	IMM	rd												1
レジスタ間接	BIXOR	#xx:3,@ERd	7	C	0	erd	0	7	5	1	IMM	0							3
絶対アドレス	BIXOR	#xx:3,@aa:8	7	E	abs			7	5	1	IMM	0							3
絶対アドレス	BIXOR	#xx:3,@aa:16	6	A	1	0		abs				7	5	1	IMM	0			4
絶対アドレス	BIXOR	#xx:3,@aa:32	6	A	3	0		abs								7	5	1 IMM	0 5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.19 BLD Bit LoaD ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BLD #xx：3, <EAd>	（<ビット番号>of<EAd>）→C	バイト

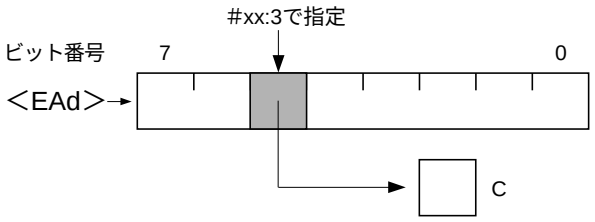
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	↕

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：指定ビットの内容が格納されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットをキャリフラグに転送します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット														実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト	
レジスタ直接	BLD	#xx:3,Rd	7	7	0	IMM	rd											1
レジスタ間接	BLD	#xx:3,@ERd	7	C	0	erd	0	7	7	0	IMM	0						3
絶対アドレス	BLD	#xx:3,@aa:8	7	E	abs			7	7	0	IMM	0						3
絶対アドレス	BLD	#xx:3,@aa:16	6	A	1	0	abs				7	7	0	IMM	0			4
絶対アドレス	BLD	#xx:3,@aa:32	6	A	3	0	abs								7	7	0 IMM 0	5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.20 BNOT Bit NOT ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BNOT #xx：3, <EAd> BNOT Rn , <EAd>	～(<ビット番号>of<EAd>) →(<ビット番号>of<EAd>)	バイト

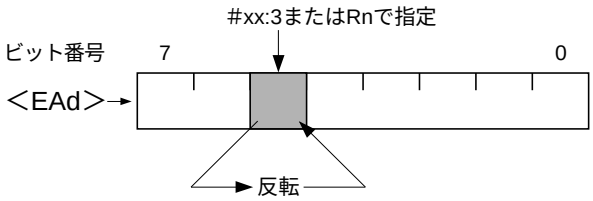
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	–

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットを反転します。ビット番号は、3 ビットのイミディエイトデータまたは 8 ビットレジスタ Rn の内容の下位 3 ビットで指定されます。
指定された 1 ビットのテストは行いません（コンディションコードは変化しません）。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7
Rn ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット														実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		
レジスタ直接	BNOT	#xx:3,Rd	7	1	0	IMM	rd												1
レジスタ間接	BNOT	#xx:3,@ERd	7	D	0	erd	0	7	1	0	IMM	0							4
絶対アドレス	BNOT	#xx:3,@aa:8	7	F	abs			7	1	0	IMM	0							4
絶対アドレス	BNOT	#xx:3,@aa:16	6	A	1		8	abs				7	1	0	IMM	0			5
絶対アドレス	BNOT	#xx:3,@aa:32	6	A	3		8	abs							7	1	0	IMM	6
レジスタ直接	BNOT	Rn,Rd	6	1	rn		rd												1
レジスタ間接	BNOT	Rn,@ERd	7	D	0	erd	0	6	1	rn	0								4
絶対アドレス	BNOT	Rn,@aa:8	7	F	abs			6	1	rn	0								4
絶対アドレス	BNOT	Rn,@aa:16	6	A	1		8	abs				6	1	rn	0				5
絶対アドレス	BNOT	Rn,@aa:32	6	A	3		8	abs							6	1	rn	0	6

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.21 BOR Bit inclusive OR ビット論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
BOR #xx：3, <EAd>	C∨（<ビット番号>of<EAd>）→C	バイト

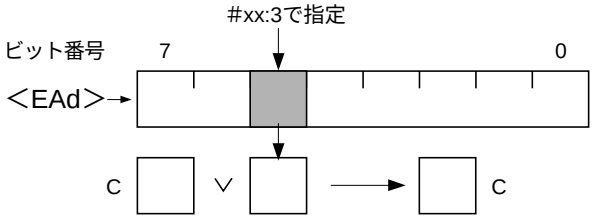
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	↕

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行結果が格納されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットとキャリフラグとの論理和をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット										実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト			
レジスタ直接	BOR	#xx:3,Rd	7 4	0 IMM rd									1
レジスタ間接	BOR	#xx:3,@ERd	7 C	0 erd 0	7 4	0 IMM 0							3
絶対アドレス	BOR	#xx:3,@aa:8	7 E	abs	7 4	0 IMM 0							3
絶対アドレス	BOR	#xx:3,@aa:16	6 A	1 0	abs		7 4	0 IMM 0					4
絶対アドレス	BOR	#xx:3,@aa:32	6 A	3 0	abs				7 4	0 IMM 0			5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.22 BSET Bit SET ビットセット

アセンブラフォーマット	オペレーション	オペランドサイズ
BSET #xx：3, <EAd> BSET Rn , <EAd>	1→（<ビット番号>of<EAd>）	バイト

(1) コンディションコード

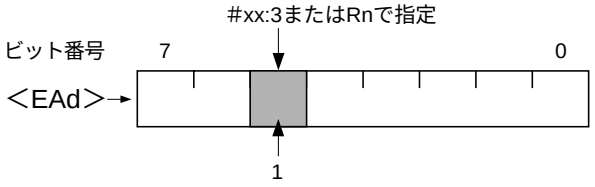
I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットを 1 にセットします。ビット番号は、3 ビットのイミディエイトデータまたは 8 ビットレジスタ Rn の内容の下位 3 ビットで指定されます。

指定された 1 ビットのテストは行いません（コンディションコードは変化しません）。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd：ER0～ER7
Rn ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット																実行ステート数
			第1バイト			第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト	
レジスタ直接	BSET	#xx:3,Rd	7	0	0	IMM	rd												1
レジスタ間接	BSET	#xx:3,@ERd	7	D	0	erd	0	7	0	0	IMM	0							4
絶対アドレス	BSET	#xx:3,@aa:8	7	F	0	abs		7	0	0	IMM	0							4
絶対アドレス	BSET	#xx:3,@aa:16	6	A	1	8		abs				7	0	0	IMM	0			5
絶対アドレス	BSET	#xx:3,@aa:32	6	A	3	8		abs								7	0	0	IMM	0	6
レジスタ直接	BSET	Rn,Rd	6	0	rn	rd													1
レジスタ間接	BSET	Rn,@ERd	7	D	0	erd	0	6	0	rn	0								4
絶対アドレス	BSET	Rn,@aa:8	7	F	0	abs		6	0	rn	0								4
絶対アドレス	BSET	Rn,@aa:16	6	A	1	8		abs				6	0	rn	0				5
絶対アドレス	BSET	Rn,@aa:32	6	A	3	8		abs								6	0	rn	0	6

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.23 BSR Branch to SubRoutine サブルーチン分岐

アセンブラフォーマット	オペレーション	オペランドサイズ
BSR disp	PC→@－SP PC＋disp→PC	－

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	－	－	－	－	－	－

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

指定されたアドレスにサブルーチン分岐します。PC の内容をリスタートアドレスとしてスタックに退避し、PC にディスプレースメントを加えたアドレスに分岐します。スタックに退避される PC の内容は本命令の直後の命令の先頭アドレスです。ディスプレースメントは符号付き 8 ビットまたは 16 ビットで、分岐できる範囲は本命令に対して－126～＋128、－32766～＋32768 バイトです。

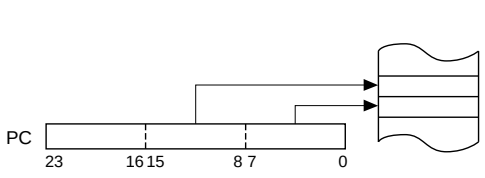
(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数	
			第1バイト		第2バイト		第3バイト	第4バイト	ノーマル	アドバンスト
プログラムカウンタ相対	BSR	d:8	5	5	disp				3	4
		d:16	5	C	0	0	disp		4	5

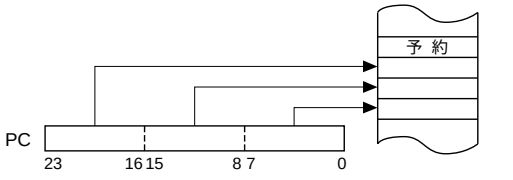
(4) 注意事項

ノーマルモードとアドバンストモードではスタックの構造が異なりますので、注意してください。ノーマルモードのとき退避される PC の内容は、下位 16 ビットのみです。

分岐先アドレスは、必ず偶数になるようにしてください。



ノーマルモード



アドバンストモード

2.2.24 BST Bit STore ビット転送

アセンブラフォーマット	オペレーション	オペランドサイズ
BST #xx：3, <EAd>	C→（<ビット番号>of<EAd>）	バイト

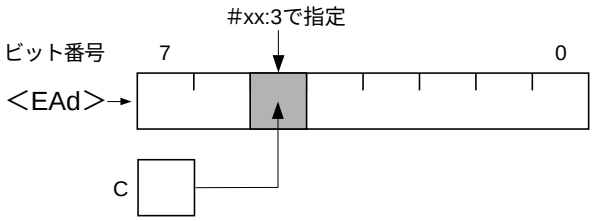
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	–

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットのロケーションに、キャリフラグの内容を転送します。ビット番号は、3 ビットのイミディエイトデータで指定されます。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット																実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		
レジスタ直接	BST	#xx:3,Rd	6	7	0	IMM	rd												1
レジスタ間接	BST	#xx:3,@ERd	7	D	0	erd	0	6	7	0	IMM	0							4
絶対アドレス	BST	#xx:3,@aa:8	7	F	abs		6	7	0	IMM	0								4
絶対アドレス	BST	#xx:3,@aa:16	6	A	1	8	abs				6	7	0	IMM	0				5
絶対アドレス	BST	#xx:3,@aa:32	6	A	3	8	abs								6	7	0	IMM	6

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.25 BTST Bit TeST ビットテスト

アセンブラフォーマット	オペレーション	オペランドサイズ
BTST #xx：3, <EAd> BTST Rn , <EAd>	～（<ビット番号>of<EAd>） →Z	バイト

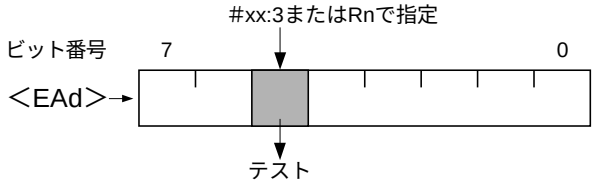
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	↕	–	–

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：指定したビットが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットの状態を調べて、その結果をゼロフラグに反映します。ビット番号は、3 ビットのイミディエイトデータまたは 8 ビットレジスタの内容の下位 3 ビットで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd：ER0～ER7
Rn ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット																実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		
レジスタ直接	BTST	#xx:3,Rd	7	3	0	IMM	rd												1
レジスタ間接	BTST	#xx:3,@ERd	7	C	0	erd	0	7	3	0	IMM	0							3
絶対アドレス	BTST	#xx:3,@aa:8	7	E	abs			7	3	0	IMM	0							3
絶対アドレス	BTST	#xx:3,@aa:16	6	A	1	0	abs				7	3	0	IMM					4
絶対アドレス	BTST	#xx:3,@aa:32	6	A	3	0	abs								7	3	0	IMM	5
レジスタ直接	BTST	Rn,Rd	6	3	rn	rd													1
レジスタ間接	BTST	Rn,@ERd	7	C	0	erd	0	6	3	rn	0								3
絶対アドレス	BTST	Rn,@aa:8	7	E	abs			6	3	rn	0								3
絶対アドレス	BTST	Rn,@aa:16	6	A	1	0	abs				6	3	rn	0					4
絶対アドレス	BTST	Rn,@aa:32	6	A	3	0	abs								6	3	rn	0	5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.26 BXOR Bit eXclusive OR ビット排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
BXOR #xx：3, <EAd>	C⊕（<ビット番号>of<EAd>）→C	バイト

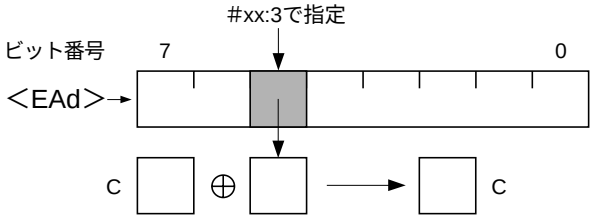
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	↕

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行結果が格納されます。

(2) 説明

デスティネーションオペランドの指定された 1 ビットとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。



(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERd ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット										実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト			
レジスタ直接	BXOR	#xx:3,Rd	7 5	0 IMM	rd								1
レジスタ間接	BXOR	#xx:3,@ERd	7 C	0 erd	0	7 5	0 IMM	0					3
絶対アドレス	BXOR	#xx:3,@aa:8	7 E	abs		7 5	0 IMM	0					3
絶対アドレス	BXOR	#xx:3,@aa:16	6 A	1	0	abs		7 5	0 IMM	0			4
絶対アドレス	BXOR	#xx:3,@aa:32	6 A	3	0	abs			7 5	0 IMM	0		5

【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。

(5) 注意事項

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.27 CLRMAC CLeaR MAC register MAC レジスタ初期化

アセンブラフォーマット	オペレーション	オペランドサイズ
CLRMAC	0→MACH, MACL	−

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

MACH、MACL レジスタを同時にクリアします。
H8S/2600 CPU のみでサポートしています。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
—	CLRMAC	—	0	1	A	0			2*

【注】* MAC命令実行後3ステート以内にCLRMAC命令を実行しようとした場合、最大で3ステート多くかかります。例えば、MAC命令とCLRMAC命令の間に1ステート命令（NOP等）が1つある場合、CLRMAC命令は2ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

(4) 注意事項

本命令を実行することで、乗算器内部のオーバフローフラグは 0 にクリアされます。

2.2.28 CMP (B) CoMPare 比較

アセンブラフォーマット	オペレーション	オペランドサイズ
CMP.B ＜EAs＞, Rd	Rd－（EAs）, CCR セット／クリア	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	↕	－	↕	↕	↕	↕

H ：ビット 3 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N ：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z ：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V ：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C ：ビット 7 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）からソースオペランドを減算し、その結果にしたがってコンディションコードをセットまたはクリアします。8 ビットレジスタ Rd の内容は変化しません。

(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
Rs ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	CMP.B	#xx:8,Rd	A	rd	IMM				1
レジスタ直接	CMP.B	Rs,Rd	1	C	rs	rd			1

2.2.29 CMP (W) CoMPare 比較

アセンブラフォーマット	オペレーション	オペランドサイズ
CMP.W ＜EAs＞, Rd	Rd－（EAs）, CCR セット／クリア	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	↕	－	↕	↕	↕	↕

H：ビット 11 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 15 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）からソースオペランドを減算し、その結果にしたがってコンディションコードをセットまたはクリアします。16 ビットレジスタ Rd の内容は変化しません。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7
Rs：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	CMP.W	#xx:16,Rd	7	9	2	rd	IMM		2
レジスタ直接	CMP.W	Rs,Rd	1	D	rs	rd			1

2.2.30 CMP (L) CoMPare 比較

アセンブラフォーマット	オペレーション	オペランドサイズ
CMP.L ＜EAs＞, ERd	ERd－（EAs）, CCR セット／クリア	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	↕	－	↕	↕	↕	↕

H：ビット 27 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 31 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

32 ビットレジスタ ERd の内容(デスティネーションオペランド)からソースオペランドを減算し、その結果にしたがって CCR の各ビットをセットまたはクリアします。32 ビットレジスタ ERd の内容は変化しません。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	第5バイト	第6バイト	
イミディエイト	CMP.L	#xx:32,ERd	7	A	2	0 erd	IMM				3
レジスタ直接	CMP.L	ERs,ERd	1	F	1 ers	0 erd					1

2.2.31 DAA Decimal Adjust Add 10 進補正

アセンブラフォーマット	オペレーション	オペランドサイズ
DAA Rd	Rd（10 進補正）→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	＊	−	↕	↕	＊	↕

H：値を保証しません。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：値を保証しません。
C：ビット 7 にキャリが発生したとき 1 にセットされ、それ以外のときは実行前の値が保持されます。

(2) 説明

ADD.B、ADDX.B 命令で、4 ビット BCD データを加算した結果が 8 ビットレジスタ Rd およびキャリフラグおよびハーフキャリフラグにあるとき、下表にしたがって 8 ビットレジスタ Rd の内容（デスティネーションオペランド）を補正（00、06、60、66 を加算）します。

補正前の C フラグ	補正前の 上位 4 ビット	補正前の H フラグ	補正前の 下位 4 ビット	加算される数 （16 進数）	補正後の C フラグ
0	0〜9	0	0〜9	00	0
0	0〜8	0	A〜F	06	0
0	0〜9	1	0〜3	06	0
0	A〜F	0	0〜9	60	1
0	9〜F	0	A〜F	66	1
0	A〜F	1	0〜3	66	1
1	0〜2	0	0〜9	60	1
1	0〜2	0	A〜F	66	1
1	0〜3	1	0〜3	66	1

(3) 使用可能な汎用レジスタ

Rd：R0L〜R7L、R0H〜R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	DAA	Rd	0	F	0	rd			1

(5) 注意事項

上記以外の場合について本命令を実行したときの結果（8 ビットレジスタ Rd の内容、および C、V、Z、N、H の各フラグ）は保証しません。

2.2.32 DAS Decimal Adjust Subtract 10 進補正

アセンブラフォーマット	オペレーション	オペランドサイズ
DAS Rd	Rd（10 進補正）→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	∗	−	↕	↕	∗	−

H：値を保証しません。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：値を保証しません。
C：実行前の値が保持されます。

(2) 説明

SUB.B、SUBX.B および NEG.B 命令で、4 ビット BCD データを減算した結果が 8 ビットレジスタ Rd、キャリフラグおよびハーフキャリフラグにあるとき、下表にしたがって 8 ビットレジスタ Rd（デスティネーションオペランド）の内容を補正（00、FA、A0、9A を加算）します。

補正前の C フラグ	補正前の 上位 4 ビット	補正前の H フラグ	補正前の 下位 4 ビット	加算される数 (16 進数)	補正後の C フラグ
0	0〜9	0	0〜9	00	0
0	0〜8	1	6〜F	FA	0
1	7〜F	0	0〜9	A0	1
1	6〜F	1	6〜F	9A	1

(3) 使用可能な汎用レジスタ

Rd：R0L〜R7L、R0H〜R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	DAS	Rd	1	F	0	rd			1

(5) 注意事項

上記以外の場合について本命令を実行したときの結果（8 ビットレジスタ Rd の内容、および C、V、Z、N、H の各フラグ）は保証しません。

2.2.33 DEC (B) DECrement デクリメント

アセンブラフォーマット	オペレーション	オペランドサイズ
DEC.B Rd	Rd－1→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	－	－	↕	↕	↕	－

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）から 1 を減算し、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	DEC.B	Rd	1	A	0	rd			1

(5) 注意事項

オーバフローは、H'80－1→H'7F のとき発生します。

2.2.34 DEC (W) DECrement デクリメント

アセンブラフォーマット	オペレーション	オペランドサイズ
DEC.W #1, Rd	Rd－1→Rd	ワード
DEC.W #2, Rd	Rd－2→Rd	

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	－	－	↕	↕	↕	－

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）から 1 または 2 を減算し、結果を 16 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	DEC.W	#1,Rd	1	B	5	rd			1
レジスタ直接	DEC.W	#2,Rd	1	B	D	rd			1

(5) 注意事項

オーバフローは、H'8000－1→H'7FFF, H'8000－2→H'7FFE, H'8001－2→H'7FFF のとき発生します。

2.2.35 DEC (L) DECrement デクリメント

アセンブラフォーマット	オペレーション	オペランドサイズ
DEC.L #1, ERd	ERd－1→ERd	ロングワード
DEC.L #2, ERd	ERd－2→ERd	

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	－	－	↕	↕	↕	－

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）から 1 または 2 を減算し、結果を 32 ビットレジスタ ERd に格納します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト		
レジスタ直接	DEC.L	#1,ERd	1	B	7	0	erd				1
レジスタ直接	DEC.L	#2,ERd	1	B	F	0	erd				1

(5) 注意事項

オーバフローは、H'80000000－1→H'7FFFFFFF, H'80000000－2→H'7FFFFFFE, H'80000001－2→H'7FFFFFFF のとき発生します。

2.2.36 DIVXS (B) DIVide eXtend as Signed 符号付き除算

アセンブラフォーマット	オペレーション	オペランドサイズ
DIVXS.B Rs, Rd	Rd÷Rs→Rd	バイト

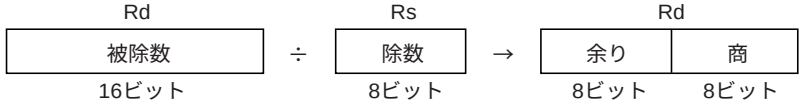
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	–	–

H：実行前の値が保持されます。
N：商が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：除数が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）を 8 ビットレジスタ Rs の内容（ソースオペランド）で符号付き除算し、結果を 16 ビットレジスタ Rd に格納します。演算は、16 ビット÷8 ビット→商 8 ビット、余り 8 ビットとして行われます。商は Rd の下位 8 ビットに、余りは上位 8 ビットに格納されます。余りの符号は、被除数の符号に一致しています。



なお、ゼロ除算またはオーバフローが発生した場合の結果は保証されません。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7
Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		
レジスタ直接	DIVXS.B	Rs,Rd	0	1	D	0	5	1	rs	rd	13

(5) 注意事項

N フラグは、被除数と除数の符号が異なるとき 1 にセットされ、符号が同じとき 0 にクリアされます。したがって、商が 0（ゼロ）で N フラグが 1 にセットされる場合があります。

2.2.37 DIVXS (W) DIVide eXtend as Signed 符号付き除算

アセンブラフォーマット	オペレーション	オペランドサイズ
DIVXS.W Rs, ERd	ERd÷Rs→ERd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	–	–

H：実行前の値が保持されます。
N：商が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：除数が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）を 16 ビットレジスタ Rs の内容（ソースオペランド）で符号付き除算し、結果を 32 ビットレジスタ ERd に格納します。演算は、32 ビット÷16 ビット→商 16 ビット、余り 16 ビットとして行われます。商は 32 ビットレジスタ ERd の下位 16 ビット（Rd）に、余りは上位 16 ビット（Ed）に格納します。余りの符号は、被除数の符号に一致しています。

なお、ゼロ除算またはオーバフローが発生した場合の結果は保証されません。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
Rs ：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット									実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト			
レジスタ直接	DIVXS.W	Rs,ERd	0	1	D	0	5	3	rs	0	erd	21

(5) 注意事項

N フラグは、被除数と除数の符号が異なるとき 1 にセットされ、符号が同じとき 0 にクリアされます。したがって、商が 0（ゼロ）で N フラグが 1 にセットされる場合があります。

2.2.38 DIVXU (B) DIVide eXtend as Unsigned 除算

アセンブラフォーマット	オペレーション	オペランドサイズ
DIVXU.B Rs, Rd	Rd÷Rs→Rd	バイト

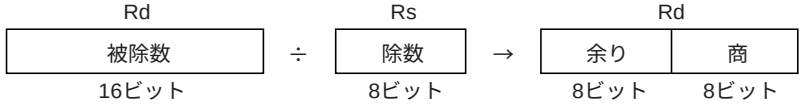
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	–	–

H：実行前の値が保持されます。
N：除数が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：除数が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）を 8 ビットレジスタ Rs の内容（ソースオペランド）で符号なし除算し、結果を 16 ビットレジスタ Rd に格納します。演算は、16 ビット÷8 ビット→商 8 ビット、余り 8 ビットとして行われます。商は Rd の下位 8 ビットに、余りは上位 8 ビットに格納します。



なお、ゼロ除算またはオーバフローが発生した場合の結果は保証されません。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7
Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	DIVXU.B	Rs,Rd	5	1	rs	rd			12

2.2.39 DIVXU (W) DIVide eXtend as Unsigned 除算

アセンブラフォーマット	オペレーション	オペランドサイズ
DIVXU.W Rs, ERd	ERd÷Rs→ERd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	–	–

H：実行前の値が保持されます。
N：除数が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：除数が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）を 16 ビットレジスタ Rs の内容（ソースオペランド）で符号なし除算し、結果を 32 ビットレジスタ ERd に格納します。演算は、32 ビット÷16 ビット→商 16 ビット、余り 16 ビットとして行われます。商は 32 ビットレジスタ ERd の下位 16 ビット（Rd）に、余りは上位 16 ビット（Ed）に格納します。

なお、ゼロ除算またはオーバフローが発生した場合の結果は保証されません。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
Rs ：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	DIVXU.W	Rs,ERd	5	3	rs	0 erd			20

2.2.40 EEPMOV (B) MOVe data to EEPROM ブロック転送

アセンブラフォーマット	オペレーション	オペランドサイズ
EEPMOV.B	if R4L≠0 then Repeat @ER5+→@ER6+ R4L−1→R4L Until R4L=0 else next;	−

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

ブロック転送命令です。ER5 で示されるメモリ上のデータを ER6 で示されるメモリへ転送し、ER5、ER6 の値をインクリメント、R4L の値をデクリメントします。R4L の内容が 0（ゼロ）となるまで上記動作を繰り返します。その後、次の命令を実行します。本命令でのデータ転送は、バイトサイズデータの連続転送となります。転送バイト数は R4L で示されます。アセンブラフォーマットのバイト表示は、8 ビットレジスタ R4L を示します（最大転送バイト数は 255 バイトとなります）。データ転送中は割り込みの検出を行いません。

本命令の実行終了時には、R4L は 0（ゼロ）を、また ER5、ER6 はそれぞれ（最終アドレス＋1）の内容を保持しています。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		
−	EEPMOV.B		7	B	5	C	5	9	8	F	4+2n*

【注】* R4Lの初期設定値がnの場合です。このとき転送データはnバイトですが、データアクセスは2（n＋1）回行われ、このデータアクセスに必要なステート数は2（n＋1）です。（n＝0、1、2、…255）

(4) 注意事項

本命令ではまず、ER5、ER6 で示されるメモリのリードを行い、その後、データのブロック転送を行います。

2.2.41 EEPMOV (W) MOVe data to EEPROM ブロック転送

アセンブラフォーマット	オペレーション	オペランドサイズ
EEPMOV. W	if R4≠0 then Repeat @ER5+→@ER6+ R4−1→R4 Until R4＝0 else next;	−

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

ブロック転送命令です。ER5 で示されるメモリ上のデータを ER6 で示されるメモリへ転送し、ER5、ER6 の値をインクリメント、R4 の値をデクリメントします。R4 の内容が 0（ゼロ）となるまで上記動作を繰り返します。その後、次の命令を実行します。本命令でのデータ転送は、バイトサイズデータの連続転送となります。転送バイト数は R4 で示されます。アセンブラフォーマットのワード表示は、16 ビットレジスタ R4 を示します（最大転送バイト数は 65535 バイトとなります）。データ転送中はすべての割り込みの検出を行います。

割り込みが発生しない状態での本命令の実行終了時には、R4 は 0（ゼロ）を、また ER5、ER6 はそれぞれ（最終アドレス＋1）の内容を保持しています。

割り込みが発生すると、転送中の 1 バイトの転送終了後、割り込み例外処理を行います。このとき R4 は残りの転送バイト数を、また ER5、ER6 はそれぞれ次の転送アドレスを示しています。割り込み例外処理で退避される PC は直後の命令の先頭アドレスです。

「（5） EEPMOV.W 命令と割り込み」を参照してください。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		
−	EEPMOV.W		7	B	D	4	5	9	8	F	4+2n*

【注】* R4の初期設定値がnの場合です。このとき転送データはnバイトですが、データアクセスは2（n＋1）回行われ、このデータアクセスに必要なステート数は2（n＋1）です。（n＝0、1、2、…65535）

(4) 注意事項

本命令ではまず、ER5、ER6 で示されるメモリのリードを行い、その後データのブロック転送を行います。

(5) EEPMOV.W 命令と割り込み

EEPMOV.W 命令実行中に割り込み要求が発生すると、転送中の 1 バイトの転送終了後、割り込み例外処理を実行します。このときのレジスタの内容は次のようになっています。

ER5 ：残りの転送元アドレスの先頭
ER6 ：残りの転送先アドレスの先頭
R4 ：残りの転送バイト数

また、この割り込み例外処理時にスタックされる PC の値は本命令の直後の命令の先頭アドレスになっています。したがって、EEPMOV.W 命令実行中に割り込み要求が発生する場合には以下のようなプログラムで対策を行ってください。

（例）

```
L1:  EEPMOV.W
     MOV.W      R4,R4
     BNE        L1
```

なお、NMI 以外の割り込み要求で、CPU 側で割り込みがマスクされている場合は受け付けられません。また、EEPMOV. B 命令では、NMI 割り込みを含めてすべての割り込みを受け付けません。

2.2.42 EXTS (W) EXTend as Signed 符号拡張

アセンブラフォーマット	オペレーション	オペランドサイズ
EXTS.W Rd	(<ビット 7>of Rd) →(<ビット 15～8>of Rd)	ワード

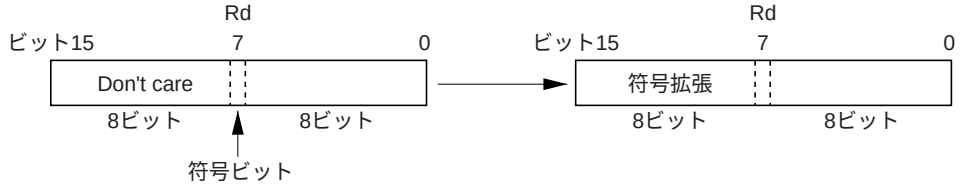
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	–

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の下位 8 ビットの符号を上位方向にコピーし、ワードサイズに符号拡張します（Rd のビット 7 をビット 15～8 にコピーします）。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	EXTS.W	Rd	1	7	D	rd			1

2.2.43 EXTS (L) EXTend as Signed 符号拡張

アセンブラフォーマット	オペレーション	オペランドサイズ
EXTS. L ERd	(<ビット 15>of ERd) →(<ビット 31～16>of ERd)	ロングワード

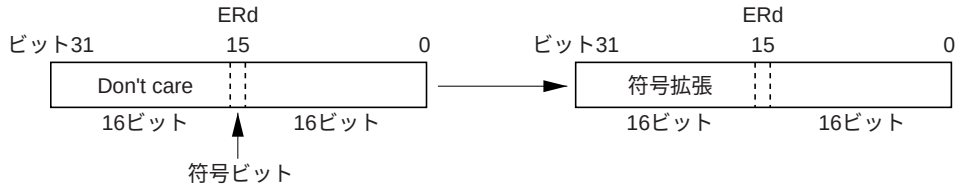
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	–

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の下位 16 ビットの符号ビットを上位方向にコピーし、ロングワードサイズに符号拡張します（ERd のビット 15 をビット 31～16 にコピーします）。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト		
レジスタ直接	EXTS.L	ERd	1	7	F	0	erd				1

2.2.44 EXTU(W) EXTend as Unsigned ゼロ拡張

アセンブラフォーマット	オペレーション	オペランドサイズ
EXTU.W Rd	0→（<ビット 15～8>of Rd）	ワード

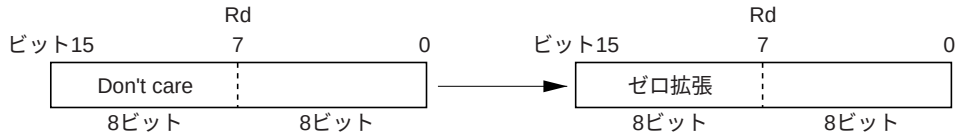
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	0	↕	0	−

H：実行前の値が保持されます。
N：常に 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の下位 8 ビットをワードサイズにゼロ拡張します。Rd の上位 8 ビット（ビット 15～8）に 0（ゼロ）が入ります。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	EXTU.W	Rd	1	7	5	rd			1

2.2.45 EXTU (L) EXTend as Unsigned ゼロ拡張

アセンブラフォーマット	オペレーション	オペランドサイズ
EXTU.L ERd	0→（<ビット 31～16>of ERd）	ロングワード

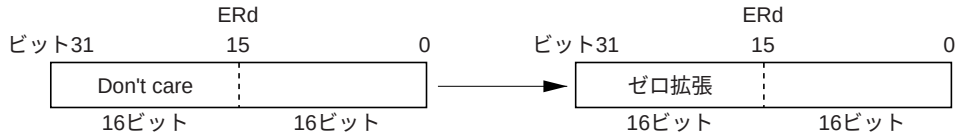
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	0	↕	0	−

H：実行前の値が保持されます。
N：常に 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の下位 16 ビット（汎用レジスタ Rd）をゼロ拡張してロングワードサイズにします。ERd の上位 16 ビット（ビット 31～16）に 0（ゼロ）が入ります。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト		
レジスタ直接	EXTU.L	ERd	1	7	7	0	erd				1

2.2.46 INC(B) INCrement インクリメント

アセンブラフォーマット	オペレーション	オペランドサイズ
INC.B Rd	Rd＋1→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	↕	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）に 1 を加算し、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	INC.B	Rd	0	A	0	rd			1

(5) 注意事項

オーバフローは H'7F＋1→H'80 のとき発生します。

2.2.47 INC (W) INCrement インクリメント

アセンブラフォーマット	オペレーション	オペランドサイズ
INC.W #1, Rd	Rd＋1→Rd	ワード
INC.W #2, Rd	Rd＋2→Rd	

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	↕	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）に 1 または 2 を加算し、結果を 16 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	INC.W	#1,Rd	0	B	5	rd			1
レジスタ直接	INC.W	#2,Rd	0	B	D	rd			1

(5) 注意事項

オーバフローは H'7FFF＋1→H'8000, H'7FFF＋2→H'8001, H'7FFE＋2→H'8000 のとき発生します。

2.2.48 INC (L) INCrement インクリメント

アセンブラフォーマット	オペレーション	オペランドサイズ
INC.L #1, ERd	ERd＋1→ERd	ロングワード
INC.L #2, ERd	ERd＋2→ERd	

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	↕	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）に 1 または 2 を加算し、結果を 32 ビットレジスタ ERd に格納します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	INC.L	#1,ERd	0	B	7	0	erd			1
レジスタ直接	INC.L	#2,ERd	0	B	F	0	erd			1

(5) 注意事項

オーバフローは H'7FFFFFFF＋1→H'80000000, H'7FFFFFFF＋2→H'80000001, H'7FFFFFFE＋2→H'80000000 のとき発生します。

2.2.49 JMP JuMP 無条件ジャンプ

アセンブラフォーマット	オペレーション	オペランドサイズ
JMP ＜EA＞	実効アドレス→PC	—

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

指定された実効アドレスに無条件に分岐します。

(3) 使用可能な汎用レジスタ

ERn：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	
			第1バイト		第2バイト			第3バイト	第4バイト		ノーマル	アドバンスト
レジスタ間接	JMP	@ERn	5	9	0	ern	0				2	
絶対アドレス	JMP	@aa:24	5	A	abs						3	
メモリ間接	JMP	@@aa:8	5	B	abs						4	5

(5) 注意事項

ノーマルモードとアドバンストモードでは、分岐アドレスの構造および実行ステート数が異なりますので注意してください。

分岐先アドレスは、必ず偶数になるようにしてください。

2.2.50 JSR Jump to SubRoutine サブルーチンジャンプ

アセンブラフォーマット	オペレーション	オペランドサイズ
JSR ＜EA＞	PC→@－SP 実効アドレス→PC	—

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

PC の内容をリスタートアドレスとしてスタックに退避し、指定された実効アドレスに分岐します。退避される PC 値は本命令の直後の命令の先頭アドレスになります。

(3) 使用可能な汎用レジスタ

ERn：ER0～ER7

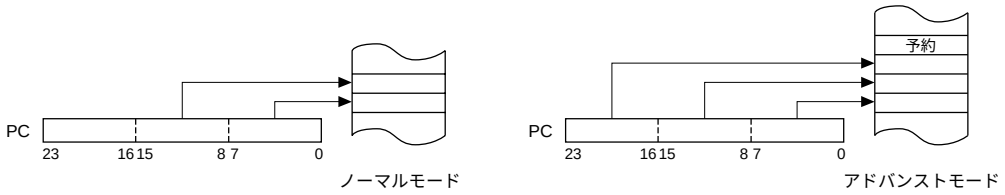
(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット							実行ステート数	
			第1バイト		第2バイト			第3バイト	第4バイト	ノーマル	アドバンスト
レジスタ間接	JSR	@ERn	5	D	0	ern	0			3	4
絶対アドレス	JSR	@aa:24	5	E	abs					4	5
メモリ間接	JSR	@@aa:8	5	F	abs					4	6

(5) 注意事項

ノーマルモードとアドバンストモードでは、スタックおよび分岐アドレスの構造が異なりますので注意してください。ノーマルモードのとき退避される PC の内容は、下位 16 ビットのみです。

分岐先アドレスは、必ず偶数になるようにしてください。



2.2.51 LDC(B) LoaD to Control register CCR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
LDC.B ＜EAs＞, CCR	（EAs）→CCR	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

I ：ソースオペランドの対応するビットの値が格納されます。
H：ソースオペランドの対応するビットの値が格納されます。
N：ソースオペランドの対応するビットの値が格納されます。
Z ：ソースオペランドの対応するビットの値が格納されます。
V ：ソースオペランドの対応するビットの値が格納されます。
C ：ソースオペランドの対応するビットの値が格納されます。

(2) 説明

ソースオペランドを CCR に転送します。
なお、本命令の実行終了時点では、NMI を含めてすべての割り込みは受け付けられません。

(3) 使用可能なレジスタ

Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	LDC.B	#xx:8,CCR	0	7	IMM				1
レジスタ直接	LDC.B	Rs,CCR	0	3	0	rs			1

2.2.52 LDC(B) LoaD to Control register EXR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
LDC.B ＜EAs＞, EXR	（EAs）→EXR	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

ソースオペランドを EXR に転送します。

なお、本命令の実行終了後 3 ステートの間は、NMI を含めてすべての割り込みは受け付けられません。

(3) 使用可能なレジスタ

Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		
イミディエイト	LDC.B	#xx:8,EXR	0	1	4	1	0	7	IMM		2
レジスタ直接	LDC.B	Rs,EXR	0	3	1	rs					1

2.2.53 LDC(W) LoaD to Control register CCR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
LDC.W ＜EAs＞, CCR	（EAs）→CCR	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

I ：ソースオペランドの対応するビットの値が格納されます。
H：ソースオペランドの対応するビットの値が格納されます。
N：ソースオペランドの対応するビットの値が格納されます。
Z ：ソースオペランドの対応するビットの値が格納されます。
V ：ソースオペランドの対応するビットの値が格納されます。
C ：ソースオペランドの対応するビットの値が格納されます。

(2) 説明

ソースオペランドを CCR に転送します。CCR はバイトサイズですが転送はワードサイズで行われ、偶数アドレスの内容が CCR に格納されます。

本命令の実行終了時点では、NMI を含めてすべての割り込みは受け付けられません。

(3) 使用可能な汎用レジスタ

ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット											実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト		
レジスタ間接	LDC.W	@ERs,CCR	0 1	4 0	6 9	0:ers 0								3
ディスプレースメント付レジスタ間接	LDC.W	@(d:16,ERs),CCR	0 1	4 0	6 F	0:ers 0	disp							4
	LDC.W	@(d:32,ERs),CCR	0 1	4 0	7 8	0:ers 0	6	B	2	0	disp			6
ポストインクリメントレジスタ間接	LDC.W	@ERs+,CCR	0 1	4 0	6 D	0:ers 0								4
絶対アドレス	LDC.W	@aa:16,CCR	0 1	4 0	6 B	0 0	abs							4
	LDC.W	@aa:32,CCR	0 1	4 0	6 B	2 0	abs							5

2.2.54 LDC(W) LoaD to Control register EXR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
LDC.W ＜EAs＞, EXR	（EAs）→EXR	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H ：実行前の値が保持されます。
N ：実行前の値が保持されます。
Z ：実行前の値が保持されます。
V ：実行前の値が保持されます。
C ：実行前の値が保持されます。

(2) 説明

ソースオペランドを EXR に転送します。EXR はバイトサイズですが転送はワードサイズで行われ、偶数アドレスの内容が EXR に格納されます。

なお、本命令の実行終了後 3 ステートの間は、NMI を含めてすべての割り込みは受け付けられません。

(3) 使用可能な汎用レジスタ

ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット																実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト	第8バイト	第9バイト	第10バイト	
レジスタ間接	LDC.W	@ERs,EXR	0	1	4	1	6	9	0	ers									3
ディスプレースメント付レジスタ間接	LDC.W	@(d:16,ERs),EXR	0	1	4	1	6	F	0	ers	disp								4
	LDC.W	@(d:32,ERs),EXR	0	1	4	1	7	8	0	ers	6	B	2	0	disp				6
ポストインクリメントレジスタ間接	LDC.W	@ERs+,EXR	0	1	4	1	6	D	0	ers									4
絶対アドレス	LDC.W	@aa:16,EXR	0	1	4	1	6	B	0	0	abs								4
	LDC.W	@aa:32,EXR	0	1	4	1	6	B	2	0	abs								5

2.2.55 LDM LoaD to Multiple register スタックよりデータ復帰

アセンブラフォーマット	オペレーション	オペランドサイズ
LDM.L @SP+, ＜レジスタリスト＞	@SP+ →ERn（レジスタ群）	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H ：実行前の値が保持されます。
N ：実行前の値が保持されます。
Z ：実行前の値が保持されます。
V ：実行前の値が保持されます。
C ：実行前の値が保持されます。

(2) 説明

退避されたデータを、スタックからレジスタリストで指定した複数のレジスタに復帰します。復帰はレジスタ番号の大きい順に行われます。

1 命令で復帰できるレジスタ数は 2 本、3 本、4 本です。そのとき指定可能なレジスタリストは以下の通りです。

2 本：ER0 - ER1、ER2 - ER3、ER4 - ER5、ER6 - ER7
3 本：ER0 - ER2、ER4 - ER6
4 本：ER0 - ER3、ER4 - ER7

(3) 使用可能な汎用レジスタ

ERn：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		
−	LDM.L	@SP+,(ERn-ERn+1)	0	1	1	0	6	D	7	0 ern+1	7
−	LDM.L	@SP+,(ERn-ERn+2)	0	1	2	0	6	D	7	0 ern+2	9
−	LDM.L	@SP+,(ERn-ERn+3)	0	1	3	0	6	D	7	0 ern+3	11

2.2.56 LDMAC LoaD to MAC register MAC レジスタ転送

アセンブラフォーマット	オペレーション	オペランドサイズ
LDMAC ERs, MAC レジスタ	ERs →MACH または ERs →MACL	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	–

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

汎用レジスタの内容を MAC レジスタ（MACH、MACL）に転送します。
MACH への転送では、汎用レジスタの下位 10 ビットのみ転送されます。
H8S/2600 CPU のみでサポートしています。

(3) 使用可能な汎用レジスタ

ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト		
レジスタ直接	LDMAC	ERs, MACH	0	3	2	0	ers				2*
レジスタ直接	LDMAC	ERs, MACL	0	3	3	0	ers				2*

【注】* MAC命令実行後3ステート以内にLDMAC命令を実行しようとした場合、最大で3ステート多くかかります。
例えば、MAC命令とLDMAC命令の間に1ステート命令（NOP等）が1つある場合、LDMAC命令は2ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

(5) 注意事項

本命令を実行することで、乗算器内部のオーバフローフラグは 0 にクリアされます。

2.2.57 MAC Multiply and ACcumulate 積和演算

アセンブラフォーマット	オペレーション	オペランドサイズ
MAC @ERn+, @ERm+	（EAn）×（EAm）＋MAC レジスタ →MAC レジスタ ERn ＋2→ERn ERm＋2→ERm	－

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	－	－	－*	－*	－*	－

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

汎用レジスタ ERn と ERm の内容をアドレスとする 16 ビットオペランドを符号付きで乗算し、結果の 32 ビットと MAC レジスタの内容とを加算し、結果を MAC レジスタに格納します。

演算後 ERn と ERm の内容をともに＋2 します。

また、システムコントロールレジスタ（SYSCR）の MACS ビットにより、飽和／非飽和演算を切り替えることができます。詳細は、当該製品のハードウェアマニュアルを参照してください。

- 非飽和演算のとき
  連結したMACH、MACLレジスタに結果の42ビットを格納します。MACHレジスタの上位22ビットにはビット41の内容を転送（符号拡張）します。

- 飽和演算のとき
  飽和演算のときは、MACLレジスタのみ有効となり、結果の範囲をH'80000000（最小値）からH'7FFFFFFF（最大値）までに制限します。結果が負の方向にオーバフローしたときはH'80000000（最小値）を、正の方向にオーバフローしたときはH'7FFFFFFF（最大値）をMACLレジスタに格納します。MACHレジスタの最下位ビットは、乗算器内部のオーバフローフラグ（V-MULT）を示します。その他のビットは保持されます。

H8S/2600 CPU のみでサポートしています。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット										実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト				
ポストインクリメントレジスタ間接	MAC	@ERn+,@ERm+	0	1	6	0	6	D	0	ern	0	erm	4

(4) 注意事項

1. MAC命令の演算結果を示すフラグ（N、Z、V）はSTMAC命令によりCCRに反映できます。
2. ERnとERmが同じレジスタの場合、実行アドレスはERnとERn＋2になります。また演算後のERnの値はERn＋4になります。
3. MAC命令実行中にMACSビットを書き替えた場合、演算結果は保証されません。必ずMAC命令の後3ステート以上経ってから書き替えてください。

(5) MAC 命令のフラグ

(a) フラグ変化について

乗算器内部には CCR とは別に MAC 命令の結果を反映する N-MULT、Z-MULT、V-MULT フラグを持っています。

これらのフラグは STMAC 命令の実行でのみ、CCR の N、Z、V フラグに反映されます。

N-MULT、Z-MULT は MAC 命令でのみ更新されます。V-MULT は CLRMAC または LDMAC 命令実行まで、過去にオーバフローしたかどうかを保持します。

以下にフラグのセット／クリア条件を示します。

N-MULT（ネガティブフラグ）

飽和演算	（セット条件）	MAC 命令の結果、MACL レジスタのビット 31 が 1 のとき。
	（クリア条件）	MAC 命令の結果、MACL レジスタのビット 31 が 0 のとき。
非飽和演算	（セット条件）	MAC 命令の結果、MACH レジスタのビット 41 が 1 のとき。
	（クリア条件）	MAC 命令の結果、MACH レジスタのビット 41 が 0 のとき。

Z-MULT（ゼロフラグ）

飽和演算	（セット条件）	MAC 命令の結果、MACL レジスタがゼロのとき。
	（クリア条件）	MAC 命令の結果、MACL レジスタがゼロでないとき。
非飽和演算	（セット条件）	MAC 命令の結果、MACH、MACL レジスタがともにゼロのとき。
	（クリア条件）	MAC 命令の結果、MACH、MACL レジスタの少なくともどちらかがゼロでないとき。

V-MULT（オーバフローフラグ）

飽和演算	（セット条件）	MAC 命令の結果が範囲 H'80000000（最小値）～H'7FFFFFFF（最大値）を越えたとき。
	（クリア条件）	CLRMAC または LDMAC 命令を実行したとき*。
非飽和演算	（セット条件）	MAC 命令の結果が範囲 H'20000000000（最小値）～ H'1FFFFFFFFFF（最大値）を越えたとき。
	（クリア条件）	CLRMAC または LDMAC 命令を実行したとき*。

【注】 * MAC 命令の結果が範囲内に収まったとしても、クリアはされません。

なお、飽和／非飽和演算の切り替え、および乗算命令（MULXU, MULXS）実行による N-MULT、Z-MULT、V-MULT フラグの変化はありません。

(b) 動作例

```
CLRMAC
MAC    @ER1+, @ER2+
MAC    @ER1+, @ER2+            ←  オーバフロー発生
:
MAC    @ER1+, @ER2+            ←  演算結果＝0
NOP
STMAC  MACH, ER3               ←  CCR（N=0, Z=1, V=1）
CLRMAC
STMAC  MACH, ER3               ←  CCR（N=0, Z=1, V=0）
```

2.2.58 MOV(B) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.B Rs, Rd	Rs→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

8 ビットレジスタ Rs の内容を 8 ビットレジスタ Rd へ転送します。このとき転送するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H
Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	MOV.B	Rs,Rd	0	C	rs	rd			1

2.2.59 MOV(W) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.W Rs, Rd	Rs→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rs の内容を 16 ビットレジスタ Rd へ転送します。このとき転送するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

Rd ：R0～R7、E0～E7
Rs ：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	MOV.W	Rs,Rd	0	D	rs	rd			1

2.2.60 MOV (L) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.L ERs, ERd	ERs→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERs の内容を 32 ビットレジスタ ERd へ転送します。このとき転送するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト				第3バイト	第4バイト	
レジスタ直接	MOV.L	ERs,ERd	0	F	1	ers	0	erd			1

2.2.61 MOV (B) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.B ＜EAs＞, Rd	（EAs）→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

ソースオペランドの内容を 8 ビットレジスタ Rd に転送します。このとき、転送するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
ERs ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット									実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト		
イミディエイト	MOV.B	#xx:8,Rd	F rd	IMM								1
レジスタ間接	MOV.B	@ERs,Rd	6 8	0 ers rd								2
ディスプレースメント付レジスタ間接	MOV.B	@(d:16,ERs),Rd	6 E	0 ers rd	disp							3
	MOV.B	@(d:32,ERs),Rd	7 8	0 ers 0	6 A	2	rd					5
ポストインクリメントレジスタ間接	MOV.B	@ERs+,Rd	6 C	0 ers rd								3
絶対アドレス	MOV.B	@aa:8,Rd	2 rd	abs								2
	MOV.B	@aa:16,Rd	6 A	0 rd	abs							3
	MOV.B	@aa:32,Rd	6 A	2 rd	abs							4

(5) 注意事項

「MOV.B @ER7+,Rd」は、SP（ER7）の内容が奇数値となるため使用しないでください。詳細は「3.3 例外処理状態」または当該製品のハードウェアマニュアルを参照してください。

@aa：8／@aa：16 のアクセス範囲については、当該製品のハードウェアマニュアルを参照してください。

2.2.62 MOV (W) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.W ＜EAs＞, Rd	（EAs）→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

ソースオペランドの内容を 16 ビットレジスタ Rd へ転送します。このとき転送するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

Rd ：R0～R7、E0～E7
ERs ：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	
イミディエイト	MOV.W	#xx:16,Rd	7 9	0 rd	IMM						2
レジスタ間接	MOV.W	@ERs,Rd	6 9	0:ers rd							2
ディスプレースメント付レジスタ間接	MOV.W	@(d:16,ERs),Rd	6 F	0:ers rd	disp						3
	MOV.W	@(d:32,ERs),Rd	7 8	0:ers 0	6 B	2 rd	disp				5
ポストインクリメントレジスタ間接	MOV.W	@ERs+,Rd	6 D	0:ers rd							3
絶対アドレス	MOV.W	@aa:16,Rd	6 B	0 rd	abs						3
	MOV.W	@aa:32,Rd	6 B	2 rd	abs						4

(5) 注意事項

1. アドレス＜EAs＞は必ず偶数になるようにしてください。
2. 「MOV.W @ER7＋,Rd」の機械語はPOP.W Rdと同一です。

2.2.63 MOV (L) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.L ＜EAs＞, ERd	（EAs）→ERd	ロングワード

(1) コンディションコード

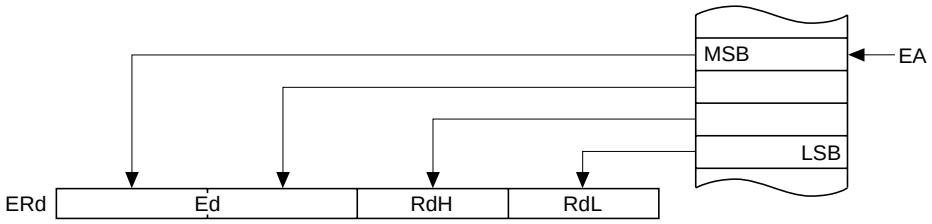
I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

ソースオペランドの内容を指定された 32 ビットレジスタ ERd へ転送します。このとき転送するデータを検査し、その結果を CCR に反映します。

実効アドレスが示す先頭の 1 ワードのメモリの内容が拡張レジスタ Ed に格納され、次の 1 ワードのメモリの内容が汎用レジスタ Rd に格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット										実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト	
イミディエイト	MOV.L	#xx:32,Rd	7 A	0 0 erd	IMM								3
レジスタ間接	MOV.L	@ERs,ERd	0 1	0 0	6 9	0 ers 0 erd							4
ディスプレースメント付レジスタ間接	MOV.L	@(d:16,ERs),ERd	0 1	0 0	6 F	0 ers 0 erd	disp						5
	MOV.L	@(d:32,ERs),ERd	0 1	0 0	7 8	0 ers 0	6	B	2	0 erd	disp		7
ポストインクリメントレジスタ間接	MOV.L	@ERs+,ERd	0 1	0 0	6 D	0 ers 0 erd							5
絶対アドレス	MOV.L	@aa:16,ERd	0 1	0 0	6 B	0 0 erd	abs						5
	MOV.L	@aa:32,ERd	0 1	0 0	6 B	2 0 erd	abs						6

(5) 注意事項

1. アドレス＜EAs＞は必ず偶数になるようにしてください。
2. 「MOV.L @ER7＋,ERd」の機械語はPOP.L ERdと同一です。

2.2.64 MOV (B) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.B Rs, ＜EAd＞	Rs→（EAd）	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

8 ビットレジスタ Rs の内容（ソースオペランド）をデスティネーションのロケーションに転送します。このとき、転送するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

Rs ：R0L～R7L、R0H～R7H
ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット												実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト	第6バイト	第7バイト	第8バイト	
レジスタ間接	MOV.B	Rs,@ERd	6	8	1 erd	rs									2
ディスプレースメント付レジスタ間接	MOV.B	Rs,@(d:16,ERd)	6	E	1 erd	rs	disp								3
	MOV.B	Rs,@(d:32,ERd)	7	8	0 erd	0	6	A	rs		disp				5
プリデクリメントレジスタ間接	MOV.B	Rs,@-ERd	6	C	1 erd	rs									3
絶対アドレス	MOV.B	Rs,@aa:8	3	rs	abs										2
	MOV.B	Rs,@aa:16	6	A	8	rs	abs								3
	MOV.B	Rs,@aa:32	6	A	A	rs	abs								4

(5) 注意事項

1. 「MOV.B Rs,@－ER7」は、SP（ER7）の内容が奇数値となるため使用しないでください。詳細は「3.3 例外処理状態」または当該製品のハードウェアマニュアルを参照してください。
2. MOV.B RnL,@－ERnまたはMOV.B RnH,@－ERnを実行すると（実行前のERnの内容－1）の下位RnLまたは上位RnHが転送されます。

2.2.65 MOV (W) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.W Rs, ＜EAd＞	Rs→（EAd）	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rs の内容（ソースオペランド）をデスティネーションのロケーションに転送します。このとき転送するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

Rs ：R0～R7、E0～E7
ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット									実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト		
レジスタ間接	MOV.W	Rs,@ERd	6 9	1 erd rs								2
ディスプレースメント付レジスタ間接	MOV.W	Rs,@(d:16,ERd)	6 F	1 erd rs	disp							3
	MOV.W	Rs,@(d:32,ERd)	7 8	0 erd 0	6	B A rs	disp					5
プリデクリメントレジスタ間接	MOV.W	Rs,@-ERd	6 D	1 erd rs								3
絶対アドレス	MOV.W	Rs,@aa:16	6 B	8 rs	abs							3
	MOV.W	Rs,@aa:32	6 B	A rs	abs							4

(5) 注意事項

1. アドレス＜EAd＞は必ず偶数になるようにしてください。
2. 「MOV.W Rs,@－ER7」の機械語はPUSH.W Rsと同一です。
3. MOV.W Rn,@－ERnを実行すると（実行前のERnの内容－2）が転送されます。

2.2.66 MOV (L) MOVe data 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOV.L ERs, ＜EAd＞	ERs→（EAd）	ロングワード

(1) コンディションコード

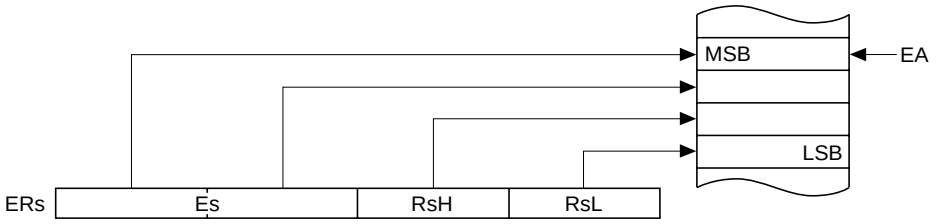
I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERs の内容（ソースオペランド）をデスティネーションのロケーションに転送します。このとき、転送するデータを検査し、その結果を CCR に反映します。

実効アドレスが示す先頭の 1 ワードに拡張レジスタの内容が、次の 1 ワードに汎用レジスタ Rd の内容が格納されます。



(3) 使用可能な汎用レジスタ

ERs：ER0～ER7
ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット 第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト	実行ステート数
レジスタ間接	MOV.L	ERs,@ERd	0 1	0 0	6 9	1 erd 0 ers							4
ディスプレースメント付レジスタ間接	MOV.L	ERs,@(d:16,ERd)	0 1	0 0	6 F	1 erd 0 ers	disp						5
	MOV.L	ERs,@(d:32,ERd)	0 1	0 0	7 8	0 erd 0	6	B	A	0 ers	disp		7
プリデクリメントレジスタ間接	MOV.L	ERs,@-ERd	0 1	0 0	6 D	1 erd 0 ers							5
絶対アドレス	MOV.L	ERs,@aa:16	0 1	0 0	6 B	8 0 ers	abs						5
	MOV.L	ERs,@aa:32	0 1	0 0	6 B	A 0 ers	abs						6

(5) 注意事項

1. アドレス＜EAd＞は必ず偶数になるようにしてください。
2. 「MOV.L ERs,@－ER7」の機械語はPUSH.L ERsと同一です。
3. MOV.L ERn,@－ERnを実行すると（実行前のERnの内容－4）が転送されます。

2.2.67 MOVFPE MOVe From Peripheral with E clock E 同期データ転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOVFPE @aa：16, Rd	（EAs）→ Rd E 同期	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビット絶対アドレスで指定されるメモリの内容を、E クロックに同期したタイミングで汎用レジスタ Rd に転送します。このとき転送するデータを検査し、結果を CCR に反映します。

【注】 E クロック出力端子を備えていない製品およびシングルチップモードでは、本命令を使用しないでください。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		
絶対アドレス	MOVFPE	@aa:16,Rd	6	A	4	rd	abs				*

【注】* 詳細は、当該製品のハードウェアマニュアルを参照してください。

(5) 注意事項

1. 本命令では、上記以外のアドレッシングモードおよびワードサイズ／ロングワードサイズのデータは扱えません。
2. 本命令のデータ転送についての詳細は、当該製品のハードウェアマニュアルを参照してください。

2.2.68 MOVTPE MOVe To Peripheral with E clock E 同期データ転送

アセンブラフォーマット	オペレーション	オペランドサイズ
MOVTPE Rs,@aa：16	Rs → (ERd) E 同期	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

汎用レジスタ Rs の内容（ソースオペランド）を、E クロックに同期したタイミングで、16 ビット絶対アドレスで指定されるデスティネーションオペランドのロケーションに転送します。このとき転送するデータを検査し、結果を CCR に反映します。

【注】 E クロック出力端子を備えていない製品およびシングルチップモードでは、本命令を使用しないでください。

(3) 使用可能な汎用レジスタ

Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
絶対アドレス	MOVTPE	Rs,@aa:16	6	A	C	rs	abs		*

【注】* 詳細は、当該製品のハードウェアマニュアルを参照してください。

(5) 注意事項

1. 本命令では、上記以外のアドレッシングモードおよびワードサイズ／ロングワードサイズのデータは扱えません。
2. 本命令のデータ転送についての詳細は、当該製品のハードウェアマニュアルを参照してください。

2.2.69 MULXS(B) MULtiply eXtend as Signed 符号付き乗算

アセンブラフォーマット	オペレーション	オペランドサイズ
MULXS.B Rs, Rd	Rd×Rs→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	–	–

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容の下位 8 ビット（デスティネーションオペランド）と 8 ビットレジスタ Rs の内容（ソースオペランド）を符号付き乗算し、結果を 16 ビットレジスタ Rd に格納します。Rd を汎用レジスタ R としたとき、Rs は RdH または RdL を指定することも可能です。

演算は、8 ビット×8 ビット→16 ビットで行われます。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7
Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		
レジスタ直接	MULXS.B	Rs,Rd	0	1	C	0	5	0	rs	rd	4*

【注】* H8S/2000 CPUでの実行ステート数は13です。

MAC命令実行後2ステート以内にMULXS.B命令を実行しようとした場合、最大で2ステート多くかかります。例えば、MAC命令とMULXS.B命令の間に1ステート命令（NOP等）が1つある場合、MULXS.B命令は1ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

2.2.70 MULXS (W) MULtiply eXtend as Signed 符号付き乗算

アセンブラフォーマット	オペレーション	オペランドサイズ
MULXS.W Rs, ERd	ERd×Rs→ERd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	–	–

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容の下位 16 ビット（デスティネーションオペランド）と 16 ビットレジスタ Rs の内容（ソースオペランド）を符号付き乗算し、結果を 32 ビットレジスタ ERd に格納します。

Rs は Ed または Rd を指定することも可能です。

演算は、16 ビット×16 ビット→32 ビットで行われます。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
Rs ：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット									実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト			
レジスタ直接	MULXS.W	Rs,ERd	0	1	C	0	5	2	rs	0	erd	5*

【注】* H8S/2000 CPUでの実行ステート数は21です。

MAC命令実行後2ステート以内にMULXS.W命令を実行しようとした場合、最大で2ステート多くかかります。例えば、MAC命令とMULXS.W命令の間に1ステート命令（NOP等）が1つある場合、MULXS.W命令は1ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

2.2.71 MULXU (B) MULtiply eXtend as Unsigned 乗算

アセンブラフォーマット	オペレーション	オペランドサイズ
MULXU.B Rs, Rd	Rd×Rs→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	–

H ：実行前の値が保持されます。
N ：実行前の値が保持されます。
Z ：実行前の値が保持されます。
V ：実行前の値が保持されます。
C ：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容の下位 8 ビット（デスティネーションオペランド）と 8 ビットレジスタ Rs の内容（ソースオペランド）を符号なし乗算し、結果を 16 ビットレジスタ Rd に格納します。Rd を汎用レジスタ R としたとき、Rs は RdH または RdL を指定することも可能です。

演算は、8 ビット×8 ビット→16 ビットで行われます。

(3) 使用可能な汎用レジスタ

Rd ：R0～R7、E0～E7
Rs ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	MULXU.B	Rs,Rd	5	0	rs	rd			3*

【注】* H8S/2000 CPUでの実行ステート数は12です。

MULXU、MULXS、STMAC命令の直後は1ステート多くかかります。
また、MAC命令実行後3ステート以内にMULXU.B命令を実行しようとした場合、最大で3ステート多くかかります。例えば、MAC命令とMULXU.B命令の間に1ステート命令（NOP等）が1つある場合、MULXU.B命令は2ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

2.2.72 MULXU (W) MULtiply eXtend as Unsigned 乗算

アセンブラフォーマット	オペレーション	オペランドサイズ
MULXU.W Rs, ERd	ERd×Rs→ERd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	–	–	–	–

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容の下位 16 ビット（デスティネーションオペランド）と 16 ビットレジスタ Rs の内容（ソースオペランド）を符号なし乗算し、結果を 32 ビットレジスタ ERd に格納します。Rs は Ed または Rd を指定することも可能です。

演算は、16 ビット×16 ビット→32 ビットで行われます。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
Rs ：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト		
レジスタ直接	MULXU.W	Rs,ERd	5	2	rs	0	erd				4*

【注】* H8S/2000 CPUでの実行ステート数は20です。

MULXU、MULXS、STMAC命令の直後は1ステート多くかかります。
また、MAC命令実行後3ステート以内にMULXU.W命令を実行しようとした場合、最大で3ステート多くかかります。例えば、MAC命令とMULXU.W命令の間に1ステート命令（NOP等）が1つある場合、MULXU.W命令は2ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

2.2.73 NEG (B) NEGate 2 進符号反転

アセンブラフォーマット	オペレーション	オペランドサイズ
NEG.B Rd	0−Rd→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	↕	−	↕	↕	↕	↕

H：ビット 3 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 7 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）の 2 の補数をとり（H' 00 から減算し）、結果を 8 ビットレジスタ Rd に格納します。ただし、実行前の Rd の内容が H' 80 の場合の結果は H' 80 となります。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	NEG.B	Rd	1	7	8	rd			1

(5) 注意事項

オーバフローは、実行前の Rd の内容が H'80 のとき発生します。

2.2.74 NEG (W) NEGate 2 進符号反転

アセンブラフォーマット	オペレーション	オペランドサイズ
NEG.W Rd	0−Rd→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	↕	−	↕	↕	↕	↕

H：ビット 11 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 15 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）の 2 の補数をとり（H'0000 から減算し）、結果を 16 ビットレジスタ Rd に格納します。ただし、実行前の Rd の内容が H'8000 の場合の結果は H'8000 となります。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	NEG.W	Rd	1	7	9	rd			1

(5) 注意事項

オーバフローは、実行前の Rd の内容が H'8000 のとき発生します。

2.2.75 NEG (L) NEGate 2 進符号反転

アセンブラフォーマット	オペレーション	オペランドサイズ
NEG.L ERd	0−ERd→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	↕	−	↕	↕	↕	↕

H：ビット 27 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 31 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）の 2 の補数をとり（H'00000000 から減算し）、結果を 32 ビットレジスタ ERd に格納します。ただし、実行前の ERd の内容が H'80000000 の場合の結果は H'80000000 となります。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト		
レジスタ直接	NEG.L	ERd	1	7	B	0	erd				1

(5) 注意事項

オーバフローは、実行前の ERd の内容が H'80000000 のとき発生します。

2.2.76 NOP No OPeration 無操作

アセンブラフォーマット	オペレーション	オペランドサイズ
NOP	PC＋2→PC	—

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H ：実行前の値が保持されます。
N ：実行前の値が保持されます。
Z ：実行前の値が保持されます。
V ：実行前の値が保持されます。
C ：実行前の値が保持されます。

(2) 説明

PC のインクリメントのみを行い、次の命令に実行が移ります。CPU の内部状態には影響を与えません。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
—	NOP		0	0	0	0			1

2.2.77 NOT (B) NOT=logical complement 論理反転

アセンブラフォーマット	オペレーション	オペランドサイズ
NOT.B Rd	～Rd→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	–

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）の 1 の補数をとり、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	NOT.B	Rd	1	7	0	rd			1

2.2.78 NOT (W) NOT=logical complement 論理反転

アセンブラフォーマット	オペレーション	オペランドサイズ
NOT.W Rd	～Rd→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	–

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）の 1 の補数をとり、結果を 16 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	NOT.W	Rd	1	7	1	rd			1

2.2.79 NOT (L) NOT=logical complement 論理反転

アセンブラフォーマット	オペレーション	オペランドサイズ
NOT.L ERd	～ERd→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	–

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）の 1 の補数をとり、結果を 32 ビットレジスタ ERd に格納します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット							実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	NOT.L	ERd	1	7	3	0	erd			1

2.2.80 OR (B) inclusive OR logical 論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
OR.B ＜EAs＞, Rd	Rd∨（EAs）→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）と、ソースオペランドの論理和をとり、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H
Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	OR.B	#xx:8,Rd	C	rd	IMM				1
レジスタ直接	OR.B	Rs,Rd	1	4	rs	rd			1

2.2.81 OR (W) inclusive OR logical 論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
OR.W ＜EAs＞, Rd	Rd∨（EAs）→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）と、ソースオペランドの論理和をとり、結果を 16 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7
Rs：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	OR.W	#xx:16,Rd	7	9	4	rd	IMM		2
レジスタ直接	OR.W	Rs,Rd	6	4	rs	rd			1

2.2.82 OR (L) inclusive OR logical 論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
OR.L ＜EAs＞, ERd	ERd∨（EAs）→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）と、ソースオペランドの論理和をとり、結果を 32 ビットレジスタ ERd に格納します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット										実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト	第6バイト	
イミディエイト	OR.L	#xx:32,ERd	7	A	4	0 erd	IMM						3
レジスタ直接	OR.L	ERs,ERd	0	1	F	0	6	4	0 ers	0 erd			2

2.2.83 ORC inclusive OR Control register CCR との論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
ORC #xx：8 CCR	CCR∨#IMM→CCR	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

I ：実行結果の対応するビットの値が格納されます。
UI：実行結果の対応するビットの値が格納されます。
H ：実行結果の対応するビットの値が格納されます。
U ：実行結果の対応するビットの値が格納されます。
N ：実行結果の対応するビットの値が格納されます。
Z ：実行結果の対応するビットの値が格納されます。
V ：実行結果の対応するビットの値が格納されます。
C ：実行結果の対応するビットの値が格納されます。

(2) 説明

CCR の内容とイミディエイトデータの論理和をとり、結果を CCR に格納します。
本命令の実行終了時点では、NMI を含めてすべての割り込みは受け付けられません。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット					実行ステート数
			第1バイト		第2バイト	第3バイト	第4バイト	
イミディエイト	ORC	#xx:8,CCR	0	4	IMM			1

2.2.84 ORC inclusive OR Control register EXR との論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
ORC #xx：8, EXR	EXR∨#IMM→EXR	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H ：実行前の値が保持されます。
N ：実行前の値が保持されます。
Z ：実行前の値が保持されます。
V ：実行前の値が保持されます。
C ：実行前の値が保持されます。

(2) 説明

EXR の内容とイミディエイトデータの論理和をとり、結果を EXR に格納します。

なお、本命令の実行終了後 3 ステートの間は、NMI を含めてすべての割り込みは受け付けられません。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット							実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト	
イミディエイト	ORC	#xx:8,EXR	0	1	4	1	0	4	IMM	2

2.2.85 POP (W) POP data スタックよりデータ復帰

アセンブラフォーマット	オペレーション	オペランドサイズ
POP.W Rn	@SP＋→Rn	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

スタックから 16 ビットレジスタ Rn へデータを復帰します。このとき復帰するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

Rn：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
−	POP.W	Rn	6	D	7	rn			3

(5) 注意事項

本命令は、MOV.W @SP＋, Rn と同一です。

2.2.86 POP (L) POP data スタックよりデータ復帰

アセンブラフォーマット	オペレーション	オペランドサイズ
POP.L ERn	@SP＋→ERn	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

スタックから 32 ビットレジスタ ERn へデータを復帰します。このとき復帰するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

ERn：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット									実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト			
−	POP.L	ERn	0	1	0	0	6	D	7	0	ern	5

(5) 注意事項

本命令は、MOV.L @SP＋,ERn と同一です。

2.2.87 PUSH (W) PUSH data スタックへデータ退避

アセンブラフォーマット	オペレーション	オペランドサイズ
PUSH.W Rn	Rn→@－SP	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	－	－	↕	↕	0	－

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rn の内容をスタックに退避します。このとき退避するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

Rn：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
－	PUSH.W	Rn	6	D	F	rn			3

(5) 注意事項

1. 本命令は、MOV.W Rn,@－SPと同一です。
2. PUSH.W R7または、PUSH.W E7を実行すると実効アドレス計算（ER7－2→ER7実行）後のR7またはE7がスタックに退避されます。

2.2.88 PUSH (L) PUSH data スタックへデータ退避

アセンブラフォーマット	オペレーション	オペランドサイズ
PUSH.L ERn	ERn→@－SP	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	－	－	↕	↕	0	－

H：実行前の値が保持されます。
N：転送データが負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：転送データが 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERn の内容をスタックに退避します。このとき退避するデータを検査し、その結果を CCR に反映します。

(3) 使用可能な汎用レジスタ

ERn：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット									実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト			
－	PUSH.L	ERn	0	1	0	0	6	D	F	0	ern	5

(5) 注意事項

1. 本命令は、MOV.L ERn,@－SPと同一です。
2. PUSH.L ER7を実行すると実効アドレス計算（ER7－4→ER7実行）後のER7がスタックに退避されます。

2.2.89 ROTL (B) ROTate Left ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTL.B Rd	Rd（左ローテート）→Rd	バイト

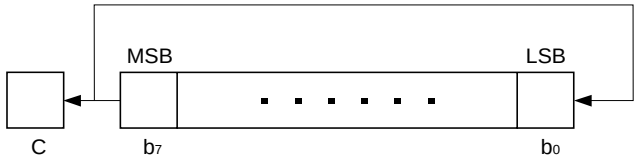
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 7 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向に 1 ビットローテート（回転）します。ローテートしてシフトアウトしたビットはビット 0 に戻り、かつキャリフラグに反映されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTL.B	Rd	1	2	8	rd			1

2.2.90 ROTL (B) ROTate Left ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTL.B #2, Rd	Rd（左ローテート）→Rd	バイト

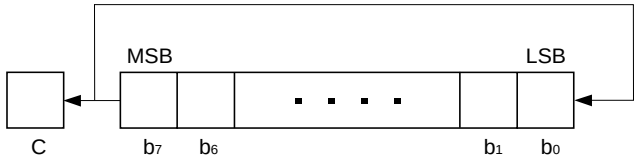
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 6 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向に 2 ビットローテート（回転）します。ローテートしてシフトアウトしたビット 7、6 は、それぞれビット 1、0 ビットに戻り、かつビット 6 はキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTL.B	#2, Rd	1	2	C	rd			1

2.2.91 ROTL (W) ROTate Left ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTL.W Rd	Rd（左ローテート）→Rd	ワード

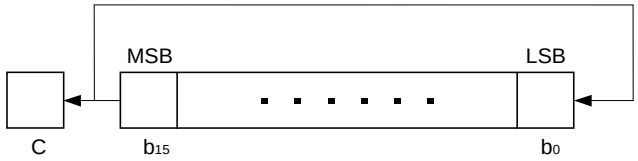
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 15 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向に 1 ビットローテート（回転）します。ローテートしてシフトアウトしたビットは、ビット 0 に戻り、かつキャリフラグに反映されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTL.W	Rd	1	2	9	rd			1

2.2.92 ROTL (W) ROTate Left ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTL.W #2, Rd	Rd（左ローテート）→Rd	ワード

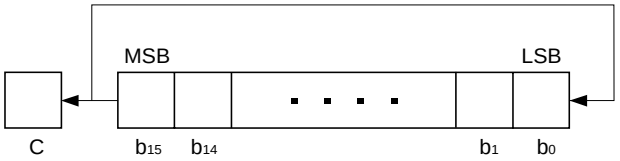
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 14 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向に 2 ビットローテート（回転）します。ローテートしてシフトアウトしたビット 15、14 は、それぞれビット 1、0 に戻り、かつビット 14 はキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTL.W	#2, Rd	1	2	D	rd			1

2.2.93 ROTL (L) ROTate Left ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTL.L ERd	ERd（左ローテート）→ERd	ロングワード

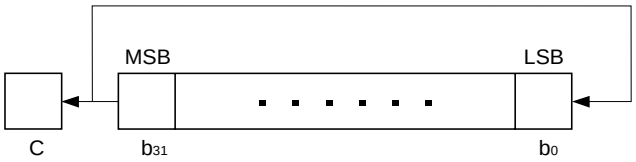
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 31 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、左方向に 1 ビットローテート（回転）します。ローテートしてシフトアウトしたビットはビット 0 に戻り、かつキャリフラグに反映されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	ROTL.L	ERd	1	2	B	0	erd			1

2.2.94 ROTL (L) ROTate Left ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTL.L #2, ERd	ERd（左ローテート）→ERd	ロングワード

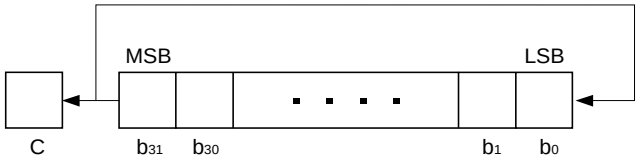
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 30 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、左方向に 2 ビットローテート（回転）します。ローテートしてシフトアウトしたビット 31、30 は、それぞれビット 1、0 に戻り、かつビット 30 はキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTL.L	#2, ERd	1	2	F	0 erd			1

2.2.95 ROTR (B) ROTate Right ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTR.B Rd	Rd（右ローテート）→Rd	バイト

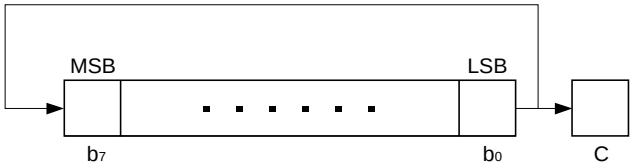
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向に 1 ビットローテート（回転）します。ローテートしてシフトアウトしたビットはビット 7 に戻り、かつキャリフラグに反映されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTR.B	Rd	1	3	8	rd			1

2.2.96 ROTR (B) ROTate Right ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTR.B #2, Rd	Rd（右ローテート）→Rd	バイト

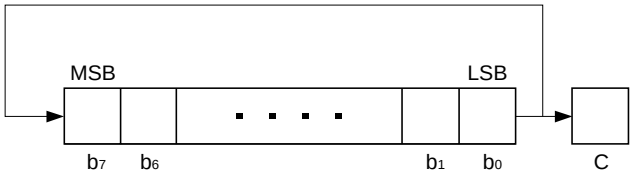
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向に 2 ビットローテート（回転）します。ローテートしてシフトアウトしたビット 1、0 はそれぞれビット 7、6 に戻り、かつビット 1 はキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTR.B	#2, Rd	1	3	C	rd			1

2.2.97 ROTR (W) ROTate Right ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTR.W Rd	Rd（右ローテート）→Rd	ワード

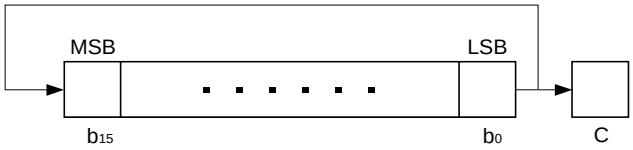
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向に 1 ビットローテート（回転）します。ローテートしてシフトアウトしたビットはビット 15 に戻り、かつキャリフラグに反映されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTR.W	Rd	1	3	9	rd			1

2.2.98 ROTR (W) ROTate Right ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTR.W #2, Rd	Rd（右ローテート）→Rd	ワード

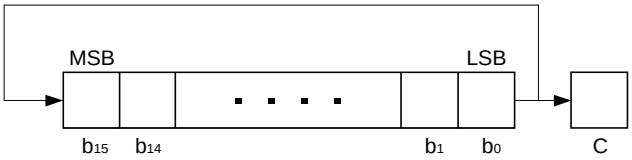
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向に 2 ビットローテート（回転）します。ローテートしてシフトアウトしたビット 1、0 はそれぞれビット 15、14 に戻り、かつビット 1 はキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTR.W	#2, Rd	1	3	D	rd			1

2.2.99 ROTR (L) ROTate Right ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTR.L ERd	ERd（右ローテート）→ERd	ロングワード

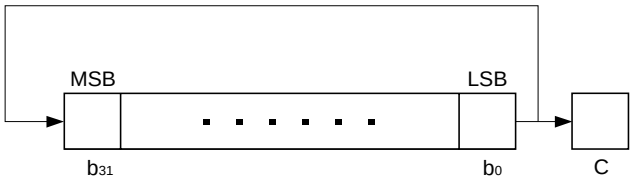
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、右方向に 1 ビットローテート（回転）します。ローテートしてシフトアウトしたビットはビット 31 に戻り、かつキャリフラグに反映されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	ROTR.L	ERd	1	3	B	0	erd			1

2.2.100 ROTR (L) ROTate Right ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTR.L #2, ERd	ERd（右ローテート）→ERd	ロングワード

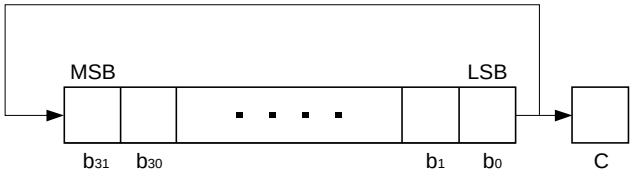
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、右方向に 2 ビットローテート（回転）します。ローテートしてシフトアウトしたビット 1、0 はそれぞれビット 31、30 に戻り、かつビット 1 はキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	ROTR.L	#2, ERd	1	3	F	0	erd			1

2.2.101 ROTXL (B) ROTate with eXtend carry Left キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXL.B Rd	Rd（キャリ付左ローテート）→Rd	バイト

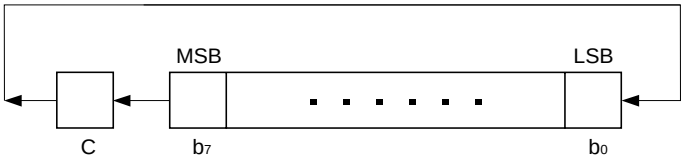
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 7 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて左方向に 1 ビットローテート（回転）します。ビット 0 にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXL.B	Rd	1	2	0	rd			1

2.2.102 ROTXL (B) ROTate with eXtend carry Left キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXL.B #2, Rd	Rd（キャリ付左ローテート）→Rd	バイト

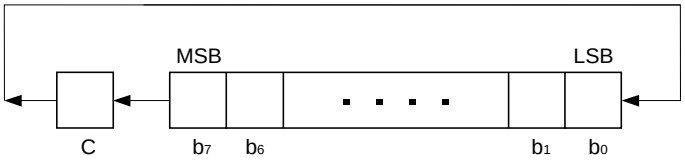
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 6 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて左方向に 2 ビットローテート（回転）します。ビット 1 にはキャリフラグの値が格納され、ビット 0 にはビット 7 の値が格納され、キャリフラグにはビット 6 の値が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXL.B	#2, Rd	1	2	4	rd			1

2.2.103 ROTXL (W) ROTate with eXtend carry Left キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXL.W Rd	Rd（キャリ付左ローテート）→Rd	ワード

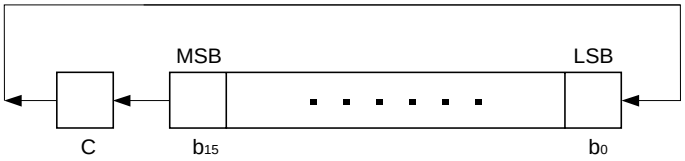
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 15 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて左方向に 1 ビットローテート（回転）します。ビット 0 にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXL.W	Rd	1	2	1	rd			1

2.2.104 ROTXL (W) ROTate with eXtend carry Left キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXL.W #2, Rd	Rd（キャリ付左ローテート）→Rd	ワード

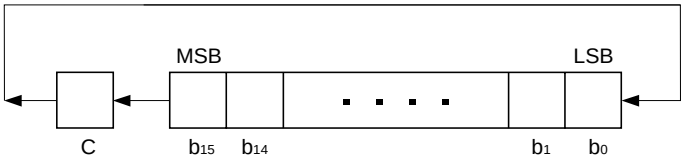
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 14 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて左方向に 2 ビットローテート（回転）します。ビット 1 にはキャリフラグの値が、ビット 0 にはビット 15 の値が、キャリフラグにはビット 14 の値がそれぞれ格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXL.W	#2, Rd	1	2	5	rd			1

2.2.105 ROTXL (L) ROTate with eXtend carry Left キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXL.L ERd	ERd（キャリ付左ローテート）→ERd	ロングワード

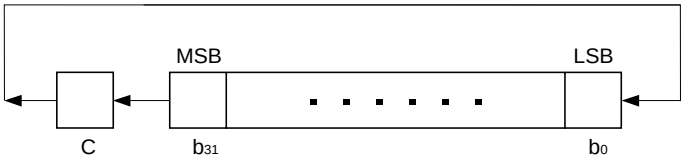
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 31 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて左方向に 1 ビットローテート（回転）します。ビット 0 にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト		
レジスタ直接	ROTXL.L	ERd	1	2	3	0	erd				1

2.2.106 ROTXL (L) ROTate with eXtend carry Left キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXL.L #2, ERd	ERd（キャリ付左ローテート）→ERd	ロングワード

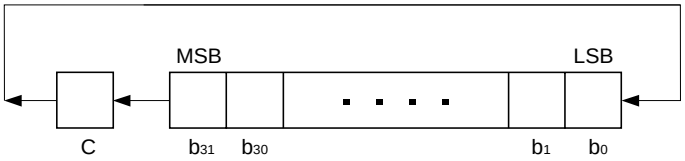
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 30 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて左方向に 2 ビットローテート（回転）します。ビット 1 にはキャリフラグの値が、ビット 0 にはビット 31 の値が、キャリフラグにはビット 30 の値がそれぞれ格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXL.L	#2, ERd	1	2	7	0 erd			1

2.2.107 ROTXR (B) ROTate with eXtend carry Right キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXR.B Rd	Rd（キャリ付右ローテート）→Rd	バイト

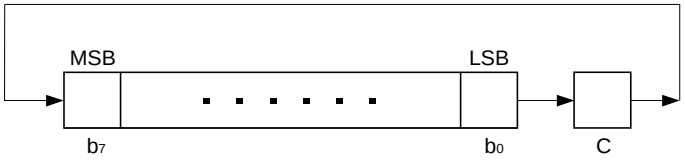
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて右方向に 1 ビットローテート（回転）します。ビット 7 にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXR.B	Rd	1	3	0	rd			1

2.2.108 ROTXR (B) ROTate with eXtend carry Right キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXR.B #2, Rd	Rd（キャリ付右ローテート）→Rd	バイト

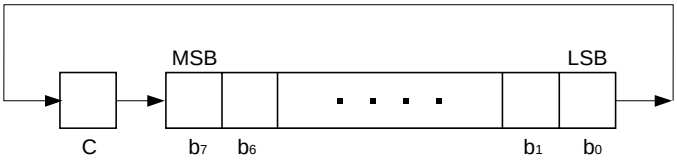
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて右方向に 2 ビットローテート（回転）します。ビット 6 にはキャリフラグの値が、ビット 7 にはビット 0 の値が、キャリフラグにはビット 1 の値がそれぞれ格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXR.B	#2, Rd	1	3	4	rd			1

2.2.109 ROTXR (W) ROTate with eXtend carry Right キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXR.W Rd	Rd（キャリ付右ローテート）→Rd	ワード

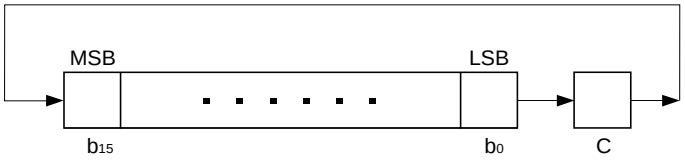
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて右方向に 1 ビットローテート（回転）します。ビット 15 にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXR.W	Rd	1	3	1	rd			1

2.2.110 ROTXR (W) ROTate with eXtend carry Right キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXR.W #2, Rd	Rd（キャリ付右ローテート）→Rd	ワード

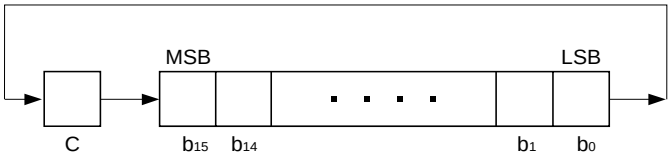
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて右方向に 2 ビットローテート（回転）します。ビット 14 にはキャリフラグの値が、ビット 15 にはビット 0 の値が、キャリフラグにはビット 1 の値がそれぞれ格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXR.W	#2, Rd	1	3	5	rd			1

2.2.111 ROTXR (L) ROTate with eXtend carry Right キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXR.L ERd	ERd（キャリ付右ローテート）→ERd	ロングワード

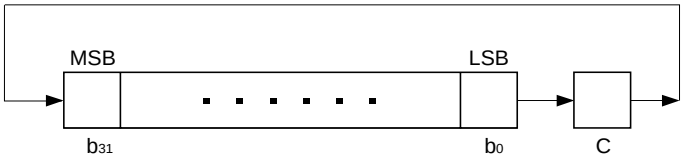
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて右方向に 1 ビットローテート（回転）します。ビット 31 にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXR.L	ERd	1	3	3	0 erd			1

2.2.112 ROTXR (L) ROTate with eXtend carry Right キャリ付ローテート

アセンブラフォーマット	オペレーション	オペランドサイズ
ROTXR.L #2, ERd	ERd（キャリ付右ローテート）→ERd	ロングワード

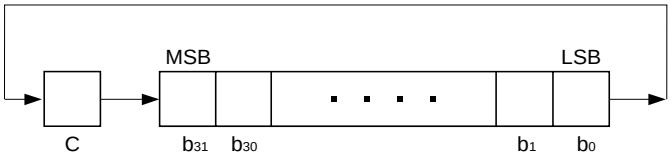
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、キャリフラグを含めて右方向に 2 ビットローテート（回転）します。ビット 30 にはキャリフラグの値が、ビット 31 にはビット 0 の値が、キャリフラグにはビット 1 の値がそれぞれ格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	ROTXR.L	#2, ERd	1	3	7	0 erd			1

2.2.113 RTE ReTurn from Exception 例外処理からのリターン

アセンブラフォーマット	オペレーション	オペランドサイズ
RTE	• EXR が無効のとき @SP+→CCR @SP+→PC • EXR が有効のとき @SP+→EXR @SP+→CCR @SP+→PC	—

(1) コンディションコード

I	UI	H	U	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

I ：スタックの内容の対応するビットの値が格納されます。
UI ：スタックの内容の対応するビットの値が格納されます。
H ：スタックの内容の対応するビットの値が格納されます。
U ：スタックの内容の対応するビットの値が格納されます。
N ：スタックの内容の対応するビットの値が格納されます。
Z ：スタックの内容の対応するビットの値が格納されます。
V ：スタックの内容の対応するビットの値が格納されます。
C ：スタックの内容の対応するビットの値が格納されます。

(2) 説明

例外処理ルーチンから復帰します。スタックから EXR、CCR と PC を復帰し、復帰した PC が示すアドレスから処理を行います。本命令を実行する直前の CCR および PC の内容は失われます。

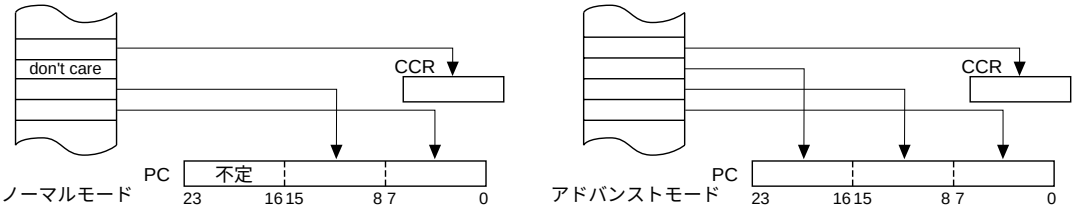
(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
—	RTE		5	6	7	0			5*

【注】* EXRが有効のときの実行ステート数は6ステートです。

(4) 注意事項

ノーマルモードとアドバンストモードでは、スタックの構造が異なりますので注意してください。



2.2.114 RTS ReTurn from Subroutine サブルーチンリターン

アセンブラフォーマット	オペレーション	オペランドサイズ
RTS	@SP＋→PC	—

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

サブルーチンから復帰します。スタックから PC を復帰し、復帰した PC が示すアドレスから処理を行います。本命令を実行する直前の PC の内容は失われます。

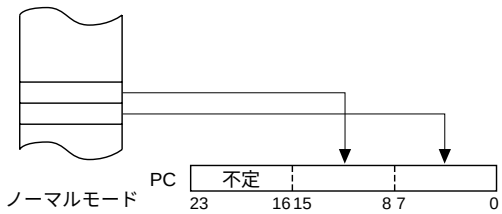
(3) オペランド形式と実行ステート数

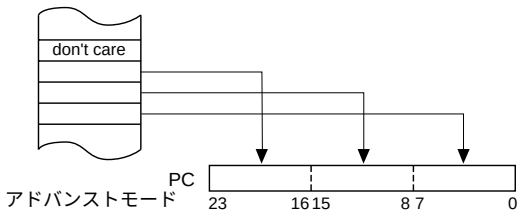
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数	
			第1バイト		第2バイト		第3バイト	第4バイト	ノーマル	アドバンスト
—	RTS		5	4	7	0			4	5

(4) 注意事項

ノーマルモードとアドバンストモードでは、スタックの構造および実行ステート数が異なりますので注意してください。

ノーマルモードのとき復帰される PC の内容は下位 16 ビットのみです。





2.2.115 SHAL (B) SHift Arithmetic Left 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAL.B Rd	Rd（左算術シフト）→Rd	バイト

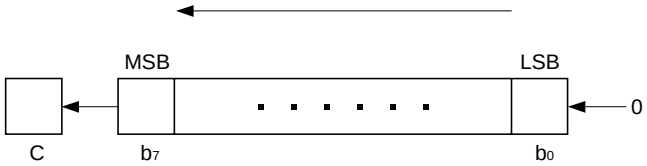
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	↕	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前のビット 7 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向へ算術的に 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 0 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAL.B	Rd	1	0	8	rd			1

(5) 注意事項

本命令と SHLL 命令とでは、オーバフローフラグの動作が異なります。

2.2.116 SHAL (B) SHift Arithmetic Left 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAL.B #2, Rd	Rd（左算術シフト）→Rd	バイト

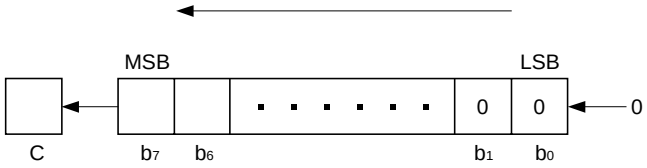
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	↕	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前のビット 6 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向へ算術的に 2 ビットシフトします。シフトアウトしたビット 6 はキャリフラグに格納され、ビット 0、ビット 1 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAL.B	#2, Rd	1	0	C	rd			1

(5) 注意事項

本命令と SHLL 命令とでは、オーバフローフラグの動作が異なります。

2.2.117 SHAL (W) SHift Arithmetic Left 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAL.W Rd	Rd（左算術シフト）→Rd	ワード

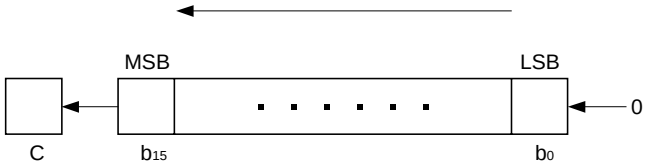
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	↕	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前のビット 15 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向へ算術的に 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 0 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAL.W	Rd	1	0	9	rd			1

(5) 注意事項

本命令と SHLL 命令とでは、オーバフローフラグの動作が異なります。

2.2.118 SHAL (W) SHift Arithmetic Left 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAL.W #2, Rd	Rd（左算術シフト）→Rd	ワード

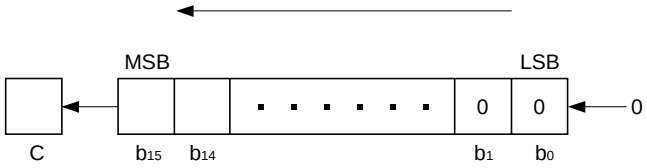
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	↕	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前のビット 14 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向へ算術的に 2 ビットシフトします。シフトアウトしたビット 14 はキャリフラグに格納され、ビット 0、ビット 1 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAL.W	#2, Rd	1	0	D	rd			1

(5) 注意事項

本命令と SHLL 命令とでは、オーバフローフラグの動作が異なります。

2.2.119 SHAL (L) SHift Arithmetic Left 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAL.L ERd	ERd（左算術シフト）→ERd	ロングワード

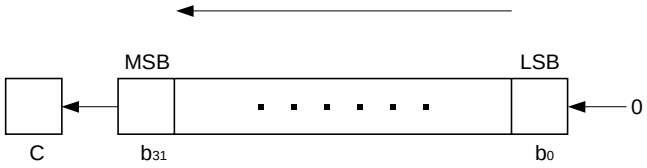
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	↕	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前のビット 31 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、左方向へ算術的に 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 0 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	SHAL.L	ERd	1	0	B	0	erd			1

(5) 注意事項

本命令と SHLL 命令とでは、オーバフローフラグの動作が異なります。

2.2.120 SHAL (L) SHift Arithmetic Left 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAL.L #2, ERd	ERd（左算術シフト）→ERd	ロングワード

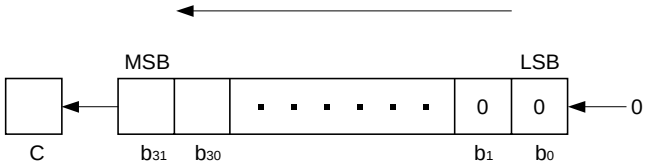
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	↕	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前のビット 30 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、左方向へ算術的に 2 ビットシフトします。シフトアウトしたビット 30 はキャリフラグに格納され、ビット 0、ビット 1 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	SHAL.L	#2, ERd	1	0	F	0	erd			1

(5) 注意事項

本命令と SHLL 命令とでは、オーバフローフラグの動作が異なります。

2.2.121 SHAR (B) SHift Arithmetic Right 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAR.B Rd	Rd（右算術シフト）→Rd	バイト

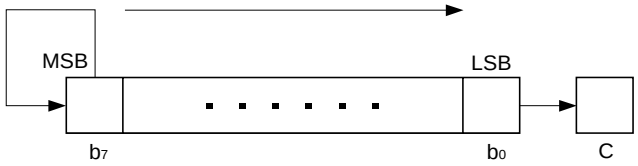
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向へ算術的に 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 7 にはシフト処理前のビット 7 がセットされます。ビット 7 は変化しないので、符号変化は起こりません。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAR.B	Rd	1	1	8	rd			1

2.2.122 SHAR (B) SHift Arithmetic Right 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAR.B #2, Rd	Rd（右算術シフト）→Rd	バイト

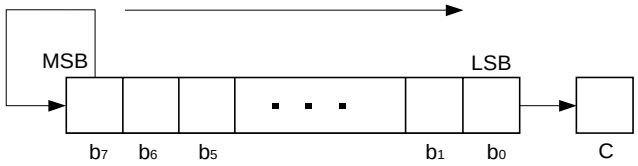
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向へ算術的に 2 ビットシフトします。シフトアウトしたビット 1 はキャリフラグに格納され、ビット 7、6 にはシフト処理前のビット 7 が格納されます。ビット 7 は変化しないので、符号変化は起こりません。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAR.B	#2, Rd	1	1	C	rd			1

2.2.123 SHAR (W) SHift Arithmetic Right 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAR.W Rd	Rd（右算術シフト）→Rd	ワード

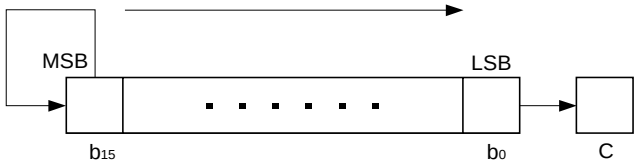
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向へ算術的に 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 15 にはシフト処理前のビット 15 が格納されます。ビット 15 は変化しないので、符号変化は起こりません。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAR.W	Rd	1	1	9	rd			1

2.2.124 SHAR (W) SHift Arithmetic Right 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAR.W #2, Rd	Rd（右算術シフト）→Rd	ワード

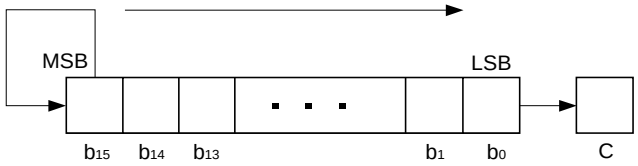
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向へ算術的に 2 ビットシフトします。シフトアウトしたビット 1 はキャリフラグに格納され、ビット 15、14 にはシフト処理前のビット 15 が格納されます。ビット 15 は変化しないので、符号変化は起こりません。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAR.W	#2, Rd	1	1	D	rd			1

2.2.125 SHAR (L) SHift Arithmetic Right 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAR.L ERd	ERd（右算術シフト）→ERd	ロングワード

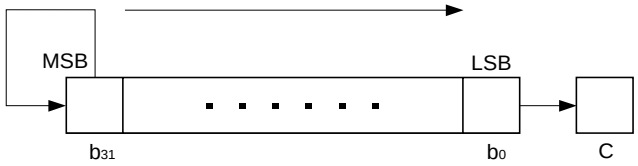
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、右方向へ算術的に 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 31 にはシフト処理前のビット 31 が格納されます。ビット 31 は変化しないので符号変化は起こりません。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	SHAR.L	ERd	1	1	B	0	erd			1

2.2.126 SHAR (L) SHift Arithmetic Right 算術シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHAR.L #2, ERd	ERd（右算術シフト）→ERd	ロングワード

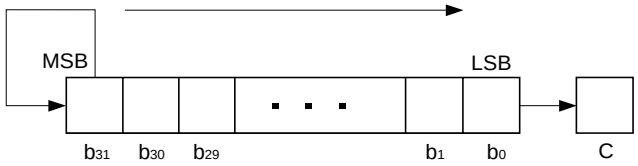
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、右方向へ算術的に 2 ビットシフトします。シフトアウトしたビット 1 はキャリフラグに格納され、ビット 31、30 にはシフト処理前のビット 31 が格納されます。ビット 31 は変化しないので、符号変化は起こりません。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHAR.L	#2, ERd	1	1	F	0 erd			1

2.2.127 SHLL (B) SHift Logical Left 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLL.B Rd	Rd（左論理シフト）→Rd	バイト

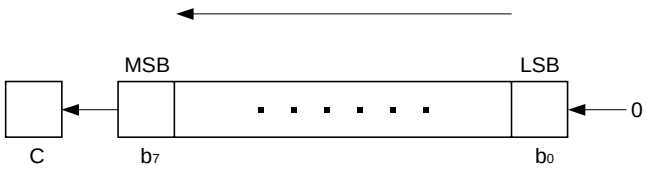
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 7 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向へ 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 0 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLL.B	Rd	1	0	0	rd			1

(5) 注意事項

本命令と SHAL 命令とでは、オーバフローフラグの動作が異なります。

2.2.128 SHLL (B) SHift Logical Left 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLL.B #2, Rd	Rd（左論理シフト）→Rd	バイト

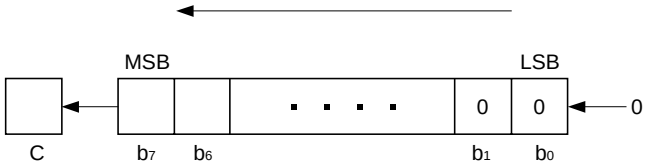
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 6 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向へ 2 ビットシフトします。シフトアウトしたビット 6 はキャリフラグに格納され、ビット 0、ビット 1 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLL.B	#2, Rd	1	0	4	rd			1

(5) 注意事項

本命令と SHAL 命令とでは、オーバフローフラグの動作が異なります。

2.2.129 SHLL (W) SHift Logical Left 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLL.W Rd	Rd（左論理シフト）→Rd	ワード

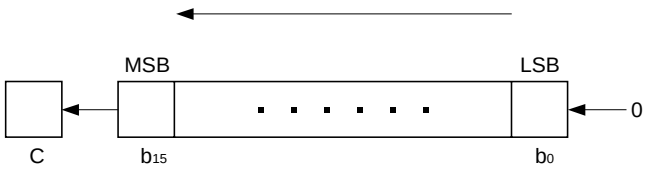
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 15 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向へ 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 0 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLL.W	Rd	1	0	1	rd			1

(5) 注意事項

本命令と SHAL 命令とでは、オーバフローフラグの動作が異なります。

2.2.130 SHLL (W) SHift Logical Left 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLL.W #2, Rd	Rd（左論理シフト）→Rd	ワード

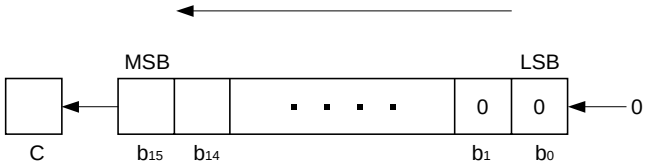
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 14 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、左方向へ 2 ビットシフトします。シフトアウトしたビット 14 はキャリフラグに格納され、ビット 0、ビット 1 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLL.W	#2, Rd	1	0	5	rd			1

(5) 注意事項

本命令と SHAL 命令とでは、オーバフローフラグの動作が異なります。

2.2.131 SHLL (L) SHift Logical Left 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLL.L ERd	ERd（左論理シフト）→ERd	ロングワード

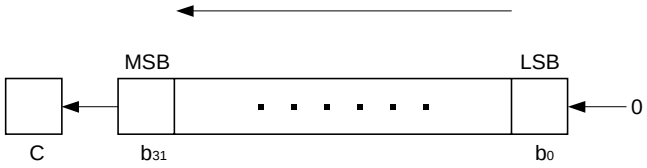
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 31 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、左方向へ 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 0 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLL.L	ERd	1	0	3	0 erd			1

(5) 注意事項

本命令と SHAL 命令とでは、オーバフローフラグの動作が異なります。

2.2.132 SHLL (L) SHift Logical Left 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLL.L #2, ERd	ERd（左論理シフト）→ERd	ロングワード

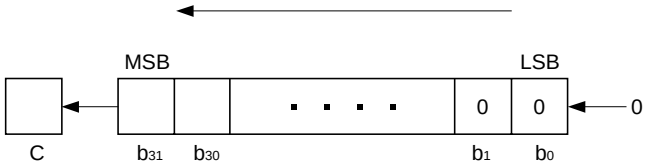
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕	↕	0	↕

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 30 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、左方向へ 2 ビットシフトします。シフトアウトしたビット 30 はキャリフラグに格納され、ビット 0、ビット 1 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLL.L	#2, ERd	1	0	7	0 erd			1

(5) 注意事項

本命令と SHAL 命令とでは、オーバフローフラグの動作が異なります。

2.2.133 SHLR (B) SHift Logical Right 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLR.B Rd	Rd（右論理シフト）→Rd	バイト

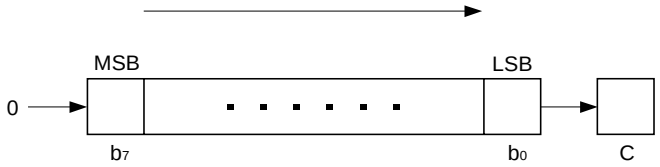
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	0	↕	0	↕

H：実行前の値が保持されます。
N：常に 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向へ 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 7 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLR.B	Rd	1	1	0	rd			1

2.2.134 SHLR (B) SHift Logical Right 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLR.B #2, Rd	Rd（右論理シフト）→Rd	バイト

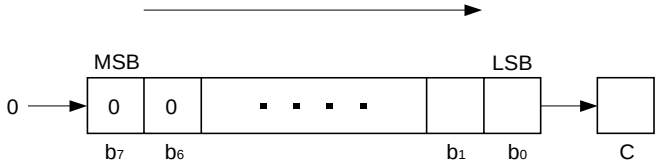
(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	0	↕	0	↕

H：実行前の値が保持されます。
N：常に 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向へ 2 ビットシフトします。シフトアウトしたビット 1 はキャリフラグに格納され、ビット 7、ビット 6 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLR.B	#2, Rd	1	1	4	rd			1

2.2.135 SHLR (W) SHift Logical Right 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLR.W Rd	Rd（右論理シフト）→Rd	ワード

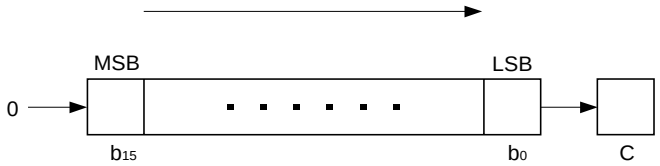
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	0	↕	0	↕

H：実行前の値が保持されます。
N：常に 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向へ 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 15 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLR.W	Rd	1	1	1	rd			1

2.2.136 SHLR (W) SHift Logical Right 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLR.W #2, Rd	Rd（右論理シフト）→Rd	ワード

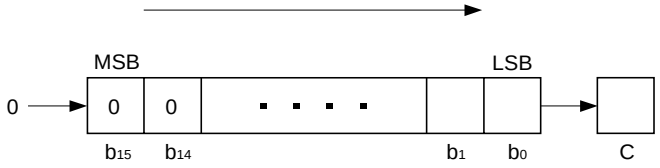
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	0	↕	0	↕

H：実行前の値が保持されます。
N：常に 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）のビット群を、右方向へ 2 ビットシフトします。シフトアウトしたビット 1 はキャリフラグに格納され、ビット 15、ビット 14 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLR.W	#2, Rd	1	1	5	rd			1

2.2.137 SHLR (L) SHift Logical Right 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLR.L ERd	ERd（右論理シフト）→ERd	ロングワード

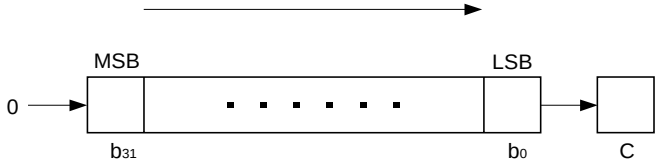
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	0	↕	0	↕

H：実行前の値が保持されます。
N：常に 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 0 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、右方向へ 1 ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット 31 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLR.L	ERd	1	1	3	0 erd			1

2.2.138 SHLR (L) SHift Logical Right 論理シフト

アセンブラフォーマット	オペレーション	オペランドサイズ
SHLR.L #2, ERd	ERd（右論理シフト）→ERd	ロングワード

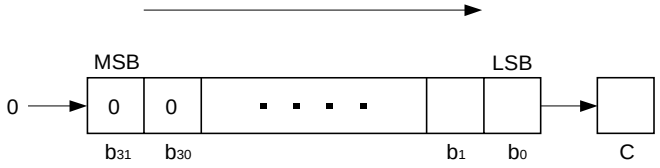
(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	0	↕	0	↕

H：実行前の値が保持されます。
N：常に 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前のビット 1 の値が格納されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）のビット群を、右方向へ 2 ビットシフトします。シフトアウトしたビット 1 はキャリフラグに格納され、ビット 31、ビット 30 には 0 が格納されます。



(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SHLR.L	#2, ERd	1	1	7	0 erd			1

2.2.139 SLEEP SLEEP 低消費電力状態命令

アセンブラフォーマット	オペレーション	オペランドサイズ
SLEEP	プログラム実行状態→低消費電力状態	—

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

SLEEP 命令を実行すると、CPU は低消費電力状態に入ります。低消費電力状態では、CPU の内部状態は保持され、命令の実行を停止し、例外処理要求の発生を待ち続けます。例外処理要求が発生すると、低消費電力状態は解除され、CPU は例外処理を開始します。このとき NMI 以外の割り込み要求では、CPU 側で割り込みがマスクされている場合、低消費電力状態は解除されません。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
—	SLEEP		0	1	8	0			2

(4) 注意事項

低消費電力状態については、当該製品のハードウェアマニュアルを参照してください。

2.2.140 STC(B) STore from Control register CCR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
STC.B CCR, Rd	CCR→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

CCR の内容を 8 ビットレジスタ Rd に転送します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	STC.B	CCR,Rd	0	2	0	rd			1

2.2.141 STC(B) STore from Control register EXR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
STC.B EXR, Rd	EXR→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

EXR の内容を 8 ビットレジスタ Rd に転送します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	STC.B	EXR,Rd	0	2	1	rd			1

2.2.142 STC(W) STore from Control register CCR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
STC.W CCR, ＜EAd＞	CCR→（EAd）	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

CCR の内容をデスティネーションのロケーションに転送します。CCR はバイトサイズですが転送はワードサイズで行われ、偶数アドレスに CCR の内容が格納されます。奇数アドレスは不定値が格納されます。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット														実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト	第8バイト	第9バイト	第10バイト	
レジスタ間接	STC.W	CCR,@ERd	0	1	4	0	6	9	1: erd	0									3
ディスプレースメント付レジスタ間接	STC.W	CCR,@(d16,ERd)	0	1	4	0	6	F	1: erd	0	disp								4
	STC.W	CCR,@(d32,ERd)	0	1	4	0	7	8	0: erd	0	6	B	A	0			disp		6
プリデクリメント/ポストインクリメントレジスタ間接	STC.W	CCR,@-ERd	0	1	4	0	6	D	1: erd	0									4
絶対アドレス	STC.W	CCR,@aa:16	0	1	4	0	6	B	8	0	abs								4
	STC.W	CCR,@aa:32	0	1	4	0	6	B	A	0	abs								5

2.2.143 STC(W) STore from Control register EXR 転送

アセンブラフォーマット	オペレーション	オペランドサイズ
STC.W EXR, ＜EAd＞	EXR→（EAd）	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

EXR の内容をデスティネーションのロケーションに転送します。EXR はバイトサイズですが転送はワードサイズで行われ、偶数アドレスに EXR の内容が格納されます。奇数アドレスは不定値が格納されます。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット										実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト	
レジスタ間接	STC.W	EXR,@ERd	0 1	4 1	6 9	1 erd 0							3
ディスプレースメント付レジスタ間接	STC.W	EXR,@(d:16,ERd)	0 1	4 1	6 F	1 erd 0	disp						4
	STC.W	EXR,@(d:32,ERd)	0 1	4 1	7 8	0 erd 0	6	B A 0	disp				6
プリデクリメント／ポストインクリメントレジスタ間接	STC.W	EXR,@-ERd	0 1	4 1	6 D	1 erd 0							4
絶対アドレス	STC.W	EXR,@aa:16	0 1	4 1	6 B	8 0	abs						4
	STC.W	EXR,@aa:32	0 1	4 1	6 B	A 0	abs						5

2.2.144 STM STore from Multiple register スタックへデータ退避

アセンブラフォーマット	オペレーション	オペランドサイズ
STM.L ＜レジスタリスト＞, @-SP	ERn（レジスタ群）→@-SP	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	−	−	−	−

H ：実行前の値が保持されます。
N ：実行前の値が保持されます。
Z ：実行前の値が保持されます。
V ：実行前の値が保持されます。
C ：実行前の値が保持されます。

(2) 説明

レジスタリストで指定した複数のレジスタを、スタックに退避します。
退避はレジスタ番号の小さい順に行われます。
1 命令で退避できるレジスタ数は 2 本、3 本、4 本です。そのとき指定可能なレジスタリストは以下の通りです。
2 本：ER0 - ER1、ER2 - ER3、ER4 - ER5、ER6 - ER7
3 本：ER0 - ER2、ER4 - ER6
4 本：ER0 - ER3、ER4 - ER7

(3) 使用可能な汎用レジスタ

ERn：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット									実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト			
−	STM.L	(ERn-ERn+1),@-SP	0	1	1	0	6	D	F	0	ern	7
−	STM.L	(ERn-ERn+2),@-SP	0	1	2	0	6	D	F	0	ern	9
−	STM.L	(ERn-ERn+3),@-SP	0	1	3	0	6	D	F	0	ern	11

(5) 注意事項

ER7 の退避を行うときは、実行アドレス計算（ER7-4→ER7 実行）後の ER7 がスタックに退避されます。

2.2.145 STMAC STore from MAC register MAC レジスタ転送

アセンブラフォーマット	オペレーション	オペランドサイズ
STMAC MAC レジスタ, ERd	MACH→ERd または MACL→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
–	–	–	–	↕ *	↕ *	↕ *	–

H：実行前の値が保持されます。
N：MAC 命令の結果、MAC レジスタの値が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：MAC 命令の結果、MAC レジスタの値が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：MAC 命令の結果、オーバフローが発生していれば 1 にセットされ、それ以外のときは 0 にクリアされます。
C：実行前の値が保持されます。

【注】 * 本命令を実行することで、乗算器内部のフラグ（N、Z、V）を CCR に反映します。
ただし、CLRMAC または LDMAC 命令実行後、MAC 命令を行わずに STMAC を実行した場合は V フラグは 0、N、Z フラグは不定となります。

(2) 説明

MAC レジスタ（MACH、MACL）の内容を汎用レジスタに転送します。MACH からの転送では、上位 22 ビットが符号拡張されて汎用レジスタに転送されます。

H8S/2600 CPU でのみサポートしています。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト		
レジスタ直接	STMAC	MACH, ERd	0	2	2	0	erd				1*
レジスタ直接	STMAC	MACL, ERd	0	2	3	0	erd				1*

【注】 * MAC命令実行後3ステート以内にSTMAC命令を実行しようとした場合、最大で3ステート多くかかります。例えば、MAC命令とSTMAC命令の間に1ステート命令（NOP等）が1つある場合、STMAC命令は2ステート多くかかります。
製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

2.2.146 SUB (B) SUBtract binary 2 進減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUB.B Rs, Rd	Rd－Rs→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	↕	－	↕	↕	↕	↕

H：ビット 3 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 7 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）から 8 ビットレジスタ Rs の内容（ソースオペランド）を減算し、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd ：R0L～R7L、R0H～R7H
Rs ：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
レジスタ直接	SUB.B	Rs,Rd	1	8	rs	rd			1

(5) 注意事項

本命令は汎用レジスタ間の減算のみ可能です。汎用レジスタの内容とイミディエイトデータの減算は SUBX.B 命令を使用することにより実現できます。この場合、「SUBX.B #xx:8, Rd」を実行する前に、Z フラグを 1 にセットし、C フラグを 0 にクリアしてください。また、イミディエイトデータ #IMM≠0 の場合、次のプログラム例も使用できます。

（1） ORC #H'05, CCR
SUBX #(IMM－1), Rd
（2） ADD #(0－IMM), Rd
XORC #H'01, CCR

2.2.147 SUB (W) SUBtract binary 2 進減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUB.W ＜EAs＞, Rd	Rd－（EAs）→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	↕	－	↕	↕	↕	↕

H：ビット 11 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 15 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）からソースオペランドを減算し、結果を 16 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7
Rs：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	SUB.W	#xx:16,Rd	7	9	3	rd	IMM		2
レジスタ直接	SUB.W	Rs,Rd	1	9	rs	rd			1

2.2.148 SUB (L) SUBtract binary 2 進減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUB.L ＜EAs＞, ERd	ERd－（EAs）→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	↕	－	↕	↕	↕	↕

H：ビット 27 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 31 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

32 ビットレジスタ ERd の内容(デスティネーションオペランド)からソースオペランドを減算し、結果を 32 ビットレジスタ ERd に格納します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット										実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	第5バイト	第6バイト			
イミディエイト	SUB.L	#xx:32,ERd	7	A	3	0 erd	IMM						3
レジスタ直接	SUB.L	ERs,ERd	1	A	1 ers	0 erd							1

2.2.149 SUBS SUBtract with Sign extention アドレスデータ 2 進減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUBS #1, ERd	ERd－1→ERd	ロングワード
SUBS #2, ERd	ERd－2→ERd	
SUBS #4, ERd	ERd－4→ERd	

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	－	－	－	－	－	－

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容(デスティネーションオペランド)から 1､2 または 4 を減算します。
SUB 命令とは異なり、コンディションコードは実行前の値を保持します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット							実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	SUBS	#1,ERd	1	B	0	0	erd			1
レジスタ直接	SUBS	#2,ERd	1	B	8	0	erd			1
レジスタ直接	SUBS	#4,ERd	1	B	9	0	erd			1

2.2.150 SUBX SUBtract with eXtend carry キャリ付減算

アセンブラフォーマット	オペレーション	オペランドサイズ
SUBX ＜EAs＞, Rd	Rd－（EAs）－C→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
－	－	↕	－	↕	↕	↕	↕

H：ビット 3 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき実行前の値が保持され、それ以外のときは 0 にクリアされます。
V：オーバフローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。
C：ビット 7 にボローが発生したとき 1 にセットされ、それ以外のときは 0 にクリアされます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）からソースオペランドとキャリフラグの値を減算し、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	SUBX	#xx:8,Rd	B	rd	IMM				1
レジスタ直接	SUBX	Rs,Rd	1	E	rs	rd			1

2.2.151 TAS Test And Set テストアンドセット

アセンブラフォーマット	オペレーション	オペランドサイズ
TAS @ERd	@ERd−0 →CCR セット 1 →（<ビット 7>of @ERd）	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

メモリの内容をテスト（0 と比較）し、その結果をコンディションコードレジスタにセットします。その後、オペランドの最上位ビット（ビット 7）を 1 にセットします。

(3) 使用可能な汎用レジスタ

ERd：ER0、ER1、ER4、ER5

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット									実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト			
レジスタ間接	TAS	@ERd	0	1	E	0	7	B	0	erd	C	4

2.2.152 TRAPA TRAP Always 無条件トラップ

アセンブラフォーマット	オペレーション	オペランドサイズ
TRAPA #x:2	• EXR が無効のとき PC→@－SP CCR→@－SP ＜ベクタ＞→PC • EXR が有効のとき PC→@－SP CCR→@－SP EXR→@－SP ＜ベクタ＞→PC	－

(1) コンディションコード

I	UI	H	U	N	Z	V	C
1	△*	－	－	－	－	－	－

I ：常に 1 にセットされます。
UI：【注】を参照してください。
H ：実行前の値が保持されます。
N ：演算前の値が保持されます。
Z ：演算前の値が保持されます。
V ：演算前の値が保持されます。
C ：演算前の値が保持されます。

【注】 * 割り込みマスクビットとして使用しているとき 1 にセットされます。ユーザビットとして使用しているときは実行前の値が保持されます。詳細は、当該製品のハードウェアマニュアルを参照してください。

(2) 説明

プログラムカウンタ（PC）とコンディションコードレジスタ（CCR）をスタックに退避し、I ビットを 1 にセットします。また、エクステンドレジスタ（EXR）が有効のときは EXR をスタックに退避しますが、I2～I0 ビットは書き換えられません。次に指定した番号に対応するベクタアドレスの内容によって示されるアドレスへ分岐します。

退避する PC の値は本命令の直後の命令の先頭アドレスになります。

#x	ベクタアドレス	
	ノーマルモード	アドバンストモード
0	H' 0010～H' 0011	H' 000020～H' 000023
1	H' 0012～H' 0013	H' 000024～H' 000027
2	H' 0014～H' 0015	H' 000028～H' 00002B
3	H' 0016～H' 0017	H' 00002C～H' 00002F

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト			第3バイト	第4バイト	
レジスタ直接	TRAPA	#x:2	5	7	00	IMM	0			7*

【注】* EXRが有効のときの実行ステート数は8ステートです。

(4) 注意事項

ノーマルモードとアドバンストモードおよび EXR 有効と無効ではスタックおよびベクタの構造が異なりますので注意してください。

2.2.153 XOR (B) eXclusive OR logical 排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
XOR.B ＜EAs＞, Rd	Rd ⊕（EAs）→Rd	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

8 ビットレジスタ Rd の内容（デスティネーションオペランド）と、ソースオペランドの排他的論理和をとり、結果を 8 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0L～R7L、R0H～R7H
Rs：R0L～R7L、R0H～R7H

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	XOR.B	#xx:8,Rd	D	rd	IMM				1
レジスタ直接	XOR.B	Rs,Rd	1	5	rs	rd			1

2.2.154 XOR (W) eXclusive OR logical 排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
XOR.W ＜EAs＞, Rd	Rd ⊕（EAs）→Rd	ワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	↕	↕	0	—

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

16 ビットレジスタ Rd の内容（デスティネーションオペランド）と、ソースオペランドの排他的論理和をとり、結果を 16 ビットレジスタ Rd に格納します。

(3) 使用可能な汎用レジスタ

Rd：R0～R7、E0～E7
Rs：R0～R7、E0～E7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数
			第1バイト		第2バイト		第3バイト	第4バイト	
イミディエイト	XOR.W	#xx:16,Rd	7	9	5	rd	IMM		2
レジスタ直接	XOR.W	Rs,Rd	6	5	rs	rd			1

2.2.155 XOR (L) eXclusive OR logical 排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
XOR.L ＜EAs＞, ERd	ERd⊕（EAs）→ERd	ロングワード

(1) コンディションコード

I	UI	H	U	N	Z	V	C
−	−	−	−	↕	↕	0	−

H：実行前の値が保持されます。
N：実行結果が負のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
Z：実行結果が 0（ゼロ）のとき 1 にセットされ、それ以外のときは 0 にクリアされます。
V：常に 0 にクリアされます。
C：実行前の値が保持されます。

(2) 説明

32 ビットレジスタ ERd の内容（デスティネーションオペランド）と、ソースオペランドとの排他的論理和をとり、結果を 32 ビットレジスタ ERd に格納します。

(3) 使用可能な汎用レジスタ

ERd：ER0～ER7
ERs：ER0～ER7

(4) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット												実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト	第6バイト			
イミディエイト	XOR.L	#xx:32,ERd	7	A	5	0 erd	IMM								3
レジスタ直接	XOR.L	ERs,ERd	0	1	F	0	6	5	0 ers	0 erd					2

2.2.156 XORC eXclusive OR Control register CCR との排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
XORC #xx:8, CCR	CCR⊕#IMM→CCR	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
↕	↕	↕	↕	↕	↕	↕	↕

I ：実行結果の対応するビットの値が格納されます。
UI：実行結果の対応するビットの値が格納されます。
H ：実行結果の対応するビットの値が格納されます。
U ：実行結果の対応するビットの値が格納されます。
N ：実行結果の対応するビットの値が格納されます。
Z ：実行結果の対応するビットの値が格納されます。
V ：実行結果の対応するビットの値が格納されます。
C ：実行結果の対応するビットの値が格納されます。

(2) 説明

CCR の内容とイミディエイトデータとの排他的論理和をとり、結果を CCR に格納します。
本命令の実行終了時点では、NMI を含めてすべての割り込みは受け付けられません。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット					実行ステート数
			第1バイト		第2バイト	第3バイト	第4バイト	
イミディエイト	XORC	#xx:8,CCR	0	5	IMM			1

2.2.157 XORC eXclusive OR Control register EXR との排他的論理和

アセンブラフォーマット	オペレーション	オペランドサイズ
XORC #xx:8, EXR	EXR⊕#IMM→EXR	バイト

(1) コンディションコード

I	UI	H	U	N	Z	V	C
—	—	—	—	—	—	—	—

H：実行前の値が保持されます。
N：実行前の値が保持されます。
Z：実行前の値が保持されます。
V：実行前の値が保持されます。
C：実行前の値が保持されます。

(2) 説明

EXR の内容とイミディエイトデータとの排他的論理和をとり、結果を EXR に格納します。

なお、本命令の実行終了後 3 ステートの間は、NMI を含めてすべての割り込みは受け付けられません。

(3) オペランド形式と実行ステート数

アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット							実行ステート数
			第1バイト		第2バイト		第3バイト		第4バイト	
イミディエイト	XORC	#xx:8,EXR	0	1	4	1	0	5	IMM	2

2.3 命令セット一覧

表 2.1 命令セット一覧

(1) データ転送命令

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション	コンディションコード						実行ステート数*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	ノーマル	アドバンスト
MOV	MOV.B #xx:8,Rd	B	2									#xx:8→Rd8	—	—	↕	↕	0	—	1	
	MOV.B Rs,Rd	B		2								Rs8→Rd8	—	—	↕	↕	0	—	1	
	MOV.B @ERs,Rd	B			2							@ERs→Rd8	—	—	↕	↕	0	—	2	
	MOV.B @(d:16,ERs),Rd	B				4						@(d:16,ERs)→Rd8	—	—	↕	↕	0	—	3	
	MOV.B @(d:32,ERs),Rd	B				8						@(d:32,ERs)→Rd8	—	—	↕	↕	0	—	5	
	MOV.B @ERs+,Rd	B					2					@ERs→Rd8,ERs32+1→ERs32	—	—	↕	↕	0	—	3	
	MOV.B @aa:8,Rd	B						2				@aa:8→Rd8	—	—	↕	↕	0	—	2	
	MOV.B @aa:16,Rd	B						4				@aa:16→Rd8	—	—	↕	↕	0	—	3	
	MOV.B @aa:32,Rd	B						6				@aa:32→Rd8	—	—	↕	↕	0	—	4	
	MOV.B Rs,@ERd	B			2							Rs8→@ERd	—	—	↕	↕	0	—	2	
	MOV.B Rs,@(d:16,ERd)	B				4						Rs8→@(d:16,ERd)	—	—	↕	↕	0	—	3	
	MOV.B Rs,@(d:32,ERd)	B				8						Rs8→@(d:32,ERd)	—	—	↕	↕	0	—	5	
	MOV.B Rs,@-ERd	B					2					ERd32-1→ERd32,Rs8→@ERd	—	—	↕	↕	0	—	3	
	MOV.B Rs,@aa:8	B						2				Rs8→@aa:8	—	—	↕	↕	0	—	2	
	MOV.B Rs,@aa:16	B						4				Rs8→@aa:16	—	—	↕	↕	0	—	3	
	MOV.B Rs,@aa:32	B						6				Rs8→@aa:32	—	—	↕	↕	0	—	4	
	MOV.W #xx:16,Rd	W	4									#xx:16→Rd16	—	—	↕	↕	0	—	2	
	MOV.W Rs,Rd	W		2								Rs16→Rd16	—	—	↕	↕	0	—	1	
	MOV.W @ERs,Rd	W			2							@ERs→Rd16	—	—	↕	↕	0	—	2	
	MOV.W @(d:16,ERs),Rd	W				4						@(d:16,ERs)→Rd16	—	—	↕	↕	0	—	3	
	MOV.W @(d:32,ERs),Rd	W				8						@(d:32,ERs)→Rd16	—	—	↕	↕	0	—	5	
	MOV.W @ERs+,Rd	W					2					@ERs→Rd16,ERs32+2→ERs32	—	—	↕	↕	0	—	3	
	MOV.W @aa:16,Rd	W						4				@aa:16→Rd16	—	—	↕	↕	0	—	3	
	MOV.W @aa:32,Rd	W						6				@aa:32→Rd16	—	—	↕	↕	0	—	4	
	MOV.W Rs,@ERd	W			2							Rs16→@ERd	—	—	↕	↕	0	—	2	
	MOV.W Rs,@(d:16,ERd)	W				4						Rs16→@(d:16,ERd)	—	—	↕	↕	0	—	3	
	MOV.W Rs,@(d:32,ERd)	W				8						Rs16→@(d:32,ERd)	—	—	↕	↕	0	—	5	
	MOV.W Rs,@-ERd	W					2					ERd32-2→ERd32,Rs16→@ERd	—	—	↕	↕	0	—	3	
	MOV.W Rs,@aa:16	W						4				Rs16→@aa:16	—	—	↕	↕	0	—	3	
	MOV.W Rs,@aa:32	W						6				Rs16→@aa:32	—	—	↕	↕	0	—	4	
	MOV.L #xx:32,ERd	L	6									#xx:32→ERd32	—	—	↕	↕	0	—	3	
	MOV.L ERs,ERd	L		2								ERs32→ERd32	—	—	↕	↕	0	—	1	
	MOV.L @ERs,ERd	L			4							@ERs→ERd32	—	—	↕	↕	0	—	4	
	MOV.L @(d:16,ERs),ERd	L				6						@(d:16,ERs)→ERd32	—	—	↕	↕	0	—	5	
	MOV.L @(d:32,ERs),ERd	L				10						@(d:32,ERs)→ERd32	—	—	↕	↕	0	—	7	
	MOV.L @ERs+,ERd	L					4					@ERs→ERd32,ERs32+4→ERs32	—	—	↕	↕	0	—	5	
	MOV.L @aa:16,ERd	L						6				@aa:16→ERd32	—	—	↕	↕	0	—	5	
	MOV.L @aa:32,ERd	L						8				@aa:32→ERd32	—	—	↕	↕	0	—	6	
	MOV.L ERs,@ERd	L			4							ERs32→@ERd	—	—	↕	↕	0	—	4	
	MOV.L ERs,@(d:16,ERd)	L				6						ERs32→@(d:16,ERd)	—	—	↕	↕	0	—	5	
	MOV.L ERs,@(d:32,ERd)	L				10						ERs32→@(d:32,ERd)	—	—	↕	↕	0	—	7	
	MOV.L ERs,@-ERd	L					4					ERd32-4→ERd32,ERs32→@ERd	—	—	↕	↕	0	—	5	
	MOV.L ERs,@aa:16	L						6				ERs32→@aa:16	—	—	↕	↕	0	—	5	
	MOV.L ERs,@aa:32	L						8				ERs32→@aa:32	—	—	↕	↕	0	—	6	
POP	POP.W Rn	W									2	@SP→Rn16,SP+2→SP	—	—	↕	↕	0	—	3	
	POP.L ERn	L									4	@SP→ERn32,SP+4→SP	—	—	↕	↕	0	—	5	
PUSH	PUSH.W Rn	W									2	SP-2→SP,Rn16→@SP	—	—	↕	↕	0	—	3	
	PUSH.L ERn	L									4	SP-4→SP,ERn32→@SP	—	—	↕	↕	0	—	5	
LDM	LDM @SP+,(ERm-ERn)	L									4	(@SP→ERn32,SP+4→SP) 復帰本数分繰り返し	—	—	—	—	—	—	7/9/11*4	
STM	STM (ERm-ERn),@-SP	L									4	(SP-4→SP,ERn32→@SP) 退避本数分繰り返し	—	—	—	—	—	—	7/9/11*4	
MOVFPE	MOVFPE @aa:16,Rd	B						4				@aa:16→Rd(Eクロック同期)	—	—	↕	↕	0	—	[1]	
MOVTPE	MOVTPE Rs,@aa:16	B						4				Rs→@aa:16(Eクロック同期)	—	—	↕	↕	0	—	[1]	

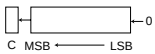
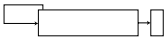
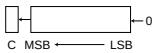
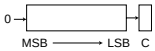
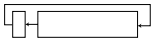
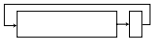
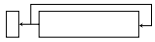
(2) 算術演算命令

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション	コンディションコード						実行ステート数*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	ノーマル	アドバンスト
ADD	ADD.B #xx:8,Rd	B	2									Rd8+#xx:8→Rd8	—	↕	↕	↕	↕	↕		1
	ADD.B Rs,Rd	B		2								Rd8+Rs8→Rd8	—	↕	↕	↕	↕	↕		1
	ADD.W #xx:16,Rd	W	4									Rd16+#xx:16→Rd16	—	[2]	↕	↕	↕	↕		2
	ADD.W Rs,Rd	W		2								Rd16+Rs16→Rd16	—	[2]	↕	↕	↕	↕		1
	ADD.L #xx:32,ERd	L	6									ERd32+#xx:32→ERd32	—	[3]	↕	↕	↕	↕		3
	ADD.L ERs,ERd	L		2								ERd32+ERs32→ERd32	—	[3]	↕	↕	↕	↕		1
ADDX	ADDX #xx:8,Rd	B	2									Rd8+#xx:8+C→Rd8	—	↕	↕	[4]	↕	↕		1
	ADDX Rs,Rd	B		2								Rd8+Rs8+C→Rd8	—	↕	↕	[4]	↕	↕		1
ADDS	ADDS #1,ERd	L		2								ERd32+1→ERd32	—	—	—	—	—	—		1
	ADDS #2,ERd	L		2								ERd32+2→ERd32	—	—	—	—	—	—		1
	ADDS #4,ERd	L		2								ERd32+4→ERd32	—	—	—	—	—	—		1
INC	INC.B Rd	B		2								Rd8+1→Rd8	—	—	↕	↕	↕	—		1
	INC.W #1,Rd	W		2								Rd16+1→Rd16	—	—	↕	↕	↕	—		1
	INC.W #2,Rd	W		2								Rd16+2→Rd16	—	—	↕	↕	↕	—		1
	INC.L #1,ERd	L		2								ERd32+1→ERd32	—	—	↕	↕	↕	—		1
	INC.L #2,ERd	L		2								ERd32+2→ERd32	—	—	↕	↕	↕	—		1
DAA	DAA Rd	B		2								Rd8 10進補正→Rd8	—	*	↕	↕	*	↕		1
SUB	SUB.B Rs,Rd	B		2								Rd8-Rs8→Rd8	—	↕	↕	↕	↕	↕		1
	SUB.W #xx:16,Rd	W	4									Rd16-#xx:16→Rd16	—	[2]	↕	↕	↕	↕		2
	SUB.W Rs,Rd	W		2								Rd16-Rs16→Rd16	—	[2]	↕	↕	↕	↕		1
	SUB.L #xx:32,ERd	L	6									ERd32-#xx:32→ERd32	—	[3]	↕	↕	↕	↕		3
	SUB.L ERs,ERd	L		2								ERd32-ERs32→ERd32	—	[3]	↕	↕	↕	↕		1
SUBX	SUBX #xx:8,Rd	B	2									Rd8-#xx:8-C→Rd8	—	↕	↕	[4]	↕	↕		1
	SUBX Rs,Rd	B		2								Rd8-Rs8-C→Rd8	—	↕	↕	[4]	↕	↕		1
SUBS	SUBS #1,ERd	L		2								ERd32-1→ERd32	—	—	—	—	—	—		1
	SUBS #2,ERd	L		2								ERd32-2→ERd32	—	—	—	—	—	—		1
	SUBS #4,ERd	L		2								ERd32-4→ERd32	—	—	—	—	—	—		1
DEC	DEC.B Rd	B		2								Rd8-1→Rd8	—	—	↕	↕	↕	—		1
	DEC.W #1,Rd	W		2								Rd16-1→Rd16	—	—	↕	↕	↕	—		1
	DEC.W #2,Rd	W		2								Rd16-2→Rd16	—	—	↕	↕	↕	—		1
	DEC.L #1,ERd	L		2								ERd32-1→ERd32	—	—	↕	↕	↕	—		1
	DEC.L #2,ERd	L		2								ERd32-2→ERd32	—	—	↕	↕	↕	—		1
DAS	DAS Rd	B		2								Rd8 10進補正→Rd8	—	*	↕	↕	*	—		1
MULXU	MULXU.B Rs,Rd	B		2								Rd8×Rs8→Rd16（符号なし乗算）	—	—	—	—	—	—		3(12*2)*5*9
	MULXU.W Rs,ERd	W		2								Rd16×Rs16→ERd32（符号なし乗算）	—	—	—	—	—	—		4(20*2)*5*9
MULXS	MULXS.B Rs,Rd	B		4								Rd8×Rs8→Rd16（符号付き乗算）	—	—	↕	↕	—	—		4(13*2)*6*9
	MULXS.W Rs,ERd	W		4								Rd16×Rs16→ERd32（符号付き乗算）	—	—	↕	↕	—	—		5(21*2)*6*9
DIVXU	DIVXU.B Rs,Rd	B		2								Rd16÷Rs8→Rd16 (RdH：余り、RdL：商) (符号なし除算)	—	—	[5]	[6]	—	—		12
	DIVXU.W Rs,ERd	W		2								ERd32÷Rs16→ERd32 (Ed：余り、Rd：商) (符号なし除算)	—	—	[5]	[6]	—	—		20
DIVXS	DIVXS.B Rs,Rd	B		4								Rd16÷Rs8→Rd16 (RdH：余り、RdL：商) (符号付き除算)	—	—	[7]	[6]	—	—		13
	DIVXS.W Rs,ERd	W		4								ERd32÷Rs16→ERd32 (Ed：余り、Rd：商) (符号付き除算)	—	—	[7]	[6]	—	—		21
CMP	CMP.B #xx:8,Rd	B	2									Rd8-#xx:8	—	↕	↕	↕	↕	↕		1
	CMP.B Rs,Rd	B		2								Rd8-Rs8	—	↕	↕	↕	↕	↕		1
	CMP.W #xx:16,Rd	W	4									Rd16-#xx:16	—	[2]	↕	↕	↕	↕		2
	CMP.W Rs,Rd	W		2								Rd16-Rs16	—	[2]	↕	↕	↕	↕		1
	CMP.L #xx:32,ERd	L	6									ERd32-#xx:32	—	[3]	↕	↕	↕	↕		3
	CMP.L ERs,ERd	L		2								ERd32-ERs32	—	[3]	↕	↕	↕	↕		1
NEG	NEG.B Rd	B		2								0-Rd8→Rd8	—	↕	↕	↕	↕	↕		1
	NEG.W Rd	W		2								0-Rd16→Rd16	—	↕	↕	↕	↕	↕		1
	NEG.L ERd	L		2								0-ERd32→ERd32	—	↕	↕	↕	↕	↕		1
EXTU	EXTU.W Rd	W		2								0→(<ビット15～8> of Rd16)	—	—	0	↕	0	—		1
	EXTU.L ERd	L		2								0→(<ビット31～16> of ERd32)	—	—	0	↕	0	—		1
EXTS	EXTS.W Rd	W		2								(<ビット7> of Rd16)→ (<ビット15～8> of Rd16)	—	—	↕	↕	0	—		1
	EXTS.L ERd	L		2								(<ビット15> of ERd32)→ (<ビット31～16> of ERd32)	—	—	↕	↕	0	—		1
TAS	TAS @ERd*8	B			4							@ERd-0→CCRセット, (1)→ (<ビット7> of @ERd)	—	—	↕	↕	0	—		4
MAC*	MAC @ERn+,@ERm+	—					4					@ERnx@ERm+MAC→MAC (符号付き乗算) ERn+2→ERn, ERm+2→ERm	—	—	—	—	—	—		4
CLRMAC*	CLRMAC	—									2	0→MACH, MACL	—	—	—	—	—	—		2*7*9
LDMAC*	LDMAC ERs,MACH	L		2								ERs→MACH	—	—	—	—	—	—		2*7*9
	LDMAC ERs,MACL	L		2								ERs→MACL	—	—	—	—	—	—		2*7*9
STMAC*	STMAC MACH,ERd	L		2								MACH→ERd	—	—	↕	↕	↕	—		1*7*9
	STMAC MACL,ERd	L		2								MACL→ERd	—	—	↕	↕	↕	—		1*7*9

(3) 論理演算命令

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション	コンディションコード						実行ステート数*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	ノーマル	アドバンスト
AND	AND.B #xx:8,Rd	B	2									Rd8∧#xx:8→Rd8	—	—	↕	↕	0	—	1	
	AND.B Rs,Rd	B		2								Rd8∧Rs8→Rd8	—	—	↕	↕	0	—	1	
	AND.W #xx:16,Rd	W	4									Rd16∧#xx:16→Rd16	—	—	↕	↕	0	—	2	
	AND.W Rs,Rd	W		2								Rd16∧Rs16→Rd16	—	—	↕	↕	0	—	1	
	AND.L #xx:32,ERd	L	6									ERd32∧#xx:32→ERd32	—	—	↕	↕	0	—	3	
	AND.L ERs,ERd	L		4								ERd32∧ERs32→ERd32	—	—	↕	↕	0	—	2	
OR	OR.B #xx:8,Rd	B	2									Rd8∨#xx:8→Rd8	—	—	↕	↕	0	—	1	
	OR.B Rs,Rd	B		2								Rd8∨Rs8→Rd8	—	—	↕	↕	0	—	1	
	OR.W #xx:16,Rd	W	4									Rd16∨#xx:16→Rd16	—	—	↕	↕	0	—	2	
	OR.W Rs,Rd	W		2								Rd16∨Rs16→Rd16	—	—	↕	↕	0	—	1	
	OR.L #xx:32,ERd	L	6									ERd32∨#xx:32→ERd32	—	—	↕	↕	0	—	3	
	OR.L ERs,ERd	L		4								ERd32∨ERs32→ERd32	—	—	↕	↕	0	—	2	
XOR	XOR.B #xx:8,Rd	B	2									Rd8⊕#xx:8→Rd8	—	—	↕	↕	0	—	1	
	XOR.B Rs,Rd	B		2								Rd8⊕Rs8→Rd8	—	—	↕	↕	0	—	1	
	XOR.W #xx:16,Rd	W	4									Rd16⊕#xx:16→Rd16	—	—	↕	↕	0	—	2	
	XOR.W Rs,Rd	W		2								Rd16⊕Rs16→Rd16	—	—	↕	↕	0	—	1	
	XOR.L #xx:32,ERd	L	6									ERd32⊕#xx:32→ERd32	—	—	↕	↕	0	—	3	
	XOR.L ERs,ERd	L		4								ERd32⊕ERs32→ERd32	—	—	↕	↕	0	—	2	
NOT	NOT.B Rd	B		2								～Rd8→Rd8	—	—	↕	↕	0	—	1	
	NOT.W Rd	W		2								～Rd16→Rd16	—	—	↕	↕	0	—	1	
	NOT.L ERd	L		2								～Rd32→Rd32	—	—	↕	↕	0	—	1	

(4) シフト命令

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション	コンディションコード						実行ステート数*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	ノーマル	アドバンスト
SHAL	SHAL.B Rd	B		2									—	—	↕	↕	↕	↕	1	
	SHAL.B #2,Rd	B		2									—	—	↕	↕	↕	↕	1	
	SHAL.W Rd	W		2									—	—	↕	↕	↕	↕	1	
	SHAL.W #2,Rd	W		2									—	—	↕	↕	↕	↕	1	
	SHAL.L ERd	L		2									—	—	↕	↕	↕	↕	1	
	SHAL.L #2,ERd	L		2									—	—	↕	↕	↕	↕	1	
SHAR	SHAR.B Rd	B		2									—	—	↕	↕	0	↕	1	
	SHAR.B #2,Rd	B		2									—	—	↕	↕	0	↕	1	
	SHAR.W Rd	W		2									—	—	↕	↕	0	↕	1	
	SHAR.W #2,Rd	W		2									—	—	↕	↕	0	↕	1	
	SHAR.L ERd	L		2									—	—	↕	↕	0	↕	1	
	SHAR.L #2,ERd	L		2									—	—	↕	↕	0	↕	1	
SHLL	SHLL.B Rd	B		2									—	—	↕	↕	0	↕	1	
	SHLL.B #2,Rd	B		2									—	—	↕	↕	0	↕	1	
	SHLL.W Rd	W		2									—	—	↕	↕	0	↕	1	
	SHLL.W #2,Rd	W		2									—	—	↕	↕	0	↕	1	
	SHLL.L ERd	L		2									—	—	↕	↕	0	↕	1	
	SHLL.L #2,ERd	L		2									—	—	↕	↕	0	↕	1	
SHLR	SHLR.B Rd	B		2									—	—	0	↕	0	↕	1	
	SHLR.B #2,Rd	B		2									—	—	0	↕	0	↕	1	
	SHLR.W Rd	W		2									—	—	0	↕	0	↕	1	
	SHLR.W #2,Rd	W		2									—	—	0	↕	0	↕	1	
	SHLR.L ERd	L		2									—	—	0	↕	0	↕	1	
	SHLR.L #2,ERd	L		2									—	—	0	↕	0	↕	1	
ROTXL	ROTXL.B Rd	B		2									—	—	↕	↕	0	↕	1	
	ROTXL.B #2,Rd	B		2									—	—	↕	↕	0	↕	1	
	ROTXL.W Rd	W		2									—	—	↕	↕	0	↕	1	
	ROTXL.W #2,Rd	W		2									—	—	↕	↕	0	↕	1	
	ROTXL.L ERd	L		2									—	—	↕	↕	0	↕	1	
	ROTXL.L #2,ERd	L		2									—	—	↕	↕	0	↕	1	
ROTXR	ROTXR.B Rd	B		2									—	—	↕	↕	0	↕	1	
	ROTXR.B #2,Rd	B		2									—	—	↕	↕	0	↕	1	
	ROTXR.W Rd	W		2									—	—	↕	↕	0	↕	1	
	ROTXR.W #2,Rd	W		2									—	—	↕	↕	0	↕	1	
	ROTXR.L ERd	L		2									—	—	↕	↕	0	↕	1	
	ROTXR.L #2,ERd	L		2									—	—	↕	↕	0	↕	1	
ROTL	ROTL.B Rd	B		2									—	—	↕	↕	0	↕	1	
	ROTL.B #2,Rd	B		2									—	—	↕	↕	0	↕	1	
	ROTL.W Rd	W		2									—	—	↕	↕	0	↕	1	
	ROTL.W #2,Rd	W		2									—	—	↕	↕	0	↕	1	
	ROTL.L ERd	L		2									—	—	↕	↕	0	↕	1	
	ROTL.L #2,ERd	L		2									—	—	↕	↕	0	↕	1	
ROTR	ROTR.B Rd	B		2									—	—	↕	↕	0	↕	1	
	ROTR.B #2,Rd	B		2									—	—	↕	↕	0	↕	1	
	ROTR.W Rd	W		2									—	—	↕	↕	0	↕	1	
	ROTR.W #2,Rd	W		2									—	—	↕	↕	0	↕	1	
	ROTR.L ERd	L		2									—	—	↕	↕	0	↕	1	
	ROTR.L #2,ERd	L		2									—	—	↕	↕	0	↕	1	

(5) ビット操作命令

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション	コンディションコード						実行ステート数*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa	|		I	H	N	Z	V	C	ノーマル	ｱﾄﾞﾊﾞﾝｽﾄ
BSET	BSET #xx:3,Rd	B		2								(#xx:3 of Rd8)←1	—	—	—	—	—	—	1	
	BSET #xx:3,@ERd	B			4							(#xx:3 of @ERd)←1	—	—	—	—	—	—	4	
	BSET #xx:3,@aa:8	B						4				(#xx:3 of @aa:8)←1	—	—	—	—	—	—	4	
	BSET #xx:3,@aa:16	B						6				(#xx:3 of @aa:16)←1	—	—	—	—	—	—	5	
	BSET #xx:3,@aa:32	B						8				(#xx:3 of @aa:32)←1	—	—	—	—	—	—	6	
	BSET Rn,Rd	B		2								(Rn8 of Rd8)←1	—	—	—	—	—	—	1	
	BSET Rn,@ERd	B			4							(Rn8 of @ERd)←1	—	—	—	—	—	—	4	
	BSET Rn,@aa:8	B						4				(Rn8 of @aa:8)←1	—	—	—	—	—	—	4	
	BSET Rn,@aa:16	B						6				(Rn8 of @aa:16)←1	—	—	—	—	—	—	5	
	BSET Rn,@aa:32	B						8				(Rn8 of @aa:32)←1	—	—	—	—	—	—	6	
BCLR	BCLR #xx:3,Rd	B		2								(#xx:3 of Rd8)←0	—	—	—	—	—	—	1	
	BCLR #xx:3,@ERd	B			4							(#xx:3 of @ERd)←0	—	—	—	—	—	—	4	
	BCLR #xx:3,@aa:8	B						4				(#xx:3 of @aa:8)←0	—	—	—	—	—	—	4	
	BCLR #xx:3,@aa:16	B						6				(#xx:3 of @aa:16)←0	—	—	—	—	—	—	5	
	BCLR #xx:3,@aa:32	B						8				(#xx:3 of @aa:32)←0	—	—	—	—	—	—	6	
	BCLR Rn,Rd	B		2								(Rn8 of Rd8)←0	—	—	—	—	—	—	1	
	BCLR Rn,@ERd	B			4							(Rn8 of @ERd)←0	—	—	—	—	—	—	4	
	BCLR Rn,@aa:8	B						4				(Rn8 of @aa:8)←0	—	—	—	—	—	—	4	
	BCLR Rn,@aa:16	B						6				(Rn8 of @aa:16)←0	—	—	—	—	—	—	5	
	BCLR Rn,@aa:32	B						8				(Rn8 of @aa:32)←0	—	—	—	—	—	—	6	
BNOT	BNOT #xx:3,Rd	B		2								(#xx:3 of Rd8)← [～(#xx:3 of Rd8)]	—	—	—	—	—	—	1	
	BNOT #xx:3,@ERd	B			4							(#xx:3 of @ERd)← [～(#xx:3 of @ERd)]	—	—	—	—	—	—	4	
	BNOT #xx:3,@aa:8	B						4				(#xx:3 of @aa:8)← [～(#xx:3 of @aa:8)]	—	—	—	—	—	—	4	
	BNOT #xx:3,@aa:16	B						6				(#xx:3 of @aa:16)←[～(#xx:3 of @aa:16)]	—	—	—	—	—	—	5	
	BNOT #xx:3,@aa:32	B						8				(#xx:3 of @aa:32)← [～(#xx:3 of @aa:32)]	—	—	—	—	—	—	6	
	BNOT Rn,Rd	B		2								(Rn8 of Rd8)← [～(Rn8 of Rd8)]	—	—	—	—	—	—	1	
	BNOT Rn,@ERd	B			4							(Rn8 of @ERd)← [～(Rn8 of @ERd)]	—	—	—	—	—	—	4	
	BNOT Rn,@aa:8	B						4				(Rn8 of @aa:8)← [～(Rn8 of @aa:8)]	—	—	—	—	—	—	4	
	BNOT Rn,@aa:16	B						6				(Rn8 of @aa:16)← [～(Rn8 of @aa:16)]	—	—	—	—	—	—	5	
	BNOT Rn,@aa:32	B						8				(Rn8 of @aa:32)← [～(Rn8 of @aa:32)]	—	—	—	—	—	—	6	
BTST	BTST #xx:3,Rd	B		2								～(#xx:3 of Rd8)→Z	—	—	—	↕	—	—	1	
	BTST #xx:3,@ERd	B			4							～(#xx:3 of @ERd)→Z	—	—	—	↕	—	—	3	
	BTST #xx:3,@aa:8	B						4				～(#xx:3 of @aa:8)→Z	—	—	—	↕	—	—	3	
	BTST #xx:3,@aa:16	B						6				～(#xx:3 of @aa:16)→Z	—	—	—	↕	—	—	4	
	BTST #xx:3,@aa:32	B						8				～(#xx:3 of @aa:32)→Z	—	—	—	↕	—	—	5	
	BTST Rn,Rd	B		2								～(Rn8 of Rd8)→Z	—	—	—	↕	—	—	1	
	BTST Rn,@ERd	B			4							～(Rn8 of @ERd)→Z	—	—	—	↕	—	—	3	
	BTST Rn,@aa:8	B						4				～(Rn8 of @aa:8)→Z	—	—	—	↕	—	—	3	
	BTST Rn,@aa:16	B						6				～(Rn8 of @aa:16)→Z	—	—	—	↕	—	—	4	
	BTST Rn,@aa:32	B						8				～(Rn8 of @aa:32)→Z	—	—	—	↕	—	—	5	
BLD	BLD #xx:3,Rd	B		2								(#xx:3 of Rd8)→C	—	—	—	—	—	↕	1	
	BLD #xx:3,@ERd	B			4							(#xx:3 of @ERd)→C	—	—	—	—	—	↕	3	
	BLD #xx:3,@aa:8	B						4				(#xx:3 of @aa:8)→C	—	—	—	—	—	↕	3	
	BLD #xx:3,@aa:16	B						6				(#xx:3 of @aa:16)→C	—	—	—	—	—	↕	4	
	BLD #xx:3,@aa:32	B						8				(#xx:3 of @aa:32)→C	—	—	—	—	—	↕	5	
BILD	BILD #xx:3,Rd	B		2								～(#xx:3 of Rd8)→C	—	—	—	—	—	↕	1	
	BILD #xx:3,@ERd	B			4							～(#xx:3 of @ERd)→C	—	—	—	—	—	↕	3	
	BILD #xx:3,@aa:8	B						4				～(#xx:3 of @aa:8)→C	—	—	—	—	—	↕	3	
	BILD #xx:3,@aa:16	B						6				～(#xx:3 of @aa:16)→C	—	—	—	—	—	↕	4	
	BILD #xx:3,@aa:32	B						8				～(#xx:3 of @aa:32)→C	—	—	—	—	—	↕	5	
BST	BST #xx:3,Rd	B		2								C→(#xx:3 of Rd8)	—	—	—	—	—	—	1	
	BST #xx:3,@ERd	B			4							C→(#xx:3 of @ERd)	—	—	—	—	—	—	4	
	BST #xx:3,@aa:8	B						4				C→(#xx:3 of @aa:8)	—	—	—	—	—	—	4	
	BST #xx:3,@aa:16	B						6				C→(#xx:3 of @aa:16)	—	—	—	—	—	—	5	
	BST #xx:3,@aa:32	B						8				C→(#xx:3 of @aa:32)	—	—	—	—	—	—	6	
BIST	BIST #xx:3,Rd	B		2								～C→(#xx:3 of Rd8)	—	—	—	—	—	—	1	
	BIST #xx:3,@ERd	B			4							～C→(#xx:3 of @ERd)	—	—	—	—	—	—	4	
	BIST #xx:3,@aa:8	B						4				～C→(#xx:3 of @aa:8)	—	—	—	—	—	—	4	
	BIST #xx:3,@aa:16	B						6				～C→(#xx:3 of @aa:16)	—	—	—	—	—	—	5	
	BIST #xx:3,@aa:32	B						8				～C→(#xx:3 of @aa:32)	—	—	—	—	—	—	6	
BAND	BAND #xx:3,Rd	B		2								C∧(#xx:3 of Rd8)→C	—	—	—	—	—	↕	1	
	BAND #xx:3,@ERd	B			4							C∧(#xx:3 of @ERd)→C	—	—	—	—	—	↕	3	
	BAND #xx:3,@aa:8	B						4				C∧(#xx:3 of @aa:8)→C	—	—	—	—	—	↕	3	
	BAND #xx:3,@aa:16	B						6				C∧(#xx:3 of @aa:16)→C	—	—	—	—	—	↕	4	
	BAND #xx:3,@aa:32	B						8				C∧(#xx:3 of @aa:32)→C	—	—	—	—	—	↕	5	

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション	コンディションコード						実行ステート数*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	ノーマル	アドバンスト
BIAND	BIAND #xx:3,Rd	B		2								C∧ [～(#xx:3 of Rd8)]→C	—	—	—	—	—	↕	1	
	BIAND #xx:3,@ERd	B			4							C∧ [～(#xx:3 of @ERd)]→C	—	—	—	—	—	↕	3	
	BIAND #xx:3,@aa:8	B						4				C∧ [～(#xx:3 of @aa:8)]→C	—	—	—	—	—	↕	3	
	BIAND #xx:3,@aa:16	B						6				C∧ [～(#xx:3 of @aa:16)]→C	—	—	—	—	—	↕	4	
	BIAND #xx:3,@aa:32	B						8				C∧ [～(#xx:3 of @aa:32)]→C	—	—	—	—	—	↕	5	
BOR	BOR #xx:3,Rd	B		2								C∨(#xx:3 of Rd8)→C	—	—	—	—	—	↕	1	
	BOR #xx:3,@ERd	B			4							C∨(#xx:3 of @ERd)→C	—	—	—	—	—	↕	3	
	BOR #xx:3,@aa:8	B						4				C∨(#xx:3 of @aa:8)→C	—	—	—	—	—	↕	3	
	BOR #xx:3,@aa:16	B						6				C∨(#xx:3 of @aa:16)→C	—	—	—	—	—	↕	4	
	BOR #xx:3,@aa:32	B						8				C∨(#xx:3 of @aa:32)→C	—	—	—	—	—	↕	5	
BIOR	BIOR #xx:3,Rd	B		2								C∨ [～(#xx:3 of Rd8)]→C	—	—	—	—	—	↕	1	
	BIOR #xx:3,@ERd	B			4							C∨ [～(#xx:3 of @ERd)]→C	—	—	—	—	—	↕	3	
	BIOR #xx:3,@aa:8	B						4				C∨ [～(#xx:3 of @aa:8)]→C	—	—	—	—	—	↕	3	
	BIOR #xx:3,@aa:16	B						6				C∨ [～(#xx:3 of @aa:16)]→C	—	—	—	—	—	↕	4	
	BIOR #xx:3,@aa:32	B						8				C∨ [～(#xx:3 of @aa:32)]→C	—	—	—	—	—	↕	5	
BXOR	BXOR #xx:3,Rd	B		2								C⊕ (#xx:3 of Rd8)→C	—	—	—	—	—	↕	1	
	BXOR #xx:3,@ERd	B			4							C⊕ (#xx:3 of @ERd)→C	—	—	—	—	—	↕	3	
	BXOR #xx:3,@aa:8	B						4				C⊕ (#xx:3 of @aa:8)→C	—	—	—	—	—	↕	3	
	BXOR #xx:3,@aa:16	B						6				C⊕ (#xx:3 of @aa:16)→C	—	—	—	—	—	↕	4	
	BXOR #xx:3,@aa:32	B						8				C⊕ (#xx:3 of @aa:32)→C	—	—	—	—	—	↕	5	
BIXOR	BIXOR #xx:3,Rd	B		2								C⊕ [～(#xx:3 of Rd8)]→C	—	—	—	—	—	↕	1	
	BIXOR #xx:3,@ERd	B			4							C⊕ [～(#xx:3 of @ERd)]→C	—	—	—	—	—	↕	3	
	BIXOR #xx:3,@aa:8	B						4				C⊕ [～(#xx:3 of @aa:8)]→C	—	—	—	—	—	↕	3	
	BIXOR #xx:3,@aa:16	B						6				C⊕ [～(#xx:3 of @aa:16)]→C	—	—	—	—	—	↕	4	
	BIXOR #xx:3,@aa:32	B						8				C⊕ [～(#xx:3 of @aa:32)]→C	—	—	—	—	—	↕	5	

(6) 分岐命令

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション		コンディションコード						実行ステート数*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa	—		分岐条件	I	H	N	Z	V	C	ノーマル	アドバンスト
Bcc	BRA d:8(BT d:8)	—							2			if condition is true then PC←PC＋d else next;	Always	—	—	—	—	—	—	2	
	BRA d:16(BT d:16)	—							4					—	—	—	—	—	—	3	
	BRN d:8(BF d:8)	—							2				Never	—	—	—	—	—	—	2	
	BRN d:16(BF d:16)	—							4					—	—	—	—	—	—	3	
	BHI d:8	—							2				C∨Z=0	—	—	—	—	—	—	2	
	BHI d:16	—							4					—	—	—	—	—	—	3	
	BLS d:8	—							2				C∨Z=1	—	—	—	—	—	—	2	
	BLS d:16	—							4					—	—	—	—	—	—	3	
	BCC d:8(BHS d:8)	—							2				C=0	—	—	—	—	—	—	2	
	BCC d:16(BHS d:16)	—							4					—	—	—	—	—	—	3	
	BCS d:8(BLO d:8)	—							2				C=1	—	—	—	—	—	—	2	
	BCS d:16(BLO d:16)	—							4					—	—	—	—	—	—	3	
	BNE d:8	—							2				Z=0	—	—	—	—	—	—	2	
	BNE d:16	—							4					—	—	—	—	—	—	3	
	BEQ d:8	—							2				Z=1	—	—	—	—	—	—	2	
	BEQ d:16	—							4					—	—	—	—	—	—	3	
	BVC d:8	—							2				V=0	—	—	—	—	—	—	2	
	BVC d:16	—							4					—	—	—	—	—	—	3	
	BVS d:8	—							2				V=1	—	—	—	—	—	—	2	
	BVS d:16	—							4					—	—	—	—	—	—	3	
	BPL d:8	—							2				N=0	—	—	—	—	—	—	2	
	BPL d:16	—							4					—	—	—	—	—	—	3	
	BMI d:8	—							2				N=1	—	—	—	—	—	—	2	
	BMI d:16	—							4					—	—	—	—	—	—	3	
	BGE d:8	—							2				N⊕V=0	—	—	—	—	—	—	2	
	BGE d:16	—							4					—	—	—	—	—	—	3	
	BLT d:8	—							2				N⊕V=1	—	—	—	—	—	—	2	
	BLT d:16	—							4					—	—	—	—	—	—	3	
	BGT d:8	—							2				Z∨(N⊕V)=0	—	—	—	—	—	—	2	
	BGT d:16	—							4					—	—	—	—	—	—	3	
	BLE d:8	—							2				Z∨(N⊕V)=1	—	—	—	—	—	—	2	
	BLE d:16	—							4					—	—	—	—	—	—	3	
JMP	JMP @ERn	—			2							PC←ERn		—	—	—	—	—	—	2	
	JMP @aa:24	—						4				PC←aa:24		—	—	—	—	—	—	3	
	JMP @@aa:8	—								2		PC←@aa:8		—	—	—	—	—	—	4	5
BSR	BSR d:8	—							2			PC→@-SP,PC←PC+d:8		—	—	—	—	—	—	3	4
	BSR d:16	—							4			PC→@-SP,PC←PC+d:16		—	—	—	—	—	—	4	5
JSR	JSR @ERn	—			2							PC→@-SP,PC←ERn		—	—	—	—	—	—	3	4
	JSR @aa:24	—						4				PC→@-SP,PC←aa:24		—	—	—	—	—	—	4	5
	JSR @@aa:8	—								2		PC→@-SP,PC←@aa:8		—	—	—	—	—	—	4	6
RTS	RTS	—									2	PC←@SP+		—	—	—	—	—	—	4	5

(7) システム制御命令

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション	コンディションコード						実行ステート数※1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa	|		I	H	N	Z	V	C	ノーマル	アドバンスト
TRAPA	TRAPA #xx:2	—									2	PC→@-SP,CCR→@-SP, EXR→@-SP,<ベクタ>→PC	1	—	—	—	—	—	7 [9※2]	8 [9※2]
RTE	RTE	—										EXR←@SP+,CCR←@SP+, PC←@SP+	↕	↕	↕	↕	↕	↕	5 [9※2]	
SLEEP	SLEEP	—										低消費電力状態に遷移	—	—	—	—	—	—	2	
LDC	LDC #xx:8,CCR	B	2									#xx:8→CCR	↕	↕	↕	↕	↕	↕	1	
	LDC #xx:8,EXR	B	4									#xx:8→EXR	—	—	—	—	—	—	2	
	LDC Rs,CCR	B		2								Rs8→CCR	↕	↕	↕	↕	↕	↕	1	
	LDC Rs,EXR	B		2								Rs8→EXR	—	—	—	—	—	—	1	
	LDC @ERs,CCR	W			4							@ERs→CCR	↕	↕	↕	↕	↕	↕	3	
	LDC @ERs,EXR	W			4							@ERs→EXR	—	—	—	—	—	—	3	
	LDC @(d:16,ERs),CCR	W				6						@(d:16,ERs)→CCR	↕	↕	↕	↕	↕	↕	4	
	LDC @(d:16,ERs),EXR	W				6						@(d:16,ERs)→EXR	—	—	—	—	—	—	4	
	LDC @(d:32,ERs),CCR	W				10						@(d:32,ERs)→CCR	↕	↕	↕	↕	↕	↕	6	
	LDC @(d:32,ERs),EXR	W				10						@(d:32,ERs)→EXR	—	—	—	—	—	—	6	
	LDC @ERs+,CCR	W					4					@ERs→CCR,ERs32+2→ERs32	↕	↕	↕	↕	↕	↕	4	
	LDC @ERs+,EXR	W					4					@ERs→EXR,ERs32+2→ERs32	—	—	—	—	—	—	4	
	LDC @aa:16,CCR	W						6				@aa:16→CCR	↕	↕	↕	↕	↕	↕	4	
	LDC @aa:16,EXR	W						6				@aa:16→EXR	—	—	—	—	—	—	4	
	LDC @aa:32,CCR	W						8				@aa:32→CCR	↕	↕	↕	↕	↕	↕	5	
	LDC @aa:32,EXR	W						8				@aa:32→EXR	—	—	—	—	—	—	5	
STC	STC CCR,Rd	B		2								CCR→Rd8	—	—	—	—	—	—	1	
	STC EXR,Rd	B		2								EXR→Rd8	—	—	—	—	—	—	1	
	STC CCR,@ERd	W			4							CCR→@ERd	—	—	—	—	—	—	3	
	STC EXR,@ERd	W			4							EXR→@ERd	—	—	—	—	—	—	3	
	STC CCR,@(d:16,ERd)	W				6						CCR→@(d:16,ERd)	—	—	—	—	—	—	4	
	STC EXR,@(d:16,ERd)	W				6						EXR→@(d:16,ERd)	—	—	—	—	—	—	4	
	STC CCR,@(d:32,ERd)	W				10						CCR→@(d:32,ERd)	—	—	—	—	—	—	6	
	STC EXR,@(d:32,ERd)	W				10						EXR→@(d:32,ERd)	—	—	—	—	—	—	6	
	STC CCR,@-ERd	W					4					ERd32-2→ERd32,CCR→@ERd	—	—	—	—	—	—	4	
	STC EXR,@-ERd	W					4					ERd32-2→ERd32,EXR→@ERd	—	—	—	—	—	—	4	
	STC CCR,@aa:16	W						6				CCR→@aa:16	—	—	—	—	—	—	4	
	STC EXR,@aa:16	W						6				EXR→@aa:16	—	—	—	—	—	—	4	
	STC CCR,@aa:32	W						8				CCR→@aa:32	—	—	—	—	—	—	5	
	STC EXR,@aa:32	W						8				EXR→@aa:32	—	—	—	—	—	—	5	
ANDC	ANDC #xx:8,CCR	B	2									CCR∧#xx:8→CCR	↕	↕	↕	↕	↕	↕	1	
	ANDC #xx:8,EXR	B	4									EXR∧#xx:8→EXR	—	—	—	—	—	—	2	
ORC	ORC #xx:8,CCR	B	2									CCR∨#xx:8→CCR	↕	↕	↕	↕	↕	↕	1	
	ORC #xx:8,EXR	B	4									EXR∨#xx:8→EXR	—	—	—	—	—	—	2	
XORC	XORC #xx:8,CCR	B	2									CCR⊕#xx:8→CCR	↕	↕	↕	↕	↕	↕	1	
	XORC #xx:8,EXR	B	4									EXR⊕#xx:8→EXR	—	—	—	—	—	—	2	
NOP	NOP	—									2	PC←PC+2	—	—	—	—	—	—	1	

(8) ブロック転送命令

ニーモニック		サイズ	アドレッシングモード／命令長（バイト）									オペレーション	コンディションコード						実行ステート数*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa			I	H	N	Z	V	C	ノーマル	アドバンスト
EEPMOV	EEPMOV.B	—									4	if R4L≠0 Repeat @ER5→@ER6 ER5+1→ER5 ER6+1→ER6 R4L-1→R4L Until R4L=0 else next;	—	—	—	—	—	—	4+2n*3	
	EEPMOV.W	—									4	if R4≠0 Repeat @ER5→@ER6 ER5+1→ER5 ER6+1→ER6 R4-1→R4 Until R4=0 else next;	—	—	—	—	—	—	4+2n*3	

【注】 * H8S/2600 CPUでのみサポートしています。
*1 実行ステート数は、命令コードおよびオペランドが内蔵メモリに存在する場合の値です。
*2 （ ）内はH8S/2000 CPUの値です。［ ］内は割り込み制御モード2、3の値です。
*3 nはR4LまたはR4の初期設定値です。
*4 復帰／退避レジスタ数が2本のとき7ステート、3本のとき9ステート、4本のとき11ステートになります。
*5 MULXU、MULXS、STMAC命令の直後は1ステート多くなります。また、MAC命令実行後3ステート以内にMULXU命令を実行しようとした場合、最大で3ステート多くなります。例えば、MAC命令とMULXU命令の間に1ステート命令（NOP等）が1つある場合、MULXU命令は2ステート多くなります。
*6 MAC命令実行後2ステート以内にMULXS命令を実行しようとした場合、最大で2ステート多くなります。例えば、MAC命令とMULXS命令の間に1ステート命令（NOP等）が1つある場合、MULXS命令は1ステート多くなります。
*7 MAC命令実行後3ステート以内にこれらの命令を実行しようとした場合、最大で3ステート多くなります。例えば、MAC命令とこれらの命令の間に1ステート命令（NOP等）が1つある場合、これらの命令は2ステート多くなります。
*8 TAS命令を使用する場合は、レジスタER0、ER1、ER4、ER5を使用してください。
*9 製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。
［1］ Eクロック同期転送命令の実行ステート数は一定ではありません。
［2］ ビット11から桁上がりまたはビット11へ桁下がりが発生したとき1にセットされ、それ以外のとき0にクリアされます。
［3］ ビット27から桁上がりまたはビット27へ桁下がりが発生したとき1にセットされ、それ以外のとき0にクリアされます。
［4］ 演算結果が0（ゼロ）のとき、演算前の値を保持し、それ以外のとき0にクリアされます。
［5］ 除数が負のとき1にセットされ、それ以外のとき0にクリアされます。
［6］ 除数が0（ゼロ）のとき1にセットされ、それ以外のとき0にクリアされます。
［7］ 商が負のとき1にセットされ、それ以外のとき0にクリアされます。
［8］ STMAC命令を実行することで、MAC命令の結果がフラグに反映されます。
［9］ EXRが有効のとき、実行ステート数は1ステート多くなります。

2.4 命令コード一覧

表2.2 命令コード一覧

命令	ニーモニック	サイズ	インストラクションフォーマット									
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト
ADD	ADD.B #xx:8,Rd	B	8 rd	IMM								
	ADD.B Rs,Rd	B	0 8	rs rd								
	ADD.W #xx:16,Rd	W	7 9	1 rd	IMM							
	ADD.W Rs,Rd	W	0 9	rs rd								
	ADD.L #xx:32,ERd	L	7 A	1 0 erd			IMM					
	ADD.L ERs,ERd	L	0 A	1 ers 0 erd								
ADDS	ADDS #1,ERd	L	0 B	0 0 erd								
	ADDS #2,ERd	L	0 B	8 0 erd								
	ADDS #4,ERd	L	0 B	9 0 erd								
ADDX	ADDX #xx:8,Rd	B	9 rd	IMM								
	ADDX Rs,Rd	B	0 E	rs rd								
AND	AND.B #xx:8,Rd	B	E rd	IMM								
	AND.B Rs,Rd	B	1 6	rs rd								
	AND.W #xx:16,Rd	W	7 9	6 rd	IMM							
	AND.W Rs,Rd	W	6 6	rs rd								
	AND.L #xx:32,ERd	L	7 A	6 0 erd			IMM					
	AND.L ERs,ERd	L	0 1	F 0	6 6	0 ers 0 erd						
ANDC	ANDC #xx:8,CCR	B	0 6	IMM								
	ANDC #xx:8,EXR	B	0 1	4 1	0 6	IMM						
BAND	BAND #xx:3,Rd	B	7 6	0 IMM rd								
	BAND #xx:3,@ERd	B	7 C	0 erd 0	7 6	0 IMM 0						
	BAND #xx:3,@aa:8	B	7 E	abs	7 6	0 IMM 0						
	BAND #xx:3,@aa:16	B	6 A	1 0	abs		7 6	0 IMM 0				
	BAND #xx:3,@aa:32	B	6 A	3 0	abs				7 6	0 IMM 0		
Bcc	BRA d:8 (BT d:8)	—	4 0	disp								
	BRA d:16 (BT d:16)	—	5 8	0 0	disp							
	BRN d:8 (BF d:8)	—	4 1	disp								
	BRN d:16 (BF d:16)	—	5 8	1 0	disp							
	BHI d:8	—	4 2	disp								
	BHI d:16	—	5 8	2 0	disp							
	BLS d:8	—	4 3	disp								
	BLS d:16	—	5 8	3 0	disp							
	BCC d:8 (BHS d:8)	—	4 4	disp								
	BCC d:16 (BHS d:16)	—	5 8	4 0	disp							
	BCS d:8 (BLO d:8)	—	4 5	disp								
	BCS d:16 (BLO d:16)	—	5 8	5 0	disp							
	BNE d:8	—	4 6	disp								
	BNE d:16	—	5 8	6 0	disp							
	BEQ d:8	—	4 7	disp								
	BEQ d:16	—	5 8	7 0	disp							
	BVC d:8	—	4 8	disp								
	BVC d:16	—	5 8	8 0	disp							
	BVS d:8	—	4 9	disp								
	BVS d:16	—	5 8	9 0	disp							

命令	ニーモニック	サイズ	インストラクションフォーマット																	
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		第9バイト	第10バイト
Bcc（続き）	BPL d:8	—	4	A	disp															
	BPL d:16	—	5	8	A	0	disp													
	BMI d:8	—	4	B	disp															
	BMI d:16	—	5	8	B	0	disp													
	BGE d:8	—	4	C	disp															
	BGE d:16	—	5	8	C	0	disp													
	BLT d:8	—	4	D	disp															
	BLT d:16	—	5	8	D	0	disp													
	BGT d:8	—	4	E	disp															
	BGT d:16	—	5	8	E	0	disp													
	BLE d:8	—	4	F	disp															
	BLE d:16	—	5	8	F	0	disp													
BCLR	BCLR #xx:3,Rd	B	7	2	0	IMM	rd													
	BCLR #xx:3,@ERd	B	7	D	0	erd	0	7	2	0	IMM	0								
	BCLR #xx:3,@aa:8	B	7	F	abs		7	2	0	IMM	0									
	BCLR #xx:3,@aa:16	B	6	A	1	8	abs				7	2	0	IMM	0					
	BCLR #xx:3,@aa:32	B	6	A	3	8	abs								7	2	0	IMM	0	
	BCLR Rn,Rd	B	6	2	rn	rd														
	BCLR Rn,@ERd	B	7	D	0	erd	0	6	2	rn	0									
	BCLR Rn,@aa:8	B	7	F	abs		6	2	rn	0										
	BCLR Rn,@aa:16	B	6	A	1	8	abs				6	2	rn	0						
	BCLR Rn,@aa:32	B	6	A	3	8	abs								6	2	rn	0		
BIAND	BIAND #xx:3,Rd	B	7	6	1	IMM	rd													
	BIAND #xx:3,@ERd	B	7	C	0	erd	0	7	6	1	IMM	0								
	BIAND #xx:3,@aa:8	B	7	E	abs		7	6	1	IMM	0									
	BIAND #xx:3,@aa:16	B	6	A	1	0	abs				7	6	1	IMM	0					
	BIAND #xx:3,@aa:32	B	6	A	3	0	abs								7	6	1	IMM	0	
BILD	BILD #xx:3,Rd	B	7	7	1	IMM	rd													
	BILD #xx:3,@ERd	B	7	C	0	erd	0	7	7	1	IMM	0								
	BILD #xx:3,@aa:8	B	7	E	abs		7	7	1	IMM	0									
	BILD #xx:3,@aa:16	B	6	A	1	0	abs				7	7	1	IMM	0					
	BILD #xx:3,@aa:32	B	6	A	3	0	abs								7	7	1	IMM	0	
BIOR	BIOR #xx:3,Rd	B	7	4	1	IMM	rd													
	BIOR #xx:3,@ERd	B	7	C	0	erd	0	7	4	1	IMM	0								
	BIOR #xx:3,@aa:8	B	7	E	abs		7	4	1	IMM	0									
	BIOR #xx:3,@aa:16	B	6	A	1	0	abs				7	4	1	IMM	0					
	BIOR #xx:3,@aa:32	B	6	A	3	0	abs								7	4	1	IMM	0	
BIST	BIST #xx:3,Rd	B	6	7	1	IMM	rd													
	BIST #xx:3,@ERd	B	7	D	0	erd	0	6	7	1	IMM	0								
	BIST #xx:3,@aa:8	B	7	F	abs		6	7	1	IMM	0									
	BIST #xx:3,@aa:16	B	6	A	1	8	abs				6	7	1	IMM	0					
	BIST #xx:3,@aa:32	B	6	A	3	8	abs								6	7	1	IMM	0	

命令	ニーモニック	サイズ	インストラクションフォーマット																			
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		第9バイト		第10バイト	
BIXOR	BIXOR #xx:3,Rd	B	7	5	1	IMM	rd															
	BIXOR #xx:3,@ERd	B	7	C	0	erd	0	7	5	1	IMM	0										
	BIXOR #xx:3,@aa:8	B	7	E	abs			7	5	1	IMM	0										
	BIXOR #xx:3,@aa:16	B	6	A	1		0	abs				7	5	1	IMM	0						
	BIXOR #xx:3,@aa:32	B	6	A	3		0	abs									7	5	1	IMM	0	
BLD	BLD #xx:3,Rd	B	7	7	0	IMM	rd															
	BLD #xx:3,@ERd	B	7	C	0	erd	0	7	7	0	IMM	0										
	BLD #xx:3,@aa:8	B	7	E	abs			7	7	0	IMM	0										
	BLD #xx:3,@aa:16	B	6	A	1		0	abs				7	7	0	IMM	0						
	BLD #xx:3,@aa:32	B	6	A	3		0	abs									7	7	0	IMM	0	
BNOT	BNOT #xx:3,Rd	B	7	1	0	IMM	rd															
	BNOT #xx:3,@ERd	B	7	D	0	erd	0	7	1	0	IMM	0										
	BNOT #xx:3,@aa:8	B	7	F	abs			7	1	0	IMM	0										
	BNOT #xx:3,@aa:16	B	6	A	1		8	abs				7	1	0	IMM	0						
	BNOT #xx:3,@aa:32	B	6	A	3		8	abs									7	1	0	IMM	0	
	BNOT Rn,Rd	B	6	1	rn		rd															
	BNOT Rn,@ERd	B	7	D	0	erd	0	6	1	rn		0										
	BNOT Rn,@aa:8	B	7	F	abs			6	1	rn		0										
	BNOT Rn,@aa:16	B	6	A	1		8	abs				6	1	rn		0						
	BNOT Rn,@aa:32	B	6	A	3		8	abs									6	1	rn		0	
BOR	BOR #xx:3,Rd	B	7	4	0	IMM	rd															
	BOR #xx:3,@ERd	B	7	C	0	erd	0	7	4	0	IMM	0										
	BOR #xx:3,@aa:8	B	7	E	abs			7	4	0	IMM	0										
	BOR #xx:3,@aa:16	B	6	A	1		0	abs				7	4	0	IMM	0						
	BOR #xx:3,@aa:32	B	6	A	3		0	abs									7	4	0	IMM	0	
BSET	BSET #xx:3,Rd	B	7	0	0	IMM	rd															
	BSET #xx:3,@ERd	B	7	D	0	erd	0	7	0	0	IMM	0										
	BSET #xx:3,@aa:8	B	7	F	abs			7	0	0	IMM	0										
	BSET #xx:3,@aa:16	B	6	A	1		8	abs				7	0	0	IMM	0						
	BSET #xx:3,@aa:32	B	6	A	3		8	abs									7	0	0	IMM	0	
	BSET Rn,Rd	B	6	0	rn		rd															
	BSET Rn,@ERd	B	7	D	0	erd	0	6	0	rn		0										
	BSET Rn,@aa:8	B	7	F	abs			6	0	rn		0										
	BSET Rn,@aa:16	B	6	A	1		8	abs				6	0	rn		0						
	BSET Rn,@aa:32	B	6	A	3		8	abs									6	0	rn		0	
BSR	BSR d:8	–	5	5	disp																	
	BSR d:16	–	5	C	0		0	disp														
BST	BST #xx:3,Rd	B	6	7	0	IMM	rd															
	BST #xx:3,@ERd	B	7	D	0	erd	0	6	7	0	IMM	0										
	BST #xx:3,@aa:8	B	7	F	abs			6	7	0	IMM	0										
	BST #xx:3,@aa:16	B	6	A	1		8	abs				6	7	0	IMM	0						
	BST #xx:3,@aa:32	B	6	A	3		8	abs									6	7	0	IMM	0	

命令	ニーモニック	サイズ	インストラクションフォーマット																	
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		第9バイト	第10バイト
BTST	BTST #xx:3,Rd	B	7	3	0 IMM	rd														
	BTST #xx:3,@ERd	B	7	C	0 erd	0	7	3	0 IMM	0										
	BTST #xx:3,@aa:8	B	7	E	abs		7	3	0 IMM	0										
	BTST #xx:3,@aa:16	B	6	A	1	0	abs				7	3	0 IMM	0						
	BTST #xx:3,@aa:32	B	6	A	3	0	abs								7	3	0 IMM	0		
	BTST Rn,Rd	B	6	3	rn	rd														
	BTST Rn,@ERd	B	7	C	0 erd	0	6	3	rn	0										
	BTST Rn,@aa:8	B	7	E	abs		6	3	rn	0										
	BTST Rn,@aa:16	B	6	A	1	0	abs				6	3	rn	0						
	BTST Rn,@aa:32	B	6	A	3	0	abs								6	3	rn	0		
BXOR	BXOR #xx:3,Rd	B	7	5	0 IMM	rd														
	BXOR #xx:3,@ERd	B	7	C	0 erd	0	7	5	0 IMM	0										
	BXOR #xx:3,@aa:8	B	7	E	abs		7	5	0 IMM	0										
	BXOR #xx:3,@aa:16	B	6	A	1	0	abs				7	5	0 IMM	0						
	BXOR #xx:3,@aa:32	B	6	A	3	0	abs								7	5	0 IMM	0		
CLRMAC*1	CLRMAC	—	0	1	A	0														
CMP	CMP.B #xx:8,Rd	B	A	rd	IMM															
	CMP.B Rs,Rd	B	1	C	rs	rd														
	CMP.W #xx:16,Rd	W	7	9	2	rd	IMM													
	CMP.W Rs,Rd	W	1	D	rs	rd														
	CMP.L #xx:32,ERd	L	7	A	2	0 erd	IMM													
	CMP.L ERs,ERd	L	1	F	1 ers	0 erd														
DAA	DAA Rd	B	0	F	0	rd														
DAS	DAS Rd	B	1	F	0	rd														
DEC	DEC.B Rd	B	1	A	0	rd														
	DEC.W #1,Rd	W	1	B	5	rd														
	DEC.W #2,Rd	W	1	B	D	rd														
	DEC.L #1,ERd	L	1	B	7	0 erd														
	DEC.L #2,ERd	L	1	B	F	0 erd														
DIVXS	DIVXS.B Rs,Rd	B	0	1	D	0	5	1	rs	rd										
	DIVXS.W Rs,ERd	W	0	1	D	0	5	3	rs	0 erd										
DIVXU	DIVXU.B Rs,Rd	B	5	1	rs	rd														
	DIVXU.W Rs,ERd	W	5	3	rs	0 erd														
EEPMOV	EEPMOV.B	—	7	B	5	C	5	9	8	F										
	EEPMOV.W	—	7	B	D	4	5	9	8	F										
EXTS	EXTS.W Rd	W	1	7	D	rd														
	EXTS.L ERd	L	1	7	F	0 erd														
EXTU	EXTU.W Rd	W	1	7	5	rd														
	EXTU.L ERd	L	1	7	7	0 erd														

命令	ニーモニック	サイズ	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト
INC	INC.B Rd	B	0 A	0 rd								
	INC.W #1,Rd	W	0 B	5 rd								
	INC.W #2,Rd	W	0 B	D rd								
	INC.L #1,ERd	L	0 B	7 0 erd								
	INC.L #2,ERd	L	0 B	F 0 erd								
JMP	JMP @ERn	–	5 9	0 ern 0								
	JMP @aa:24	–	5 A	abs								
	JMP @@aa:8	–	5 B	abs								
JSR	JSR @ERn	–	5 D	0 ern 0								
	JSR @aa:24	–	5 E	abs								
	JSR @@aa:8	–	5 F	abs								
LDC	LDC #xx:8,CCR	B	0 7	IMM								
	LDC #xx:8,EXR	B	0 1	4 1	0 7	IMM						
	LDC Rs,CCR	B	0 3	0 rs								
	LDC Rs,EXR	B	0 3	1 rs								
	LDC @ERs,CCR	W	0 1	4 0	6 9	0 ers 0						
	LDC @ERs,EXR	W	0 1	4 1	6 9	0 ers 0						
	LDC @(d:16,ERs),CCR	W	0 1	4 0	6 F	0 ers 0	disp					
	LDC @(d:16,ERs),EXR	W	0 1	4 1	6 F	0 ers 0	disp					
	LDC @(d:32,ERs),CCR	W	0 1	4 0	7 8	0 ers 0	6 B	2 0	disp			
	LDC @(d:32,ERs),EXR	W	0 1	4 1	7 8	0 ers 0	6 B	2 0	disp			
	LDC @ERs+,CCR	W	0 1	4 0	6 D	0 ers 0						
	LDC @ERs+,EXR	W	0 1	4 1	6 D	0 ers 0						
	LDC @aa:16,CCR	W	0 1	4 0	6 B	0 0	abs					
	LDC @aa:16,EXR	W	0 1	4 1	6 B	0 0	abs					
	LDC @aa:32,CCR	W	0 1	4 0	6 B	2 0	abs					
	LDC @aa:32,EXR	W	0 1	4 1	6 B	2 0	abs					
LDM	LDM.L @SP+, (ERn-ERn+1)	L	0 1	1 0	6 D	7 0 ern+1						
	LDM.L @SP+, (ERn-ERn+2)	L	0 1	2 0	6 D	7 0 ern+2						
	LDM.L @SP+, (ERn-ERn+3)	L	0 1	3 0	6 D	7 0 ern+3						
LDMAC*[1]	LDMAC ERs,MACH	L	0 3	2 0 ers								
	LDMAC ERs,MACL	L	0 3	3 0 ers								
MAC*[1]	MAC @ERn+,@ERm+	–	0 1	6 0	6 D	0 ern 0 erm						
MOV	MOV.B #xx:8,Rd	B	F rd	IMM								
	MOV.B Rs,Rd	B	0 C	rs rd								
	MOV.B @ERs,Rd	B	6 8	0 ers rd								
	MOV.B @(d:16,ERs),Rd	B	6 E	0 ers rd	disp							
	MOV.B @(d:32,ERs),Rd	B	7 8	0 ers 0	6 A	2 rd	disp					
	MOV.B @ERs+,Rd	B	6 C	0 ers rd								
	MOV.B @aa:8,Rd	B	2 rd	abs								
	MOV.B @aa:16,Rd	B	6 A	0 rd	abs							
	MOV.B @aa:32,Rd	B	6 A	2 rd	abs							
	MOV.B Rs,@ERd	B	6 8	1 erd rs								
	MOV.B Rs,@(d:16,ERd)	B	6 E	1 erd rs	disp							
	MOV.B Rs,@(d:32,ERd)	B	7 8	0 erd 0	6 A	A rs	disp					

命令	ニーモニック	サイズ	インストラクションフォーマット																	
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト		第9バイト	第10バイト
MOV（続き）	MOV.B Rs,@-ERd	B	6	C	1 erd	rs														
	MOV.B Rs,@aa:8	B	3	rs	abs															
	MOV.B Rs,@aa :16	B	6	A	8	rs	abs													
	MOV.B Rs,@aa:32	B	6	A	A	rs	abs													
	MOV.W #xx:16,Rd	W	7	9	0	rd	IMM													
	MOV.W Rs,Rd	W	0	D	rs	rd														
	MOV.W @ERs,Rd	W	6	9	0 ers	rd														
	MOV.W @(d:16,ERs),Rd	W	6	F	0 ers	rd	disp													
	MOV.W @(d:32,ERs),Rd	W	7	8	0 ers	0	6	B	2	rd	disp									
	MOV.W @ERs+,Rd	W	6	D	0 ers	rd														
	MOV.W @aa:16,Rd	W	6	B	0	rd	abs													
	MOV.W @aa:32,Rd	W	6	B	2	rd	abs													
	MOV.W Rs,@ERd	W	6	9	1 erd	rs														
	MOV.W Rs,@(d:16,ERd)	W	6	F	1 erd	rs	disp													
	MOV.W Rs,@(d:32,ERd)	W	7	8	0 erd	0	6	B	A	rs	disp									
	MOV.W Rs,@-ERd	W	6	D	1 erd	rs														
	MOV.W Rs,@aa:16	W	6	B	8	rs	abs													
	MOV.W Rs,@aa:32	W	6	B	A	rs	abs													
	MOV.L #xx:32,Rd	L	7	A	0	0 erd	IMM													
	MOV.L ERs,ERd	L	0	F	1 ers	0 erd														
	MOV.L @ERs,ERd	L	0	1	0	0	6	9	0 ers	0 erd										
	MOV.L @(d:16,ERs) ,ERd	L	0	1	0	0	6	F	0 ers	0 erd	disp									
	MOV.L @(d:32,ERs) ,ERd	L	0	1	0	0	7	8	0 ers	0	6	B	2	0 erd	disp					
	MOV.L @ERs+,ERd	L	0	1	0	0	6	D	0 ers	0 erd										
	MOV.L @aa:16 ,ERd	L	0	1	0	0	6	B	0	0 erd	abs									
	MOV.L @aa:32 ,ERd	L	0	1	0	0	6	B	2	0 erd	abs									
	MOV.L ERs,@ERd	L	0	1	0	0	6	9	1 erd	0 ers										
	MOV.L ERs,@(d:16 ,ERd)	L	0	1	0	0	6	F	1 erd	0 ers	disp									
	MOV.L ERs,@(d:32 ,ERd)*2	L	0	1	0	0	7	8	0 erd	0	6	B	A	0 ers	disp					
	MOV.L ERs,@-ERd	L	0	1	0	0	6	D	1 erd	0 ers										
	MOV.L ERs,@aa:16	L	0	1	0	0	6	B	8	0 ers	abs									
	MOV.L ERs,@aa:32	L	0	1	0	0	6	B	A	0 ers	abs									
MOVFPE	MOVFPE @aa:16,Rd	B	6	A	4	rd	abs													
MOVTPE	MOVTPE Rs,@aa:16	B	6	A	C	rs	abs													
MULXS	MULXS.B Rs,Rd	B	0	1	C	0	5	0	rs	rd										
	MULXS.W Rs,ERd	W	0	1	C	0	5	2	rs	0 erd										
MULXU	MULXU.B Rs,Rd	B	5	0	rs	rd														
	MULXU.W Rs,ERd	W	5	2	rs	0 erd														
NEG	NEG.B Rd	B	1	7	8	rd														
	NEG.W Rd	W	1	7	9	rd														
	NEG.L ERd	L	1	7	B	0 erd														
NOP	NOP	−	0	0	0	0														

命令	ニーモニック	サイズ	インストラクションフォーマット										
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト
NOT	NOT.B Rd	B	1 7	0 rd								
	NOT.W Rd	W	1 7	1 rd								
	NOT.L ERd	L	1 7	3 0 erd								
OR	OR.B #xx:8,Rd	B	C rd	IMM								
	OR.B Rs,Rd	B	1 4	rs rd								
	OR.W #xx:16,Rd	W	7 9	4 rd	IMM							
	OR.W Rs,Rd	W	6 4	rs rd								
	OR.L #xx:32,ERd	L	7 A	4 0 erd	IMM							
	OR.L ERs,ERd	L	0 1	F 0	6 4	0 ers 0 erd						
ORC	ORC #xx:8,CCR	B	0 4	IMM								
	ORC #xx:8,EXR	B	0 1	4 1	0 4	IMM						
POP	POP.W Rn	W	6 D	7 rn								
	POP.L ERn	L	0 1	0 0	6 D	7 0 ern						
PUSH	PUSH.W Rn	W	6 D	F rn								
	PUSH.L ERn	L	0 1	0 0	6 D	F 0 ern						
ROTL	ROTL.B Rd	B	1 2	8 rd								
	ROTL.B #2, Rd	B	1 2	C rd								
	ROTL.W Rd	W	1 2	9 rd								
	ROTL.W #2, Rd	W	1 2	D rd								
	ROTL.L ERd	L	1 2	B 0 erd								
	ROTL.L #2, ERd	L	1 2	F 0 erd								
ROTR	ROTR.B Rd	B	1 3	8 rd								
	ROTR.B #2, Rd	B	1 3	C rd								
	ROTR.W Rd	W	1 3	9 rd								
	ROTR.W #2, Rd	W	1 3	D rd								
	ROTR.L ERd	L	1 3	B 0 erd								
	ROTR.L #2, ERd	L	1 3	F 0 erd								
ROTXL	ROTXL.B Rd	B	1 2	0 rd								
	ROTXL.B #2, Rd	B	1 2	4 rd								
	ROTXL.W Rd	W	1 2	1 rd								
	ROTXL.W #2, Rd	W	1 2	5 rd								
	ROTXL.L ERd	L	1 2	3 0 erd								
	ROTXL.L #2, ERd	L	1 2	7 0 erd								
ROTXR	ROTXR.B Rd	B	1 3	0 rd								
	ROTXR.B #2, Rd	B	1 3	4 rd								
	ROTXR.W Rd	W	1 3	1 rd								
	ROTXR.W #2, Rd	W	1 3	5 rd								
	ROTXR.L ERd	L	1 3	3 0 erd								
	ROTXR.L #2, ERd	L	1 3	7 0 erd								
RTE	RTE	–	5 6	7 0								
RTS	RTS	–	5 4	7 0								

命令	ニーモニック	サイズ	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト
SHAL	SHAL.B Rd	B	1 0	8 rd								
	SHAL.B #2, Rd	B	1 0	C rd								
	SHAL.W Rd	W	1 0	9 rd								
	SHAL.W #2, Rd	W	1 0	D rd								
	SHAL.L ERd	L	1 0	B 0 erd								
	SHAL.L #2, ERd	L	1 0	F 0 erd								
SHAR	SHAR.B Rd	B	1 1	8 rd								
	SHAR.B #2, Rd	B	1 1	C rd								
	SHAR.W Rd	W	1 1	9 rd								
	SHAR.W #2, Rd	W	1 1	D rd								
	SHAR.L ERd	L	1 1	B 0 erd								
	SHAR.L #2, ERd	L	1 1	F 0 erd								
SHLL	SHLL.B Rd	B	1 0	0 rd								
	SHLL.B #2, Rd	B	1 0	4 rd								
	SHLL.W Rd	W	1 0	1 rd								
	SHLL.W #2, Rd	W	1 0	5 rd								
	SHLL.L ERd	L	1 0	3 0 erd								
	SHLL.L #2, ERd	L	1 0	7 0 erd								
SHLR	SHLR.B Rd	B	1 1	0 rd								
	SHLR.B #2, Rd	B	1 1	4 rd								
	SHLR.W Rd	W	1 1	1 rd								
	SHLR.W #2, Rd	W	1 1	5 rd								
	SHLR.L ERd	L	1 1	3 0 erd								
	SHLR.L #2, ERd	L	1 1	7 0 erd								
SLEEP	SLEEP	–	0 1	8 0								
STC	STC.B CCR,Rd	B	0 2	0 rd								
	STC.B EXR,Rd	B	0 2	1 rd								
	STC.W CCR,@ERd	W	0 1	4 0	6 9	1 erd 0						
	STC.W EXR,@ERd	W	0 1	4 1	6 9	1 erd 0						
	STC.W CCR,@(d:16,ERd)	W	0 1	4 0	6 F	1 erd 0	disp					
	STC.W EXR,@(d:16,ERd)	W	0 1	4 1	6 F	1 erd 0	disp					
	STC.W CCR,@(d:32,ERd)	W	0 1	4 0	7 8	0 erd 0	6 B	A 0	disp			
	STC.W EXR,@(d:32,ERd)	W	0 1	4 1	7 8	0 erd 0	6 B	A 0	disp			
	STC.W CCR,@-ERd	W	0 1	4 0	6 D	1 erd 0						
	STC.W EXR,@-ERd	W	0 1	4 1	6 D	1 erd 0						
	STC.W CCR,@aa:16	W	0 1	4 0	6 B	8 0	abs					
	STC.W EXR,@aa:16	W	0 1	4 1	6 B	8 0	abs					
	STC.W CCR,@aa:32	W	0 1	4 0	6 B	A 0	abs					
	STC.W EXR,@aa:32	W	0 1	4 1	6 B	A 0	abs					
STM	STM .L(ERn-ERn+1) , @-SP	L	0 1	1 0	6 D	F 0 ern						
	STM.L (ERn-ERn+2) , @-SP	L	0 1	2 0	6 D	F 0 ern						
	STM.L (ERn-ERn+3) , @-SP	L	0 1	3 0	6 D	F 0 ern						
STMAC*[1]	STMAC MACH,ERd	L	0 2	2 0 ers								
	STMAC MACL,ERd	L	0 2	3 0 ers								

命令	ニーモニック	サイズ	インストラクションフォーマット															
			第1バイト		第2バイト		第3バイト		第4バイト		第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト
SUB	SUB.B Rs,Rd	B	1	8	rs	rd										
	SUB.W #xx:16,Rd	W	7	9	3	rd	IMM									
	SUB.W Rs,Rd	W	1	9	rs	rd										
	SUB.L #xx:32,ERd	L	7	A	3	0 erd	IMM									
	SUB.L ERs,ERd	L	1	A	1 ers	0 erd										
SUBS	SUBS #1,ERd	L	1	B	0	0 erd										
	SUBS #2,ERd	L	1	B	8	0 erd										
	SUBS #4,ERd	L	1	B	9	0 erd										
SUBX	SUBX #xx:8,Rd	B	B	rd	IMM											
	SUBX Rs,Rd*3	B	1	E	rs	rd										
TAS	TAS @ERd	B	0	1	E	0	7 B		0 erd C							
TRAPA	TRAPA #x:2	—	5	7	00 IMM	0										
XOR	XOR.B #xx:8,Rd	B	D	rd	IMM											
	XOR.B Rs,Rd	B	1	5	rs	rd										
	XOR.W #xx:16,Rd	W	7	9	5	rd	IMM									
	XOR.W Rs,Rd	W	6	5	rs	rd										
	XOR.L #xx:32,ERd	L	7	A	5	0 erd	IMM									
	XOR.L ERs,ERd	L	0	1	F	0	6	5	0 ers 0 erd							
XORC	XORC #xx:8,CCR	B	0	5	IMM											
	XORC #xx:8,EXR	B	0	1	4	1	0	5	IMM							

【注】*1　H8S/2600 CPUでのみサポートしています。

*2　MOV.L ERs,@ (d:32,ERd) 命令の第4バイト、ビット7は1、0どちらでも動作可能です。

*3　TAS命令を使用する場合は、レジスタER0、ER1、ER4、ER5を使用してください。

《記号説明》

IMM： イミディエイトデータ（2、3、8、16、32 ビット）
abs： 絶対アドレス（8、16、24、32 ビット）
disp： ディスプレースメント（8、16、32 ビット）
rs、rd、rn： レジスタフィールド（4 ビットで、8 ビットレジスタまたは 16 ビットレジスタを指定します。rs、rd、rn はそれぞれオペランド形式の Rs、Rd、Rn に対応します。）
ers、erd、ern、erm： レジスタフィールド（3 ビットで、アドレスレジスタまたは 32 ビットレジスタを指定します。ers、erd、ern、erm はそれぞれオペランド形式の ERs、ERd、ERn、ERm に対応します。）

レジスタフィールドと汎用レジスタの対応を下表に示します。

アドレスレジスタ 32 ビットレジスタ		16 ビットレジスタ		8 ビットレジスタ	
レジスタフィールド	汎用レジスタ	レジスタフィールド	汎用レジスタ	レジスタフィールド	汎用レジスタ
000	ER0	0000	R0	0000	R0H
001	ER1	0001	R1	0001	R1H
⋮	⋮	⋮	⋮	⋮	⋮
111	ER7	0111	R7	0111	R7H
		1000	E0	1000	R0L
		1001	E1	1001	R1L
		⋮	⋮	⋮	⋮
		1111	E7	1111	R7L

2.5 オペレーションコードマップ

表 2.3 にオペレーションコードマップを示します。

表2.3 オペレーションコードマップ (1)

命令コード：

第1バイト		第2バイト	
AH	AL	BH	BL

BHの最上位ビットが0の場合を示します。

BHの最上位ビットが1の場合を示します。

AH＼AL	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	表2.3 (2)	STC / STMAC*	LDC / LDMAC*	ORC	XORC	ANDC	LDC	ADD		表2.3 (2)	表2.3 (2)	MOV		ADDX	表2.3 (2)
1	表2.3 (2)	表2.3 (2)	表2.3 (2)	表2.3 (2)	OR	XOR	AND	表2.3 (2)	SUB		表2.3 (2)	表2.3 (2)	CMP		SUBX	表2.3 (2)
2	MOV.B															
3																
4	BRA	BRN	BHI	BLS	BCC	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE
5	MULXU	DIVXU	MULXU	DIVXU	RTS	BSR	RTE	TRAPA	表2.3 (2)	JMP			BSR	JSR		
6	BSET	BNOT	BCLR	BTST	OR	XOR	AND	BST / BIST	MOV		表2.3 (2)		MOV			
7					BOR / BIOR	BXOR / BIXOR	BAND / BIAND	BLD / BILD	MOV	表2.3 (2)	表2.3 (2)	EEPMOV	表2.3 (3)			
8	ADD															
9	ADDX															
A	CMP															
B	SUBX															
C	OR															
D	XOR															
E	AND															
F	MOV															

【注】 * H8S/2600CPUでのみサポートしています。

表2.3　オペレーションコードマップ（2）

命令コード：

第1バイト		第2バイト	
AH	AL	BH	BL

AH AL \ BH	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01	MOV	LDM / STM			LDC / STC		MAC*		SLEEP		CLRMAC*		表2.3 (3)	表2.3 (3)	TAS	表2.3 (3)
0A	INC								ADD							
0B	ADDS					INC		INC	ADDS					INC		INC
0F	DAA								MOV							
10	SHLL			SHLL				SHLL	SHAL			SHAL				SHAL
11	SHLR			SHLR				SHLR	SHAR			SHAR				SHAR
12	ROTXL			ROTXL				ROTXL	ROTL			ROTL				ROTL
13	ROTXR			ROTXR				ROTXR	ROTR			ROTR				ROTR
17	NOT			NOT		EXTU		EXTU	NEG			NEG		EXTS		EXTS
1A	DEC								SUB							
1B	SUBS					DEC		DEC	SUBS					DEC		DEC
1F	DAS								CMP							
58	BRA	BRN	BHI	BLS	BCC	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE
6A	MOV	表2.3 (4)	MOV	表2.3 (4)	MOVFPE				MOV		MOV		MOVTPE			
79	MOV	ADD	CMP	SUB	OR	XOR	AND									
7A	MOV	ADD	CMP	SUB	OR	XOR	AND									

【注】＊ H8S/2600CPUでのみサポートしています。

表2.3　オペレーションコードマップ（3）

命令コード：

第1バイト		第2バイト		第3バイト		第4バイト	
AH	AL	BH	BL	CH	CL	DH	DL

DHの最上位ビットが0の場合を示します。
DHの最上位ビットが1の場合を示します。

AH AL BH BL CH ＼ CL	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01C05	MULXS		MULXS													
01D05		DIVXS		DIVXS												
01F06					OR	XOR	AND									
7Cr06*1				BTST												
7Cr07*1				BTST	BOR / BIOR	BXOR / BIXOR	BAND / BIAND	BLD / BILD								
7Dr06*1	BSET	BNOT	BCLR					BST / BIST								
7Dr07*1	BSET	BNOT	BCLR													
7Eaa6*2				BTST												
7Eaa7*2				BTST	BOR / BIOR	BXOR / BIXOR	BAND / BIAND	BLD / BILD								
7Faa6*2	BSET	BNOT	BCLR					BST / BIST								
7Faa7*2	BSET	BNOT	BCLR													

【注】 *1 r はレジスタ指定部
*2 aaは絶対アドレス指定

表2.3　オペレーションコードマップ（4）

命令コード：

第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト	
AH	AL	BH	BL	CH	CL	DH	DL	EH	EL	FH	FL

FHの最上位ビットが0の場合を示します。
FHの最上位ビットが1の場合を示します。

EL / AHALBHBLCHCLDHDLEH	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
6A10aaaa6*				BTST												
6A10aaaa7*					BOR / BIOR	BXOR / BIXOR	BAND / BIAND	BLD / BILD								
6A18aaaa6*	BSET	BNOT	BCLR					BST / BIST								
6A18aaaa7*																

命令コード：

第1バイト		第2バイト		第3バイト		第4バイト		第5バイト		第6バイト		第7バイト		第8バイト	
AH	AL	BH	BL	CH	CL	DH	DL	EH	EL	FH	FL	GH	GL	HH	HL

HHの最上位ビットが0の場合を示します。
HHの最上位ビットが1の場合を示します。

GL / AHALBHBL ... FHFLGH	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
6A30aaaaaaaa6*				BTST												
6A30aaaaaaaa7*					BOR / BIOR	BXOR / BIXOR	BAND / BIAND	BLD / BILD								
6A38aaaaaaaa6*	BSET	BNOT	BCLR					BST / BIST								
6A38aaaaaaaa7*																

【注】* aaは絶対アドレス指定

2.6 命令実行ステート数

H8S/2600 CPU の各命令についての実行状態と実行ステート数の計算方法を示します。

表 2.5 に各命令の実行状態として、命令実行中に行われる命令フェッチ、データリード／ライト等のサイクル数を示し、表 2.4 に各々のサイクルに必要なステート数を示します。

命令の実行ステート数は次の計算式で計算されます。各々のサイクルに必要なステート数は製品によって異なります。詳細は当該製品のハードウェアマニュアルを参照してください。

実行ステート数＝$I \cdot S_I + J \cdot S_J + K \cdot S_K + L \cdot S_L + M \cdot S_M + N \cdot S_N$

実行ステート数計算例

（例）

アドバンストモード、プログラム領域およびスタック領域を外部空間に設定、内部周辺モジュールアクセス時 8 ビットバス幅で 2 ステートアクセス、外部デバイスアクセス時 16 ビットバス幅で 3 ステートアクセス 1 ウェイト挿入とした場合。

1. BSET #0, @FFFFC7:8
   表2.5より
   I＝L＝2、J＝K＝M＝N＝0
   表2.4より
   S_I＝4、S_L＝2
   実行ステート数＝2×4＋2×2＝12

2. JSR @@30
   表2.5より
   I＝J＝K＝2、L＝M＝N＝0
   表2.4より
   S_I＝S_J＝S_K＝4
   実行ステート数＝2×4＋2×4＋2×4＝24

表 2.4 実行状態（サイクル）に要するステート数

実行状態（サイクル）	アクセス対象						
	内蔵メモリ	内蔵周辺モジュール		外部デバイス			
				8 ビットバス		16 ビットバス	
		8 ビットバス	16 ビットバス	2 ステートアクセス	3 ステートアクセス	2 ステートアクセス	3 ステートアクセス
命令フェッチ S_I	1	2n	n	4	6+2m	2	3+m*
分岐アドレスリード S_J							
スタック操作 S_K							
バイトデータアクセス S_L		n		2	3+m		
ワードデータアクセス S_M		2n		4	6+2m		
内部動作 S_N	1						

【注】 * MOVFPE、MOVTPE については当該製品のハードウェアマニュアルを参照してください。

【記号説明】

m：外部デバイスアクセス時のウェイトステート数。

n ：内蔵周辺モジュールアクセス時のステート数。具体的な数値は当該製品のハードウェアマニュアルを参照してください。

表 2.5 命令実行状態（サイクル数）

命令	ニーモニック	命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
		I	J	K	L	M	N
ADD	ADD.B #xx:8,Rd	1					
	ADD.B Rs,Rd	1					
	ADD.W #xx:16,Rd	2					
	ADD.W Rs,Rd	1					
	ADD.L #xx:32,ERd	3					
	ADD.L ERs,ERd	1					
ADDS	ADDS #1/2/4,ERd	1					
ADDX	ADDX #xx:8,Rd	1					
	ADDX Rs,Rd	1					
AND	AND.B #xx:8,Rd	1					
	AND.B Rs,Rd	1					
	AND.W #xx:16,Rd	2					
	AND.W Rs,Rd	1					
	AND.L #xx:32,ERd	3					
	AND.L ERs,ERd	2					
ANDC	ANDC #xx:8,CCR	1					
	ANDC #xx:8,EXR	2					
BAND	BAND #xx:3,Rd	1					
	BAND #xx:3,@ERd	2			1		
	BAND #xx:3,@aa:8	2			1		
	BAND #xx:3,@aa:16	3			1		
	BAND #xx:3,@aa:32	4			1		
Bcc	BRA d:8 (BT d:8)	2					
	BRN d:8 (BF d:8)	2					
	BHI d:8	2					
	BLS d:8	2					
	BCC d:8 (BHS d:8)	2					
	BCS d:8 (BLO d:8)	2					
	BNE d:8	2					
	BEQ d:8	2					
	BVC d:8	2					
	BVS d:8	2					
	BPL d:8	2					
	BMI d:8	2					
	BGE d:8	2					
	BLT d:8	2					
	BGT d:8	2					
	BLE d:8	2					
	BRA d:16 (BT d:16)	2					1
	BRN d:16 (BF d:16)	2					1

命令	ニーモニック	命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
		I	J	K	L	M	N
Bcc	BHI d:16	2					1
	BLS d:16	2					1
	BCC d:16 (BHS d:16)	2					1
	BCS d:16 (BLO d:16)	2					1
	BNE d:16	2					1
	BEQ d:16	2					1
	BVC d:16	2					1
	BVS d:16	2					1
	BPL d:16	2					1
	BMI d:16	2					1
	BGE d:16	2					1
	BLT d:16	2					1
	BGT d:16	2					1
	BLE d:16	2					1
BCLR	BCLR #xx:3,Rd	1					
	BCLR #xx:3,@ERd	2			2		
	BCLR #xx:3,@aa:8	2			2		
	BCLR #xx:3,@aa:16	3			2		
	BCLR #xx:3,@aa:32	4			2		
	BCLR Rn,Rd	1					
	BCLR Rn,@ERd	2			2		
	BCLR Rn,@aa:8	2			2		
	BCLR Rn,@aa:16	3			2		
	BCLR Rn,@aa:32	4			2		
BIAND	BIAND #xx:3,Rd	1					
	BIAND #xx:3,@ERd	2			1		
	BIAND #xx:3,@aa:8	2			1		
	BIAND #xx:3,@aa:16	3			1		
	BIAND #xx:3,@aa:32	4			1		
BILD	BILD #xx:3,Rd	1					
	BILD #xx:3,@ERd	2			1		
	BILD #xx:3,@aa:8	2			1		
	BILD #xx:3,@aa:16	3			1		
	BILD #xx:3,@aa:32	4			1		
BIOR	BIOR #xx:8,Rd	1					
	BIOR #xx:8,@ERd	2			1		
	BIOR #xx:8,@aa:8	2			1		
	BIOR #xx:8,@aa:16	3			1		
	BIOR #xx:8,@aa:32	4			1		

命令	ニーモニック	命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
		I	J	K	L	M	N
BIST	BIST #xx:3,Rd	1					
	BIST #xx:3,@ERd	2			2		
	BIST #xx:3,@aa:8	2			2		
	BIST #xx:3,@aa:16	3			2		
	BIST #xx:3,@aa:32	4			2		
BIXOR	BIXOR #xx:3,Rd	1					
	BIXOR #xx:3,@ERd	2			1		
	BIXOR #xx:3,@aa:8	2			1		
	BIXOR #xx:3,@aa:16	3			1		
	BIXOR #xx:3,@aa:32	4			1		
BLD	BLD #xx:3,Rd	1					
	BLD #xx:3,@ERd	2			1		
	BLD #xx:3,@aa:8	2			1		
	BLD #xx:3,@aa:16	3			1		
	BLD #xx:3,@aa:32	4			1		
BNOT	BNOT #xx:3,Rd	1					
	BNOT #xx:3,@ERd	2			2		
	BNOT #xx:3,@aa:8	2			2		
	BNOT #xx:3,@aa:16	3			2		
	BNOT #xx:3,@aa:32	4			2		
	BNOT Rn,Rd	1					
	BNOT Rn,@ERd	2			2		
	BNOT Rn,@aa:8	2			2		
	BNOT Rn,@aa:16	3			2		
	BNOT Rn,@aa:32	4			2		
BOR	BOR #xx:3,Rd	1					
	BOR #xx:3,@ERd	2			1		
	BOR #xx:3,@aa:8	2			1		
	BOR #xx:3,@aa:16	3			1		
	BOR #xx:3,@aa:32	4			1		
BSET	BSET #xx:3,Rd	1					
	BSET #xx:3,@ERd	2			2		
	BSET #xx:3,@aa:8	2			2		
	BSET #xx:3,@aa:16	3			2		
	BSET #xx:3,@aa:32	4			2		
	BSET Rn,Rd	1					
	BSET Rn,@ERd	2			2		
	BSET Rn,@aa:8	2			2		
	BSET Rn,@aa:16	3			2		
	BSET Rn,@aa:32	4			2		

命令	ニーモニック		命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
			I	J	K	L	M	N
BSR	BSR d:8	ノーマル	2		1			
		アドバンスト	2		2			
	BSR d:16	ノーマル	2		1			1
		アドバンスト	2		2			1
BST	BST #xx:3,Rd		1					
	BST #xx:3,@ERd		2			2		
	BST #xx:3,@aa:8		2			2		
	BST #xx:3,@aa:16		3			2		
	BST #xx:3,@aa:32		4			2		
BTST	BTST #xx:3,Rd		1					
	BTST #xx:3,@ERd		2			1		
	BTST #xx:3,@aa:8		2			1		
	BTST #xx:3,@aa:16		3			1		
	BTST #xx:3,@aa:32		4			1		
	BTST Rn,Rd		1					
	BTST Rn,@ERd		2			1		
	BTST Rn,@aa:8		2			1		
	BTST Rn,@aa:16		3			1		
	BTST Rn,@aa:32		4			1		
BXOR	BXOR #xx:3,Rd		1					
	BXOR #xx:3,@ERd		2			1		
	BXOR #xx:3,@aa:8		2			1		
	BXOR #xx:3,@aa:16		3			1		
	BXOR #xx:3,@aa:32		4			1		
CLRMAC*3	CLRMAC		1					1*4*6
CMP	CMP.B #xx:8,Rd		1					
	CMP.B Rs,Rd		1					
	CMP.W #xx:16,Rd		2					
	CMP.W Rs,Rd		1					
	CMP.L #xx:32,ERd		3					
	CMP.L ERs,ERd		1					
DAA	DAA Rd		1					
DAS	DAS Rd		1					
DEC	DEC.B Rd		1					
	DEC.W #1/2,Rd		1					
	DEC.L #1/2,ERd		1					
DIVXS	DIVXS.B Rs,Rd		2					11
	DIVXS.W Rs,ERd		2					19
DIVXU	DIVXU.B Rs,Rd		1					11
	DIVXU.W Rs,ERd		1					19
EEPMOV	EEPMOV.B		2			2n＋2*1		
	EEPMOV.W		2			2n＋2*1		

命令	ニーモニック		命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
			I	J	K	L	M	N
EXTS	EXTS.W Rd		1					
	EXTS.L ERd		1					
EXTU	EXTU.W Rd		1					
	EXTU.L ERd		1					
INC	INC.B Rd		1					
	INC.W #1/2,Rd		1					
	INC.L #1/2,ERd		1					
JMP	JMP @ERn		2					
	JMP @aa:24		2					1
	JMP @@aa:8	ノーマル	2	1				1
		アドバンスト	2	2				1
JSR	JSR @ERn	ノーマル	2		1			
		アドバンスト	2		2			
	JSR @aa:24	ノーマル	2		1			1
		アドバンスト	2		2			1
	JSR @@aa:8	ノーマル	2	1	1			
		アドバンスト	2	2	2			
LDC	LDC #xx:8,CCR		1					
	LDC #xx:8,EXR		2					
	LDC Rs,CCR		1					
	LDC Rs,EXR		1					
	LDC @ERs,CCR		2				1	
	LDC @ERs,EXR		2				1	
	LDC @(d:16,ERs),CCR		3				1	
	LDC @(d:16,ERs),EXR		3				1	
	LDC @(d:32,ERs),CCR		5				1	
	LDC @(d:32,ERs),EXR		5				1	
	LDC @ERs+,CCR		2				1	1
	LDC @ERs+,EXR		2				1	1
	LDC @aa:16,CCR		3				1	
	LDC @aa:16,EXR		3				1	
	LDC @aa:32,CCR		4				1	
	LDC @aa:32,EXR		4				1	
LDM	LDM.L @SP+, (ERn-ERn+1)		2		4			1
	LDM.L @SP+, (ERn-ERn+2)		2		6			1
	LDM.L @SP+, (ERn-ERn+3)		2		8			1
LDMAC *3	LDMAC ERs, MACH		1					1*4*6
	LDMAC ERs, MACL		1					1*4*6
MAC*3	MAC @ERn+, @ERm+		2				2	

命令	ニーモニック	命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
		I	J	K	L	M	N
MOV	MOV.B #xx:8,Rd	1					
	MOV.B Rs,Rd	1					
	MOV.B @ERs,Rd	1			1		
	MOV.B @(d:16,ERs),Rd	2			1		
	MOV.B @(d:32,ERs),Rd	4			1		
	MOV.B @ERs+,Rd	1			1		1
	MOV.B @aa:8,Rd	1			1		
	MOV.B @aa:16,Rd	2			1		
	MOV.B @aa:32,Rd	3			1		
	MOV.B Rs,@ERd	1			1		
	MOV.B Rs,@(d:16,ERd)	2			1		
	MOV.B Rs,@(d:32,ERd)	4			1		
	MOV.B Rs,@-ERd	1			1		1
	MOV.B Rs,@aa:8	1			1		
	MOV.B Rs,@aa:16	2			1		
	MOV.B Rs,@aa:32	3			1		
	MOV.W #xx:16,Rd	2					
	MOV.W Rs,Rd	1					
	MOV.W @ERs,Rd	1				1	
	MOV.W @(d:16,ERs),Rd	2				1	
	MOV.W @(d:32,ERs),Rd	4				1	
	MOV.W @ERs+,Rd	1				1	1
	MOV.W @aa:16,Rd	2				1	
	MOV.W @aa:32,Rd	3				1	
	MOV.W Rs,@ERd	1				1	
	MOV.W Rs,@(d:16,ERd)	2				1	
	MOV.W Rs,@(d:32,ERd)	4				1	
	MOV.W Rs,@-ERd	1				1	1
	MOV.W Rs,@aa:16	2				1	
	MOV.W Rs,@aa:32	3				1	
	MOV.L #xx:32,ERd	3					
	MOV.L ERs,ERd	1					
	MOV.L @ERs,ERd	2				2	
	MOV.L @(d:16,ERs),ERd	3				2	
	MOV.L @(d:32,ERs),ERd	5				2	
	MOV.L @ERs+,ERd	2				2	1
	MOV.L @aa:16,ERd	3				2	
	MOV.L @aa:32,ERd	4				2	
	MOV.L ERs,@ERd	2				2	

命令	ニーモニック		命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
			I	J	K	L	M	N
MOV	MOV.L ERs,@(d:16,ERd)		3				2	
	MOV.L ERs,@(d:32,ERd)		5				2	
	MOV.L ERs,@-ERd		2				2	1
	MOV.L ERs,@aa:16		3				2	
	MOV.L ERs,@aa:32		4				2	
MOVFPE	MOVFPE @aa:16,Rd		2			1*[2]		
MOVTPE	MOVTPE Rs,@aa:16		2			1*[2]		
MULXS	MULXS.B Rs, Rd	H8S/2600	2					2*[4]*[6]
		H8S/2000	2					11
	MULXS.W Rs, ERd	H8S/2600	2					3*[4]*[6]
		H8S/2000	2					19
MULXU	MULXU.B Rs, Rd	H8S/2600	1					2*[4]*[6]
		H8S/2000	1					11
	MULXU.W Rs, ERd	H8S/2600	1					3*[4]*[6]
		H8S/2000	1					19
NEG	NEG.B Rd		1					
	NEG.W Rd		1					
	NEG.L ERd		1					
NOP	NOP		1					
NOT	NOT.B Rd		1					
	NOT.W Rd		1					
	NOT.L ERd		1					
OR	OR.B #xx:8,Rd		1					
	OR.B Rs,Rd		1					
	OR.W #xx:16,Rd		2					
	OR.W Rs,Rd		1					
	OR.L #xx:32,ERd		3					
	OR.L ERs,ERd		2					
ORC	ORC #xx:8,CCR		1					
	ORC #xx:8,EXR		2					
POP	POP.W Rn		1				1	1
	POP.L ERn		2				2	1
PUSH	PUSH.W Rn		1				1	1
	PUSH.L ERn		2				2	1
ROTL	ROTL.B Rd		1					
	ROTL.B #2,Rd		1					
	ROTL.W Rd		1					
	ROTL.W #2,Rd		1					
	ROTL.L ERd		1					
	ROTL.L #2,ERd		1					

命令	ニーモニック		命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
			I	J	K	L	M	N
ROTR	ROTR.B Rd		1					
	ROTR.B #2,Rd		1					
	ROTR.W Rd		1					
	ROTR.W #2,Rd		1					
	ROTR.L ERd		1					
	ROTR.L #2,ERd		1					
ROTXL	ROTXL.B Rd		1					
	ROTXL.B #2,Rd		1					
	ROTXL.W Rd		1					
	ROTXL.W #2,Rd		1					
	ROTXL.L ERd		1					
	ROTXL.L #2,ERd		1					
ROTXR	ROTXR.B Rd		1					
	ROTXR.B #2,Rd		1					
	ROTXR.W Rd		1					
	ROTXR.W #2,Rd		1					
	ROTXR.L ERd		1					
	ROTXR.L #2,ERd		1					
RTE	RTE		2		2/3*1			1
RTS	RTS	ノーマル	2		1			1
		アドバンスト	2		2			1
SHAL	SHAL.B Rd		1					
	SHAL.B #2,Rd		1					
	SHAL.W Rd		1					
	SHAL.W #2,Rd		1					
	SHAL.L ERd		1					
	SHAL.L #2,ERd		1					
SHAR	SHAR.B Rd		1					
	SHAR.B #2,Rd		1					
	SHAR.W Rd		1					
	SHAR.W #2,Rd		1					
	SHAR.L ERd		1					
	SHAR.L #2,ERd		1					
SHLL	SHLL.B Rd		1					
	SHLL.B #2,Rd		1					
	SHLL.W Rd		1					
	SHLL.W #2,Rd		1					
	SHLL.L ERd		1					
	SHLL.L #2,ERd		1					

命令	ニーモニック		命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
			I	J	K	L	M	N
SHLR	SHLR.B Rd		1					
	SHLR.B #2,Rd		1					
	SHLR.W Rd		1					
	SHLR.W #2,Rd		1					
	SHLR.L ERd		1					
	SHLR.L #2,ERd		1					
SLEEP	SLEEP		1					1
STC	STC.B CCR,Rd		1					
	STC.B EXR,Rd		1					
	STC.W CCR,@ERd		2				1	
	STC.W EXR,@ERd		2				1	
	STC.W CCR,@(d:16,ERd)		3				1	
	STC.W EXR,@(d:16,ERd)		3				1	
	STC.W CCR,@(d:32,ERd)		5				1	
	STC.W EXR,@(d:32,ERd)		5				1	
	STC.W CCR,@-ERd		2				1	1
	STC.W EXR,@-ERd		2				1	1
	STC.W CCR,@aa:16		3				1	
	STC.W EXR,@aa:16		3				1	
	STC.W CCR,@aa:32		4				1	
	STC.W EXR,@aa:32		4				1	
STM	STM.L (ERn-ERn+1),@-SP		2		4			1
	STM.L (ERn-ERn+2),@-SP		2		6			1
	STM.L (ERn-ERn+3),@-SP		2		8			1
STMAC *3	STMAC MACH,ERd		1					0*4*6
	STMAC MACL,ERd		1					0*4*6
SUB	SUB.B Rs,Rd		1					
	SUB.W #xx:16,Rd		2					
	SUB.W Rs,Rd		1					
	SUB.L #xx:32,ERd		3					
	SUB.L ERs,ERd		1					
SUBS	SUBS #1/2/4,ERd		1					
SUBX	SUBX #xx:8,Rd		1					
	SUBX Rs,Rd		1					
TAS	TAS @ERd*5		2			2		
TRAPA	TRAPA #x:2	ノーマル	2	1	2/3*1			2
		アドバンスト	2	2	2/3*1			2

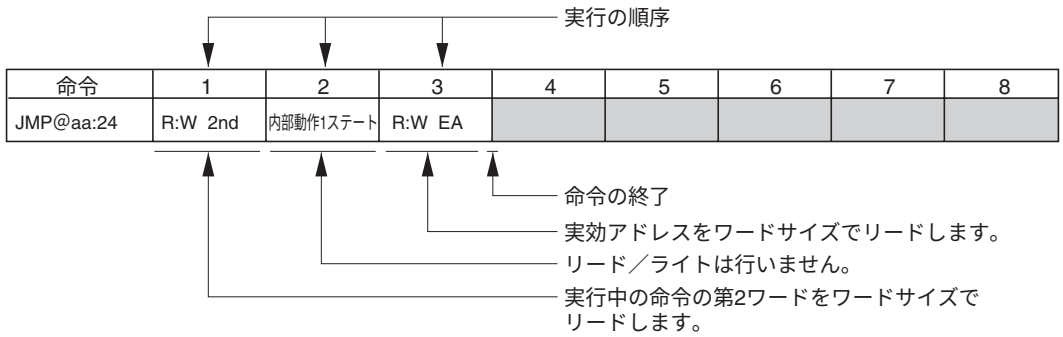
命令	ニーモニック	命令フェッチ	分岐アドレスリード	スタック操作	バイトデータアクセス	ワードデータアクセス	内部動作
		I	J	K	L	M	N
XOR	XOR.B #xx:8,Rd	1					
	XOR.B Rs,Rd	1					
	XOR.W #xx:16,Rd	2					
	XOR.W Rs,Rd	1					
	XOR.L #xx:32,ERd	3					
	XOR.L ERs,ERd	2					
XORC	XORC #xx:8,CCR	1					
	XORC #xx:8,EXR	2					

【注】 *1 EXR が無効なとき 2、有効なとき 3 になります。
*2 単独で使用するとき 4、連続で使用するとき 5 になります。
*3 H8S/2600 CPU でのみサポートしています。
*4 直前の命令によって、内部動作が 0～3 ステート追加される場合があります。
*5 TAS 命令を使用する場合は、レジスタ ER0、ER1、ER4、ER5 を使用してください。
*6 製品によって異なる場合があります。当該製品のハードウェアマニュアルを参照してください。

2.7 命令実行中のバス状態

本 CPU の個々の命令についての実行状態を表 2.6 に示します。実行状態に必要なステート数に関しては、「表 2.4 実行状態（サイクル）に要するステート数」を参照してください。

《表の見方》



《記号説明》

R:B	バイトサイズリードを行います。
R:W	ワードサイズリードを行います。
W:B	バイトサイズライトを行います。
W:W	ワードサイズライトを行います。
2nd	第 2 ワード（第 3・第 4 バイト）のアドレスです。
3rd	第 3 ワード（第 5・第 6 バイト）のアドレスです。
4th	第 4 ワード（第 7・第 8 バイト）のアドレスです。
5th	第 5 ワード（第 9・第 10 バイト）のアドレスです。
NEXT	実行中の命令の直後の命令の先頭アドレスです。
EA	実効アドレスです。
VEC	ベクタアドレスです。

8 ビットバス・3 ステートアクセス・ウェイトなしの場合、上記命令実行中のアドレスバス、$\overline{RD}$、$\overline{WR}$（$\overline{HWR}$ または $\overline{LWR}$）のタイミングを図 2.1 に示します。

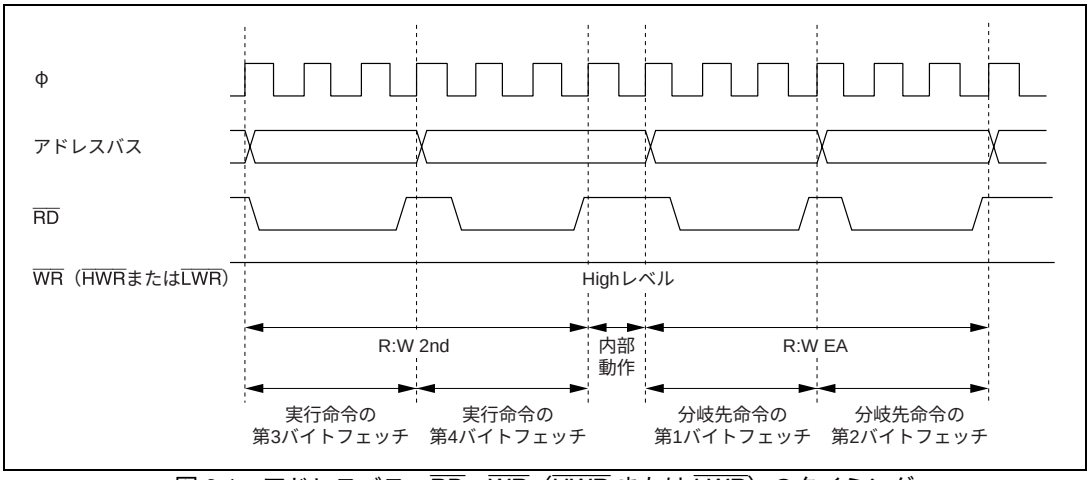


図 2.1　アドレスバス、$\overline{RD}$、$\overline{WR}$（$\overline{HWR}$ または $\overline{LWR}$）のタイミング
（8 ビットバス・3 ステートアクセス・ウェイトなしの場合）

表 2.6 命令の実行状態

命令	1	2	3	4	5	6	7	8	9
ADD.B #xx:8,Rd	R:W NEXT								
ADD.B Rs,Rd	R:W NEXT								
ADD.W #xx:16,Rd	R:W 2nd	R:W NEXT							
ADD.W Rs,Rd	R:W NEXT								
ADD.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT						
ADD.L ERs,ERd	R:W NEXT								
ADDS #1/2/4,ERd	R:W NEXT								
ADDX #xx:8,Rd	R:W NEXT								
ADDX Rs,Rd	R:W NEXT								
AND.B #xx:8,Rd	R:W NEXT								
AND.B Rs,Rd	R:W NEXT								
AND.W #xx:16,Rd	R:W 2nd	R:W NEXT							
AND.W Rs,Rd	R:W NEXT								
AND.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT						
AND.L ERs,ERd	R:W 2nd	R:W NEXT							
ANDC #xx:8,CCR	R:W NEXT								
ANDC #xx:8,EXR	R:W 2nd	R:W NEXT							
BAND #xx:3,Rd	R:W NEXT								
BAND #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT						
BAND #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT						
BAND #xx:3,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BAND #xx:3,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
BRA d:8(BT d:8)	R:W NEXT	R:W EA							
BRN d:8(BF d:8)	R:W NEXT	R:W EA							
BHI d:8	R:W NEXT	R:W EA							
BLS d:8	R:W NEXT	R:W EA							
BCC d:8(BHS d:8)	R:W NEXT	R:W EA							
BCS d:8(BLO d:8)	R:W NEXT	R:W EA							
BNE d:8	R:W NEXT	R:W EA							
BEQ d:8	R:W NEXT	R:W EA							
BVC d:8	R:W NEXT	R:W EA							
BVS d:8	R:W NEXT	R:W EA							
BPL d:8	R:W NEXT	R:W EA							
BMI d:8	R:W NEXT	R:W EA							
BGE d:8	R:W NEXT	R:W EA							
BLT d:8	R:W NEXT	R:W EA							
BGT d:8	R:W NEXT	R:W EA							
BLE d:8	R:W NEXT	R:W EA							
BRA d:16(BT d:16)	R:W 2nd	内部動作 1 ステート	R:W EA						
BRN d:16(BF d:16)	R:W 2nd	内部動作 1 ステート	R:W EA						
BHI d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BLS d:16	R:W 2nd	内部動作 1 ステート	R:W EA						

命令	1	2	3	4	5	6	7	8	9
BCC d:16(BHS d:16)	R:W 2nd	内部動作 1 ステート	R:W EA						
BCS d:16(BLO d:16)	R:W 2nd	内部動作 1 ステート	R:W EA						
BNE d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BEQ d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BVC d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BVS d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BPL d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BMI d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BGE d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BLT d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BGT d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BLE d:16	R:W 2nd	内部動作 1 ステート	R:W EA						
BCLR #xx:3,Rd	R:W NEXT								
BCLR #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BCLR #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BCLR #xx:3,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT	W:B EA				
BCLR #xx:3,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT	W:B EA			
BCLR Rn,Rd	R:W NEXT								
BCLR Rn,@ERd	R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BCLR Rn,@aa:8	R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BCLR Rn,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT	W:B EA				
BCLR Rn,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT	W:B EA			
BIAND #xx:3,Rd	R:W NEXT								
BIAND #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT						
BIAND #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT						
BIAND #xx:3,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BIAND #xx:3,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
BILD #xx:3,Rd	R:W NEXT								
BILD #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT						
BILD #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT						
BILD #xx:3,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BILD #xx:3,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
BIOR #xx:3,Rd	R:W NEXT								
BIOR #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT						
BIOR #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT						
BIOR #xx:3,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BIOR #xx:3,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				

命令		1	2	3	4	5	6	7	8	9
BIST #xx:3,Rd		R:W NEXT								
BIST #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BIST #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BIST #xx:3,@aa:16		R:W 2nd	R:W 3rd	R:B EA	R:W NEXT	W:B EA				
BIST #xx:3,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT	W:B EA			
BIXOR #xx:3,Rd		R:W NEXT								
BIXOR #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT						
BIXOR #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT						
BIXOR #xx:3,@aa:16		R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BIXOR #xx:3,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
BLD #xx:3,Rd		R:W NEXT								
BLD #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT						
BLD #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT						
BLD #xx:3,@aa:16		R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BLD #xx:3,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
BNOT #xx:3,Rd		R:W NEXT								
BNOT #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BNOT #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BNOT #xx:3,@aa:16		R:W 2nd	R:W 3rd	R:B EA	R:W NEXT	W:B EA				
BNOT #xx:3,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT	W:B EA			
BNOT Rn,Rd		R:W NEXT								
BNOT Rn,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BNOT Rn,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BNOT Rn,@aa:16		R:W 2nd	R:W 3rd	R:B EA	R:W NEXT	W:B EA				
BNOT Rn,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT	W:B EA			
BOR #xx:3,Rd		R:W NEXT								
BOR #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT						
BOR #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT						
BOR #xx:3,@aa:16		R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BOR #xx:3,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
BSET #xx:3,Rd		R:W NEXT								
BSET #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BSET #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BSET #xx:3,@aa:16		R:W 2nd	R:W 3rd	R:B EA	R:W NEXT	W:B EA				
BSET #xx:3,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT	W:B EA			
BSET Rn,Rd		R:W NEXT								
BSET Rn,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BSET Rn,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BSET Rn,@aa:16		R:W 2nd	R:W 3rd	R:B EA	R:W NEXT	W:B EA				
BSET Rn,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT	W:B EA			
BSR d:8	ノーマル	R:W NEXT	R:W EA	W:W スタック						
	アドバンスト	R:W NEXT	R:W EA	W:W スタック(H)	W:W スタック(L)					
BSR d:16	ノーマル	R:W 2nd	内部動作 1 ステート	R:W EA	W:W スタック					
	アドバンスト	R:W 2nd	内部動作 1 ステート	R:W EA	W:W スタック(H)	W:W スタック(L)				

命令	1	2	3	4	5	6	7	8	9
BST #xx:3,Rd	R:W NEXT								
BST #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BST #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT	W:B EA					
BST #xx:3,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT	W:B EA				
BST #xx:3,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT	W:B EA			
BTST #xx:3,Rd	R:W NEXT								
BTST #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT						
BTST #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT						
BTST #xx:3,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BTST #xx:3,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
BTST Rn,Rd	R:W NEXT								
BTST Rn,@ERd	R:W 2nd	R:B EA	R:W NEXT						
BTST Rn,@aa:8	R:W 2nd	R:B EA	R:W NEXT						
BTST Rn,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BTST Rn,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
BXOR #xx:3,Rd	R:W NEXT								
BXOR #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT						
BXOR #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT						
BXOR #xx:3,@aa:16	R:W 2nd	R:W 3rd	R:B EA	R:W NEXT					
BXOR #xx:3,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:B EA	R:W NEXT				
CLRMAC*9	R:W NEXT	内部動作 1 ステート*10							
CMP.B #xx:8,Rd	R:W NEXT								
CMP.B Rs,Rd	R:W NEXT								
CMP.W #xx:16,Rd	R:W 2nd	R:W NEXT							
CMP.W Rs,Rd	R:W NEXT								
CMP.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT						
CMP.L ERs,ERd	R:W NEXT								
DAA Rd	R:W NEXT								
DAS Rd	R:W NEXT								
DEC.B Rd	R:W NEXT								
DEC.W #1/2,Rd	R:W NEXT								
DEC.L #1/2,ERd	R:W NEXT								
DIVXS.B Rs,Rd	R:W 2nd	R:W NEXT	内部動作 11 ステート						
DIVXS.W Rs,ERd	R:W 2nd	R:W NEXT	内部動作 19 ステート						
DIVXU.B Rs,Rd	R:W NEXT	内部動作 11 ステート							
DIVXU.W Rs,ERd	R:W NEXT	内部動作 19 ステート							
EEPMOV.B	R:W 2nd	R:B EAs*1	R:B EAd*1	R:B EAs*2	W:B EAd*2	R:W NEXT			
EEPMOV.W	R:W 2nd	R:B EAs*1	R:B EAd*1	R:B EAs*2	W:B EAd*2	R:W NEXT			
EXTS.W Rd	R:W NEXT			← n 回繰り返す*2 →					
EXTS.L ERd	R:W NEXT								
EXTU.W Rd	R:W NEXT								
EXTU.L ERd	R:W NEXT								
INC.B Rd	R:W NEXT								
INC.W #1/2,Rd	R:W NEXT								
INC.L #1/2,ERd	R:W NEXT								
JMP @ERn	R:W NEXT	R:W EA							

命令		1	2	3	4	5	6	7	8	9
JMP @aa:24		R:W 2nd	内部動作 1 ステート	R:W EA						
JMP @@aa:8	ノーマル	R:W NEXT	R:W aa:8	内部動作 1 ステート	R:W EA					
	アドバンスト	R:W NEXT	R:W aa:8(H)	R:W aa:8(L)	内部動作 1 ステート	R:W EA				
JSR @ERn	ノーマル	R:W NEXT	R:W EA	W:W スタック						
	アドバンスト	R:W NEXT	R:W EA	W:W:M スタック(H)	W:W スタック(L)					
JSR @aa:24	ノーマル	R:W 2nd	内部動作 1 ステート	R:W EA	W:W スタック					
	アドバンスト	R:W 2nd	内部動作 1 ステート	R:W EA	W:W スタック(H)	W:W スタック(L)				
JSR @@aa:8	ノーマル	R:W NEXT	R:W aa:8	W:W スタック	R:W EA					
	アドバンスト	R:W NEXT	R:W aa:8(H)	R:W aa:8(L)	W:W スタック(H)	W:W スタック(L)	R:W EA			
LDC #xx:8,CCR		R:W NEXT								
LDC #xx:8,EXR		R:W 2nd	R:W NEXT							
LDC Rs,CCR		R:W NEXT								
LDC Rs,EXR		R:W NEXT								
LDC @ERs,CCR		R:W 2nd	R:W NEXT	R:W EA						
LDC @ERs,EXR		R:W 2nd	R:W NEXT	R:W EA						
LDC@(d:16,ERs),CCR		R:W 2nd	R:W 3rd	R:W NEXT	R:W EA					
LDC@(d:16,ERs),EXR		R:W 2nd	R:W 3rd	R:W NEXT	R:W EA					
LDC@(d:32,ERs),CCR		R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	R:W EA			
LDC@(d:32,ERs),EXR		R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	R:W EA			
LDC @ERs+,CCR		R:W 2nd	R:W NEXT	内部動作 1 ステート	R:W EA					
LDC @ERs+,EXR		R:W 2nd	R:W NEXT	内部動作 1 ステート	R:W EA					
LDC @aa:16,CCR		R:W 2nd	R:W 3rd	R:W NEXT	R:W EA					
LDC @aa:16,EXR		R:W 2nd	R:W 3rd	R:W NEXT	R:W EA					
LDC @aa:32,CCR		R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:W EA				
LDC @aa:32,EXR		R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:W EA				
LDM.L @SP+, (ERn-ERn+1)		R:W 2nd	R:W NEXT	内部動作 1 ステート	R:W スタック(H)*3	R:W スタック(L)*3				
LDM.L @SP+, (ERn-ERn+2)		R:W 2nd	R:W NEXT	内部動作 1 ステート	R:W スタック(H)*3	R:W スタック(L)*3				
LDM.L @SP+, (ERn-ERn+3)		R:W 2nd	R:W NEXT	内部動作 1 ステート	R:W スタック(H)*3	R:W スタック(L)*3				
LDMAC ERs,MACH*9		R:W NEXT	内部動作 1 ステート*10		← n 回繰り返す*3 →					
LDMAC ERs,MACL*9		R:W NEXT	内部動作 1 ステート*10							
MAC @ERn+, @ERm+*9		R:W 2nd	R:W NEXT	R:W EAn	R:W EAm					
MOV.B #xx:8,Rd		R:W NEXT								

命令	1	2	3	4	5	6	7	8	9
MOV.B Rs,Rd	R:W NEXT								
MOV.B @ERs,Rd	R:W NEXT	R:B EA							
MOV.B @(d:16,ERs), Rd	R:W 2nd	R:W NEXT	R:B EA						
MOV.B @(d:32,ERs), Rd	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:B EA				
MOV.B @ERs+,Rd	R:W NEXT	内部動作 1 ステート	R:B EA						
MOV.B @aa:8,Rd	R:W NEXT	R:B EA							
MOV.B @aa:16,Rd	R:W 2nd	R:W NEXT	R:B EA						
MOV.B @aa:32,Rd	R:W 2nd	R:W 3rd	R:W NEXT	R:B EA					
MOV.B Rs,@ERd	R:W NEXT	W:B EA							
MOV.B Rs, @(d:16,ERd)	R:W 2nd	R:W NEXT	W:B EA						
MOV.B Rs, @(d:32,ERd)	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	W:B EA				
MOV.B Rs,@-ERd	R:W NEXT	内部動作 1 ステート	W:B EA						
MOV.B Rs,@aa:8	R:W NEXT	W:B EA							
MOV.B Rs,@aa:16	R:W 2nd	R:W NEXT	W:B EA						
MOV.B Rs,@aa:32	R:W 2nd	R:W 3rd	R:W NEXT	W:B EA					
MOV.W #xx:16,Rd	R:W 2nd	R:W NEXT							
MOV.W Rs,Rd	R:W NEXT								
MOV.W @ERs,Rd	R:W NEXT	R:W EA							
MOV.W @(d:16,ERs), Rd	R:W 2nd	R:W NEXT	R:W EA						
MOV.W @(d:32,ERs), Rd	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:W EA				
MOV.W @ERs+,Rd	R:W NEXT	内部動作 1 ステート	R:W EA						
MOV.W @aa:16,Rd	R:W 2nd	R:W NEXT	R:W EA						
MOV.W @aa:32,Rd	R:W 2nd	R:W 3rd	R:W NEXT	R:B EA					
MOV.W Rs,@ERd	R:W NEXT	W:W EA							
MOV.W Rs, @(d:16,ERd)	R:W 2nd	R:W NEXT	W:W EA						
MOV.W Rs, @(d:32,ERd)	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	W:W EA				
MOV.W Rs,@-ERd	R:W NEXT	内部動作 1 ステート	W:W EA						
MOV.W Rs,@aa:16	R:W 2nd	R:W NEXT	W:W EA						
MOV.W Rs,@aa:32	R:W 2nd	R:W 3rd	R:W NEXT	W:W EA					
MOV.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT						
MOV.L ERs,ERd	R:W NEXT								
MOV.L @ERs,ERd	R:W 2nd	R:W NEXT	R:W EA	R:W EA+2					
MOV.L @(d:16,ERs), ERd	R:W 2nd	R:W 3rd	R:W NEXT	R:W EA	R:W EA+2				
MOV.L @(d:32,ERs), ERd	R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	R:W EA	R:W EA+2		

命令		1	2	3	4	5	6	7	8	9
MOV.L @ERs+,ERd		R:W 2nd	R:W NEXT	内部動作 1 ステート	R:W EA	R:W EA+2				
MOV.L @aa:16,ERd		R:W 2nd	R:W 3rd	R:W NEXT	R:W EA	R:W EA+2				
MOV.L @aa:32,ERd		R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:W EA	R:W EA+2			
MOV.L ERs,@ERd		R:W 2nd	R:W NEXT	W:W EA	W:W EA+2					
MOV.L ERs, @(d:16,ERd)		R:W 2nd	R:W 3rd	R:W NEXT	W:W EA	W:W EA+2				
MOV.L ERs, @(d:32,ERd)		R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	W:W:M EA	W:W EA+2		
MOV.L ERs,@-ERd		R:W 2nd	R:W NEXT	内部動作 1 ステート	W:W EA	W:W EA+2				
MOV.L ERs,@aa:16		R:W 2nd	R:W 3rd	R:W NEXT	W:W EA	W:W EA+2				
MOV.L ERs,@aa:32		R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	W:W EA	W:W EA+2			
MOVFPE @aa:16,Rd		R:W 2nd	R:W NEXT	R:W*4 EA						
MOVTPE Rs,@aa:16		R:W 2nd	R:W NEXT	W:B*4 EA						
MULXS.B Rs, Rd	H8S/ 2600	R:W 2nd	R:W NEXT	内部動作 2 ステート*10						
	H8S/ 2000	R:W 2nd	R:W NEXT	内部動作 11 ステート						
MULXS.W Rs, ERd	H8S/ 2600	R:W 2nd	R:W NEXT	内部動作 3 ステート*10						
	H8S/ 2000	R:W 2nd	R:W NEXT	内部動作 19 ステート						
MULXU.B Rs, Rd	H8S/ 2600	R:W NEXT	内部動作 2 ステート*10							
	H8S/ 2000	R:W NEXT	内部動作 11 ステート							
MULXU.W Rs, ERd	H8S/ 2600	R:W NEXT	内部動作 3 ステート*10							
	H8S/ 2000	R:W NEXT	内部動作 19 ステート							
NEG.B Rd		R:W NEXT								
NEG.W Rd		R:W NEXT								
NEG.L ERd		R:W NEXT								
NOP		R:W NEXT								
NOT.B Rd		R:W NEXT								
NOT.W Rd		R:W NEXT								
NOT.L ERd		R:W NEXT								
OR.B #xx:8,Rd		R:W NEXT								
OR.B Rs,Rd		R:W NEXT								
OR.W #xx:16,Rd		R:W 2nd	R:W NEXT							
OR.W Rs,Rd		R:W NEXT								
OR.L #xx:32,ERd		R:W 2nd	R:W 3rd	R:W NEXT						
OR.L ERs,ERd		R:W 2nd	R:W NEXT							
ORC #xx:8,CCR		R:W NEXT								
ORC #xx:8,EXR		R:W 2nd	R:W NEXT							
POP.W Rn		R:W NEXT	内部動作 1 ステート	R:W EA						

命令		1	2	3	4	5	6	7	8	9
POP.L ERn		R:W 2nd	R:W NEXT	内部動作 1 ステート	R:W EA	R:W EA+2				
PUSH.W Rn		R:W NEXT	内部動作 1 ステート	W:W EA						
PUSH.L ERn		R:W 2nd	R:W NEXT	内部動作	W:W EA	W:W EA+2				
ROTL.B Rd		R:W NEXT								
ROTL.B #2,Rd		R:W NEXT								
ROTL.W Rd		R:W NEXT								
ROTL.W #2,Rd		R:W NEXT								
ROTL.L ERd		R:W NEXT								
ROTL.L #2,ERd		R:W NEXT								
ROTR.B Rd		R:W NEXT								
ROTR.B #2,Rd		R:W NEXT								
ROTR.W Rd		R:W NEXT								
ROTR.W #2,Rd		R:W NEXT								
ROTR.L ERd		R:W NEXT								
ROTR.L #2,ERd		R:W NEXT								
ROTXL.B Rd		R:W NEXT								
ROTXL.B #2,Rd		R:W NEXT								
ROTXL.W Rd		R:W NEXT								
ROTXL.W #2,Rd		R:W NEXT								
ROTXL.L ERd		R:W NEXT								
ROTXL.L #2,ERd		R:W NEXT								
ROTXR.B Rd		R:W NEXT								
ROTXR.B #2,Rd		R:W NEXT								
ROTXR.W Rd		R:W NEXT								
ROTXR.W #2,Rd		R:W NEXT								
ROTXR.L ERd		R:W NEXT								
ROTXR.L #2,ERd		R:W NEXT								
RTE		R:W NEXT	R:W スタック (EXR)	R:W スタック(H)	R:W スタック(L)	内部動作 1 ステート	R:W*5			
RTS	ノーマル	R:W NEXT	R:W スタック	内部動作 1 ステート	R:W*5					
	アドバンスト	R:W NEXT	R:W スタック(H)	R:W スタック(L)	内部動作 1 ステート	R:W*5				
SHAL.B Rd		R:W NEXT								
SHAL.B #2,Rd		R:W NEXT								
SHAL.W Rd		R:W NEXT								
SHAL.W #2,Rd		R:W NEXT								
SHAL.L ERd		R:W NEXT								
SHAL.L #2,ERd		R:W NEXT								
SHAR.B Rd		R:W NEXT								
SHAR.B #2,Rd		R:W NEXT								
SHAR.W Rd		R:W NEXT								
SHAR.W #2,Rd		R:W NEXT								
SHAR.L ERd		R:W NEXT								

命令	1	2	3	4	5	6	7	8	9
SHAR.L #2,ERd	R:W NEXT								
SHLL.B Rd	R:W NEXT								
SHLL.B #2,Rd	R:W NEXT								
SHLL.W Rd	R:W NEXT								
SHLL.W #2,Rd	R:W NEXT								
SHLL.L ERd	R:W NEXT								
SHLL.L #2,ERd	R:W NEXT								
SHLR.B Rd	R:W NEXT								
SHLR.B #2,Rd	R:W NEXT								
SHLR.W Rd	R:W NEXT								
SHLR.W #2,Rd	R:W NEXT								
SHLR.L ERd	R:W NEXT								
SHLR.L #2,ERd	R:W NEXT								
SLEEP	R:W NEXT	内部動作 1 ステート							
STC CCR,Rd	R:W NEXT								
STC EXR,Rd	R:W NEXT								
STC CCR,@ERd	R:W 2nd	R:W NEXT	W:W EA						
STC EXR,@ERd	R:W 2nd	R:W NEXT	W:W EA						
STC CCR, @(d:16,ERd)	R:W 2nd	R:W 3rd	R:W NEXT	W:W EA					
STC EXR, @(d:16,ERd)	R:W 2nd	R:W 3rd	R:W NEXT	W:W EA					
STC CCR, @(d:32,ERd)	R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	W:W EA			
STC EXR, @(d:32,ERd)	R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	W:W EA			
STC CCR,@-ERd	R:W 2nd	R:W NEXT	内部動作 1 ステート	W:W EA					
STC EXR,@-ERd	R:W 2nd	R:W NEXT	内部動作 1 ステート	W:W EA					
STC CCR,@aa:16	R:W 2nd	R:W 3rd	R:W NEXT	W:W EA					
STC EXR,@aa:16	R:W 2nd	R:W 3rd	R:W NEXT	W:W EA					
STC CCR,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	W:W EA				
STC EXR,@aa:32	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	W:W EA				
STM.L (ERn-ERn+1), @-SP	R:W 2nd	R:W NEXT	内部動作 1 ステート	W:W スタック(H)*3	W:W スタック(L)*3				
STM.L (ERn-ERn+2), @-SP	R:W 2nd	R:W NEXT	内部動作 1 ステート	W:W スタック(H)*3	W:W スタック(L)*3				
STM.L (ERn-ERn+3), @-SP	R:W 2nd	R:W NEXT	内部動作 1 ステート	W:W スタック(H)*3	W:W スタック(L)*3				
STMAC MACH,ERd*9	R:W NEXT	*10		← n 回繰り返す*3 →					
STMAC MACL,ERd*9	R:W NEXT	*10							
SUB.B Rs,Rd	R:W NEXT								
SUB.W #xx:16,Rd	R:W 2nd	R:W NEXT							
SUB.W Rs,Rd	R:W NEXT								
SUB.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT						

命令		1	2	3	4	5	6	7	8	9
SUB.L ERs,ERd		R:W NEXT								
SUBS #1/2/4,ERd		R:W NEXT								
SUBX #xx:8,Rd		R:W NEXT								
SUBX Rs,Rd		R:W NEXT								
TAS @ERd*11		R:W 2nd	R:W NEXT	R:B EA	W:B EA					
TRAPA #x:2	ノーマル	R:W NEXT	内部動作 1 ステート	W:W スタック(L)	W:W スタック(H)	W:W スタック (EXR)	R:W VEC	内部動作 1 ステート	R:W*8	
	アドバンスト	R:W NEXT	内部動作 1 ステート	W:W スタック(L)	W:W スタック(H)	W:W スタック (EXR)	R:W VEC	R:W VEC+2	内部動作 1 ステート	R:W*8
XOR.B #xx8,Rd		R:W NEXT								
XOR.B Rs,Rd		R:W NEXT								
XOR.W #xx:16,Rd		R:W 2nd	R:W NEXT							
XOR.W Rs,Rd		R:W NEXT								
XOR.L #xx:32,ERd		R:W 2nd	R:W 3rd	R:W NEXT						
XOR.L ERs,ERd		R:W 2nd	R:W NEXT							
XORC #xx:8,CCR		R:W NEXT								
XORC #xx:8,EXR		R:W 2nd	R:W NEXT							
リセット 例外処理	ノーマル	R:W VEC	内部動作 1 ステート	R:W*6						
	アドバンスト	R:W VEC	R:W VEC+2	内部動作 1 ステート	R:W*6					
割り込み 例外処理	ノーマル	R:W*7	内部動作 1 ステート	W:W スタック(L)	W:W スタック(H)	W:W スタック (EXR)	R:W VEC	内部動作 1 ステート	R:W*8	
	アドバンスト	R:W*7	内部動作 1 ステート	W:W スタック(L)	W:W スタック(H)	W:W スタック (EXR)	R:W VEC	R:W VEC+2	内部動作 1 ステート	R:W*8

【注】 *1 EAs は ER5、EAd は ER6 の内容です。

*2 EAs は ER5、EAd は ER6 の内容で、実行後それぞれ 1 が加算されます。
また、n は R4L または R4 の初期値であり、n=0 のときこれらの実行は行われません。

*3 2 本退避／復帰時は 2 回、3 本退避／復帰時は 3 回、4 本退避／復帰時は 4 回繰り返します。

*4 バイトサイズリード／ライトに必要なステート数は当該製品のハードウェアマニュアルを参照してください。

*5 リターン後の先頭アドレスです。

*6 プログラムのスタートアドレスです。

*7 プリフェッチアドレスです。退避される PC に 2 を加算したアドレスです。
また、スリープモード、ソフトウェアスタンバイモードからの復帰時にはリード動作は行われず、内部動作となります。

*8 割り込み処理ルーチンの先頭アドレスです。

*9 H8S/2600 CPU でのみサポートしています。

*10 直前の命令によって、内部動作が 0～3 ステート追加される場合があります。

*11 TAS 命令を使用する場合は、レジスタ ER0、ER1、ER4、ER5 を使用してください。

2.8 コンディションコードの変化

CPU の各命令について、命令実行後のコンディションコードの変化を示します。以下に、表中で使われている記号を説明します。

m=	31	：ロングワードサイズのとき
	15	：ワードサイズのとき
	7	：バイトサイズのとき
Si		：ソースオペランドのビット i
Di		：デスティネーションオペランドのビット i
Ri		：結果のビット i
Dn		：デスティネーションオペランドの指定されたビット
−		：影響なし
↕		：実行結果に応じて変化（定義参照）
0		：常に 0 にクリア
1		：常に 1 にセット
*		：値を保証しません
Z'		：実行前の Z フラグ
C'		：実行前の C フラグ

表 2.7　コンディションコードの変化

命　令	H	N	Z	V	C	定　義
ADD	↕	↕	↕	↕	↕	H＝Sm-4・Dm-4 + Dm-4・$\overline{Rm-4}$ + Sm-4・$\overline{Rm-4}$ N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ V＝Sm・Dm・$\overline{Rm}$ + $\overline{Sm}$・$\overline{Dm}$・Rm C＝Sm・Dm + Dm・$\overline{Rm}$ + Sm・$\overline{Rm}$
ADDS	—	—	—	—	—	
ADDX	↕	↕	↕	↕	↕	H＝Sm-4・Dm-4 + Dm-4・$\overline{Rm-4}$ + Sm-4・$\overline{Rm-4}$ N＝Rm Z＝Z'・$\overline{Rm}$・　……　・$\overline{R0}$ V＝Sm・Dm・$\overline{Rm}$ + $\overline{Sm}$・$\overline{Dm}$・Rm C＝Sm・Dm + Dm・$\overline{Rm}$ + Sm・$\overline{Rm}$
AND	—	↕	↕	0	—	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$
ANDC	↕	↕	↕	↕	↕	実行結果の対応するビットの値が格納されます。 EXR のときはどのフラグも変化しません。
BAND	—	—	—	—	↕	C＝C'・Dn
Bcc	—	—	—	—	—	
BCLR	—	—	—	—	—	
BIAND	—	—	—	—	↕	C＝C'・$\overline{Dn}$
BILD	—	—	—	—	↕	C＝$\overline{Dn}$
BIOR	—	—	—	—	↕	C＝C' + $\overline{Dn}$
BIST	—	—	—	—	—	
BIXOR	—	—	—	—	↕	C＝C'・Dn + $\overline{C'}$・$\overline{Dn}$
BLD	—	—	—	—	↕	C＝Dn
BNOT	—	—	—	—	—	
BOR	—	—	—	—	↕	C＝C' + Dn
BSET	—	—	—	—	—	
BSR	—	—	—	—	—	
BST	—	—	—	—	—	
BTST	—	—	↕	—	—	Z=$\overline{Dn}$
BXOR	—	—	—	—	↕	C＝C'・$\overline{Dn}$ + $\overline{C'}$・Dn
CLRMAC*	—	—	—	—	—	
CMP	↕	↕	↕	↕	↕	H＝Sm-4・$\overline{Dm-4}$ + $\overline{Dm-4}$・Rm-4 + Sm-4・Rm-4 N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ V＝$\overline{Sm}$・Dm・$\overline{Rm}$ + Sm・$\overline{Dm}$・Rm C＝Sm・$\overline{Dm}$ + $\overline{Dm}$・Rm + Sm・Rm
DAA	*	↕	↕	*	↕	N=Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ C：10 進加算のキャリ

命　令	H	N	Z	V	C	定　義
DAS	＊	↕	↕	＊	↕	N=Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$ C：10 進減算のボロー
DEC	－	↕	↕	↕	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$ V＝Dm・$\overline{Rm}$
DIVXS	－	↕	↕	－	－	N＝Sm・$\overline{Dm}$ + $\overline{Sm}$・Dm Z＝$\overline{Sm}$・$\overline{Sm\text{-}1}$・　……　・$\overline{S0}$
DIVXU	－	↕	↕	－	－	N＝Sm Z＝$\overline{Sm}$・$\overline{Sm\text{-}1}$・　……　・$\overline{S0}$
EEPMOV	－	－	－	－	－	
EXTS	－	↕	↕	0	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
EXTU	－	0	↕	0	－	Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
INC	－	↕	↕	↕	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$ V＝$\overline{Dm}$・Rm
JMP	－	－	－	－	－	
JSR	－	－	－	－	－	
LDC	↕	↕	↕	↕	↕	実行結果の対応するビットの値が格納されます。 EXR のときはどのフラグも変化しません。
LDM	－	－	－	－	－	
LDMAC*	－	－	－	－	－	
MAC*	－	－	－	－	－	
MOV	－	↕	↕	0	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
MOVFPE	－	↕	↕	0	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
MOVTPE	－	↕	↕	0	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
MULXS	－	↕	↕	－	－	N＝R2m Z＝$\overline{R2m}$・$\overline{R2m\text{-}1}$・　……　・$\overline{R0}$
MULXU	－	－	－	－	－	
NEG	↕	↕	↕	↕	↕	H＝Dm-4 + Rm-4 N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$ V＝Dm・Rm C＝Dm + Rm
NOP	－	－	－	－	－	
NOT	－	↕	↕	0	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
OR	－	↕	↕	0	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$

命　令	H	N	Z	V	C	定　義
ORC	↕	↕	↕	↕	↕	実行結果の対応するビットの値が格納されます。 EXR のときはどのフラグも変化しません。
POP	－	↕	↕	0	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$
PUSH	－	↕	↕	0	－	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$
ROTL	－	↕	↕	0	↕	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ C＝Dm（1 ビットのとき）、C＝Dm-1（2 ビットのとき）
ROTR	－	↕	↕	0	↕	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ C＝D0（1 ビットのとき）、C＝D1（2 ビットのとき）
ROTXL	－	↕	↕	0	↕	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ C＝Dm（1 ビットのとき）、C＝Dm-1（2 ビットのとき）
ROTXR	－	↕	↕	0	↕	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ C＝D0（1 ビットのとき）、C＝D1（2 ビットのとき）
RTE	↕	↕	↕	↕	↕	実行結果の対応するビットの値が格納されます。
RTS	－	－	－	－	－	
SHAL	－	↕	↕	↕	↕	
SHAR	－	↕	↕	0	↕	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ C＝D0（1 ビットのとき）、C＝D1（2 ビットのとき）
SHLL	－	↕	↕	0	↕	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ C＝Dm（1 ビットのとき）、C＝Dm-1（2 ビットのとき）
SHLR	－	0	↕	0	↕	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm-1}$・　……　・$\overline{R0}$ C＝D0（1 ビットのとき）、C＝D1（2 ビットのとき）
SLEEP	－	－	－	－	－	
STC	－	－	－	－	－	
STM	－	－	－	－	－	
STMAC*	－	↕	↕	↕	－	N＝MAC 命令の結果、MAC レジスタが負のとき Z＝MAC 命令の結果、MAC レジスタが 0 のとき V＝MAC 命令の結果、オーバフローが発生したとき

命　令	H	N	Z	V	C	定　義
SUB	↕	↕	↕	↕	↕	H＝Sm-4・$\overline{Dm\text{-}4}$ + $\overline{Dm\text{-}4}$・Rm-4 + Sm-4・Rm-4 N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$ V＝$\overline{Sm}$・Dm・$\overline{Rm}$ + Sm・$\overline{Dm}$・Rm C＝Sm・$\overline{Dm}$ + $\overline{Dm}$・Rm + Sm・Rm
SUBS	―	―	―	―	―	
SUBX	↕	↕	↕	↕	↕	H＝Sm-4・$\overline{Dm\text{-}4}$ + $\overline{Dm\text{-}4}$・Rm-4 + Sm-4・Rm-4 N＝Rm Z＝Z'・$\overline{Rm}$・　……　・$\overline{R0}$ V＝$\overline{Sm}$・Dm・$\overline{Rm}$ + Sm・$\overline{Dm}$・Rm C＝Sm・$\overline{Dm}$ + $\overline{Dm}$・Rm + Sm・Rm
TAS	―	↕	↕	0	―	N＝Dm Z＝$\overline{Dm}$・$\overline{Dm\text{-}1}$・　……　・$\overline{D0}$
TRAPA	―	―	―	―	―	
XOR	―	↕	↕	0	―	N＝Rm Z＝$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
XORC	↕	↕	↕	↕	↕	実行結果の対応するビットの値が格納されます。 EXR のときはどのフラグも変化しません。

【注】 ＊　H8S/2600 CPU でのみサポートしています。

3. 処理状態

3.1 概要

本 CPU の処理状態には、リセット状態、例外処理状態、プログラム実行状態、バス権解放状態、および低消費電力状態の 5 種類があります。

処理状態の分類を図 3.1 に、処理状態間の遷移を図 3.2 に示します。

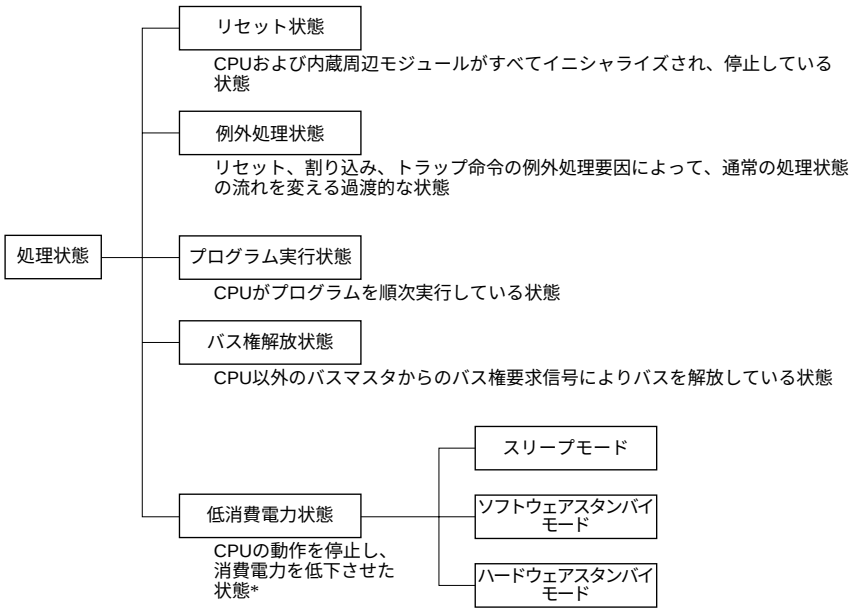
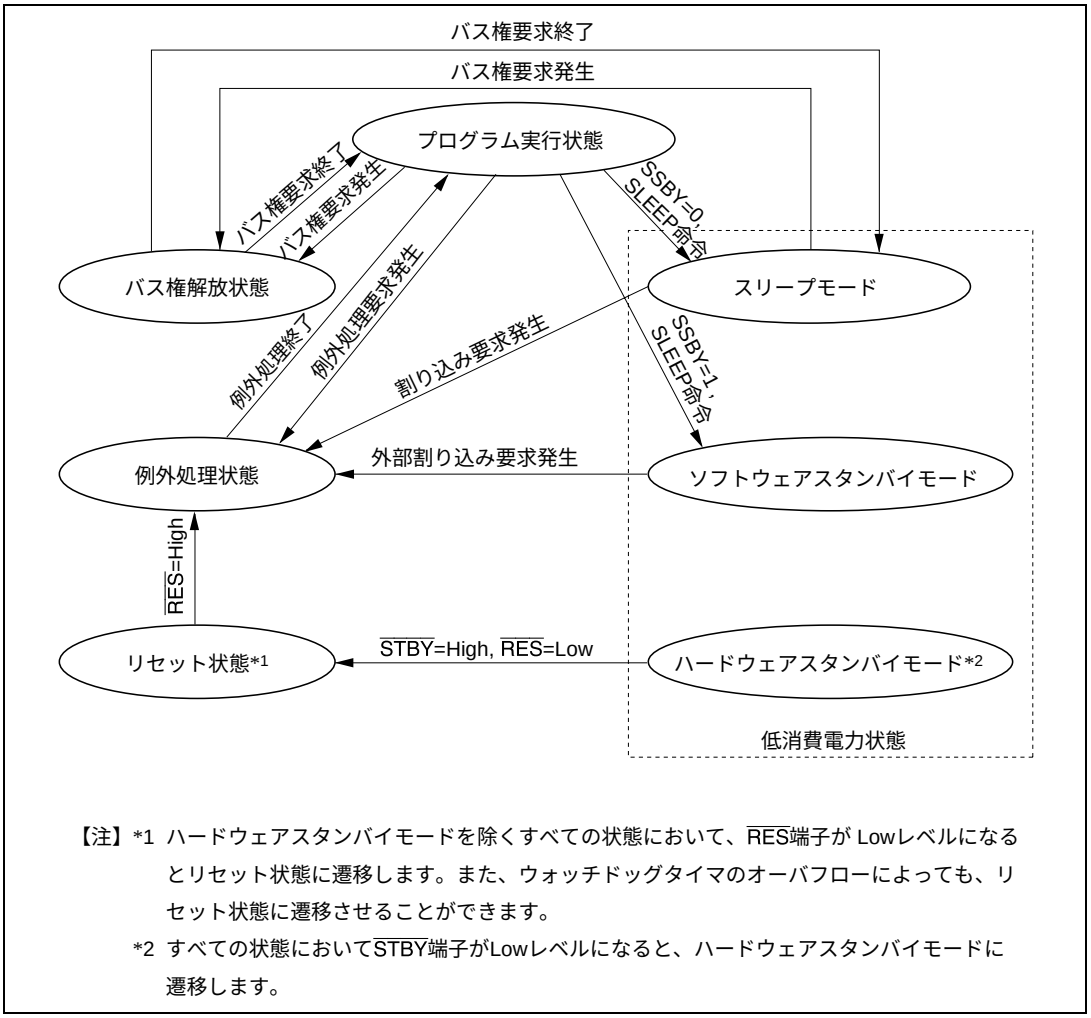


図 3.1 処理状態の分類



【注】*1 ハードウェアスタンバイモードを除くすべての状態において、RES端子が Lowレベルになるとリセット状態に遷移します。また、ウォッチドッグタイマのオーバフローによっても、リセット状態に遷移させることができます。

*2 すべての状態においてSTBY端子がLowレベルになると、ハードウェアスタンバイモードに遷移します。

図 3.2　状態遷移図

3.2 リセット状態

$\overline{\text{RES}}$ 端子が Low レベルになると、実行中の処理は全て中止され、CPU はリセット状態になります。リセット状態ではすべての割り込みが禁止されます。

$\overline{\text{RES}}$ 端子を Low レベルから High レベルにすると、リセット例外処理を開始します。

ウォッチドッグタイマのオーバフローによって、リセット状態とすることもできます。詳細は当該製品のハードウェアマニュアルを参照してください。

3.3 例外処理状態

例外処理状態は、リセット、割り込み、またはトラップ命令の例外処理要因によって起動され、CPU が通常の処理状態の流れを変え、例外処理ベクタテーブルからスタートアドレス（ベクタ）を取り出し、そのスタートアドレスに分岐する過渡的な状態です。

3.3.1 例外処理の種類と優先度

例外処理要因には、リセット、トレース、割り込み、およびトラップ命令があります。表 3.1 に、例外処理の種類と優先度を示します。トラップ命令例外処理は、プログラム実行状態で常に受け付けられます。SYSCR によって設定される割り込み制御モードによって、例外処理やスタックの構造が異なります。

表 3.1 例外処理の種類と優先度

優先度	例外処理要因	例外処理検出タイミング	例外処理開始タイミング
高 ▲	リセット	クロック同期	$\overline{\text{RES}}$ 端子が Low レベルから High レベルに変化すると、ただちに例外処理を開始します。
	トレース	命令実行終了時または例外処理終了時*1	T ビット＝1 の状態で命令または例外処理の実行終了時開始します。
	割り込み	命令実行終了時または例外処理終了時*2	割り込み要求が発生すると、命令実行終了時または例外処理終了時に例外処理を開始します。
低	トラップ命令	TRAPA 命令実行時	TRAPA 命令を実行すると、例外処理を開始します。*3

【注】 *1 トレースは割り込み制御モード 2、3 でのみ有効です。トレース例外処理は、RTE 命令の実行終了後には実行しません。

*2 ANDC、ORC、XORC、LDC 命令の実行終了時点、またはリセット例外処理の終了時点では割り込みの検出を行いません。

*3 トラップ命令例外処理は、プログラム実行状態で常に受け付けられます。

割り込み制御モード、例外処理要因と例外処理の詳細については当該製品のハードウェアマニュアルを参照してください。

3.3.2 リセット例外処理

$\overline{\text{RES}}$ 端子を Low レベルにして、リセット状態とした後、$\overline{\text{RES}}$ 端子を High レベルにすると、リセット例外処理を開始します。

リセット例外処理が起動されると、CPU は、例外処理ベクタテーブルからスタートアドレス（ベクタ）を取り出し、そのスタートアドレスからプログラムの実行を開始します。

リセット例外処理実行中、および終了後は、NMI を含めた全ての割り込みが禁止されます。

3.3.3 トレース

トレースは、割り込み制御モード 2、3 で有効です。

EXR の T ビットが 1 にセットされていると、トレースモードになります。トレースモードが設定されていると、1 命令の実行を終了するたびにトレース例外処理を開始します。

トレース例外処理実行後、EXR の T ビットが 0 にクリアされ、トレースモードが解除されます。割り込みマスクは影響を受けません。

スタックに退避された T ビットは 1 を保持しており、RTE 命令を実行して、トレース例外処理ルーチンから復帰した後は、再び、トレースモードになります。

RTE 命令実行後は、トレース例外処理を行いません。

割り込み制御モード 0、1 では、T ビットの状態に依らず、トレースモードにはなりません。

3.3.4 割り込み例外処理およびトラップ命令例外処理

割り込み例外処理およびトラップ命令例外処理が起動されると、CPU は、SP（ER7）を参照してプログラムカウンタとコントロールレジスタをスタックに退避します。コントロールレジスタの割り込みマスクビットを再設定します。次に、例外処理ベクタテーブルからスタートアドレス（ベクタ）を取り出し、そのスタートアドレスからプログラムの実行を開始します。

例外処理終了後のスタックの状態を図 3.3 に示します。

3.3.5 使用上の注意事項

(1) 割り込みの発生とディスエーブルとの競合

割り込みイネーブルビットをクリアして割り込み要求をディスエーブルする場合、割り込みディスエーブルはその命令実行終了後に有効になります。BCLR 命令、MOV 命令などで割り込みイネーブルビットをクリアする場合、命令実行中にその割り込みが発生すると、命令実行終了時点では当該割り込みはイネーブル状態にあるため、命令実行終了後にその割り込み例外処理を開始します。ただし、その割り込みより優先順位の高い割り込み要求がある場合には優先順位の高い割り込み例外処理を実行し、その割り込みは無視されます。割り込み要因フラグを 0 にクリアする場合も同様です。CPU 側で割り込みをマスクした状態でイネーブルビットまたは割り込み要因フラグを 0 にクリアすれば、上記の競合は発生しません。

(2) 割り込みを禁止している命令

実行直後に割り込み要求を受け付けない命令として、LDC、ANDC、ORC、XORC 命令があります。これらの命令実行終了後は NMI 割り込みを含めて割り込みが禁止され、必ず次の命令を実行します。これらの命令により I ビットを設定した場合、命令実行終了の 2 ステート後に新しい値が有効になります。

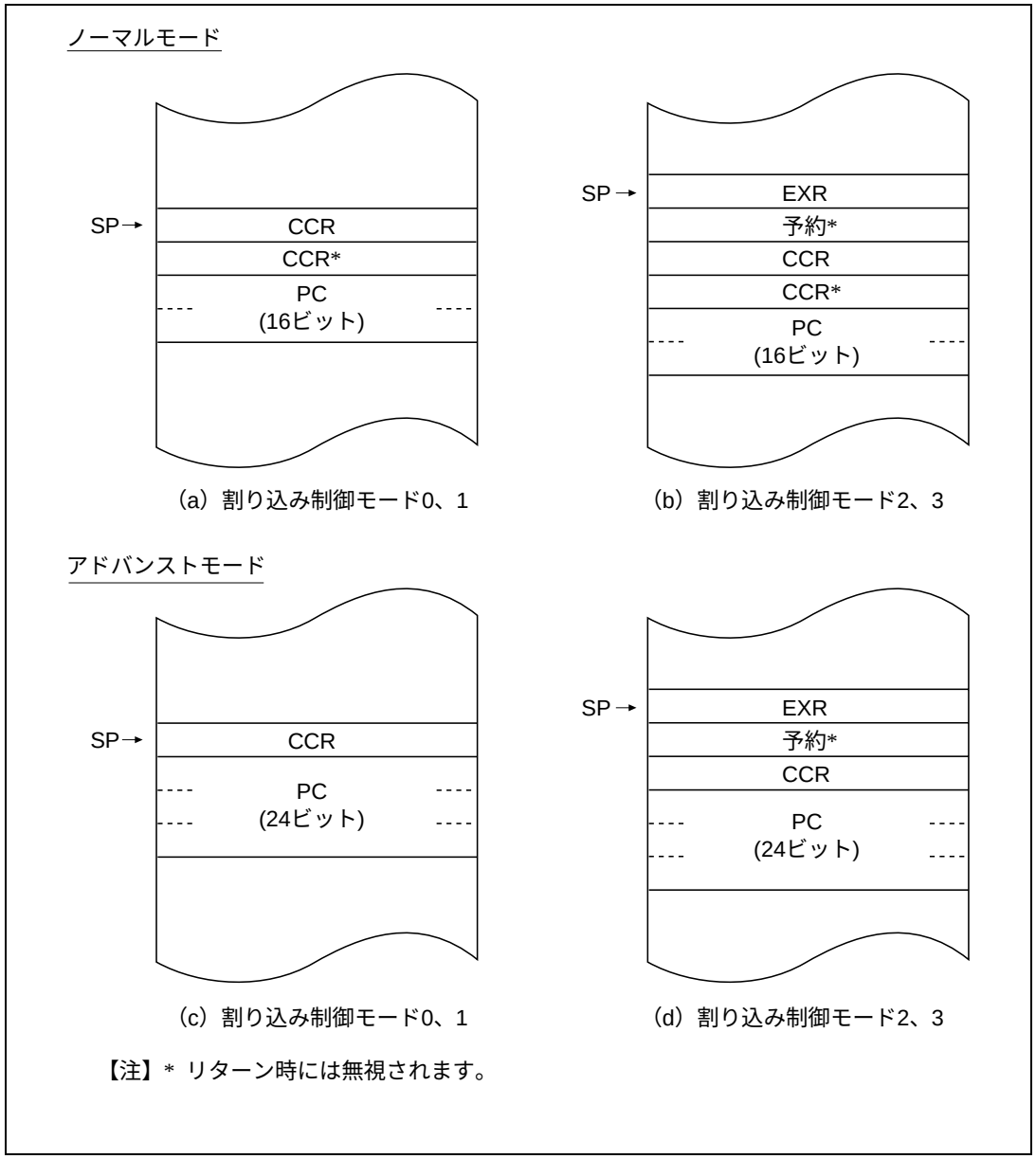
(3) EEPMOV 命令実行中の割り込み

EEPMOV.B 命令と EEPMOV.W 命令では、割り込み動作が異なります。

EEPMOV.B 命令のときは、転送中に NMI を含めた割り込み要求があっても転送終了まで割り込みを受け付けません。

EEPMOV.W 命令のときは、転送中に割り込み要求があった場合、転送サイクルの切れ目で割り込み例外処理が開始されます。このときスタックされる PC の値は次の命令のアドレスとなります。このため、EEPMOV.W 命令実行中に割り込みが発生する場合には、以下のプログラムとしてください。

```
L1:    EEPMOV.W
       MOV.W     R4, R4
       BNE       L1
```



（a）割り込み制御モード0、1

（b）割り込み制御モード2、3

（c）割り込み制御モード0、1

（d）割り込み制御モード2、3

【注】* リターン時には無視されます。

図 3.3　例外処理終了後のスタックの状態（例）

3.4 プログラム実行状態

CPU がプログラムを順次実行している状態です。

3.5 バス権解放状態

CPU 以外のバスマスタによるバス権要求に対して、バス権を解放した状態です。バス権解放状態では、CPU は、内部動作を除き、動作を停止します。

なお、CPU 以外のバスマスタには DMA コントローラ（DMAC）、およびデータトランスファコントローラ（DTC）などがあります。

詳細は、当該製品のハードウェアマニュアルを参照してください。

3.6 低消費電力状態

低消費電力状態には、CPU の動作を停止した状態と、CPU の動作を停止しない状態があります。CPU の動作を停止した低消費電力状態には、スリープモード、ソフトウェアスタンバイモード、ハードウェアスタンバイモードがあります。

また、その他の低消費電力状態には、中速モード、モジュールストップモードなどがあります。中速モードでは CPU およびそのほかのバスマスタが中速クロックで動作します。モジュールストップモードでは、モジュール単位で、CPU 以外のモジュールの動作を停止します。詳細は当該製品のハードウェアマニュアルを参照してください。

3.6.1 スリープモード

スリープモードには、SYSCR の SSBY ビットを 0 にクリアした状態で、SLEEP 命令を実行することによって遷移します。スリープモードでは、CPU の動作は SLEEP 命令実行直後で停止します。CPU の内部レジスタの内容は保持されます。

3.6.2 ソフトウェアスタンバイモード

ソフトウェアスタンバイモードには、SYSCR の SSBY ビットを 1 にセットした状態で、SLEEP 命令を実行することによって遷移します。ソフトウェアスタンバイモードでは、CPU およびクロックをはじめ MCU の全ての動作が停止します。内蔵周辺モジュールはリセット状態になりますが、規定の電圧が与えられている限り、CPU の内部レジスタの内容および内蔵 RAM の内容は保持されます。また、I/O ポートの状態も保持されます。

3.6.3 ハードウェアスタンバイモード

ハードウェアスタンバイモードには、$\overline{\text{STBY}}$ 端子を Low レベルにすることによって遷移します。ハードウェアスタンバイモードでは、CPU およびクロックをはじめ MCU の全ての動作が停止します。内蔵周辺モジュールはリセット状態になりますが、規定の電圧が与えられている限り、内蔵 RAM の内容は保持されます。

4. 基本動作タイミング

4.1 概要

本 CPU は、システムクロック（φ）を基準に動作しています。φの立ち上がりから次の立ち上がりまでの 1 単位をステートと呼びます。メモリサイクルまたはバスサイクルは、1、2 または 3 ステートで構成され、内蔵メモリ、内蔵周辺モジュール、または外部アドレス空間によってそれぞれ異なるアクセスを行います。詳細は当該製品のハードウェアマニュアルを参照してください。

4.2 内蔵メモリ（ROM、RAM）

内蔵メモリのアクセスは 1 ステートアクセスを行います。このとき、データバス幅は 16 ビットで、バイトおよびワードサイズアクセスが可能です。内蔵メモリアクセスサイクルを図 4.1 に、端子状態を図 4.2 に示します。

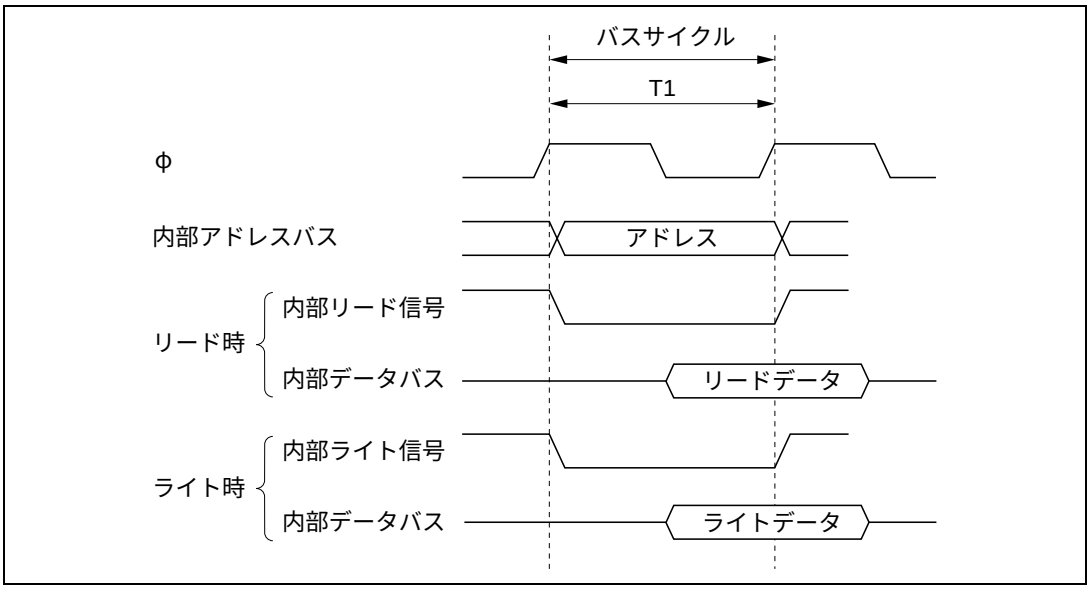


図 4.1　内蔵メモリアクセスサイクル

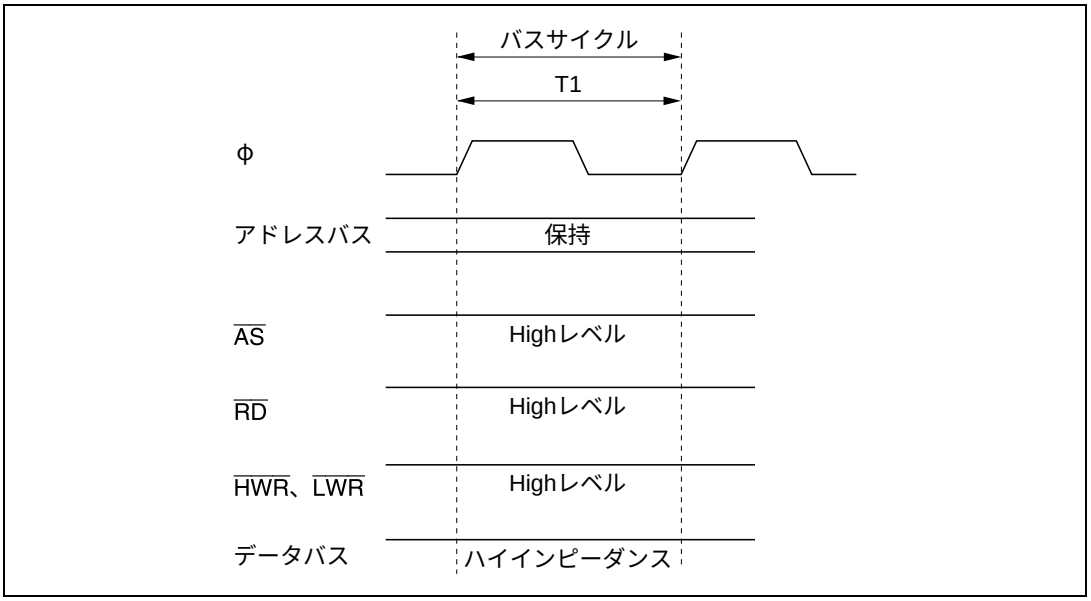


図 4.2　内蔵メモリアクセス時の端子状態

4.3 内蔵周辺モジュールアクセスタイミング

内蔵周辺モジュールのアクセスは 2 ステートで行われます。このとき、データバス幅は 8 ビットまたは 16 ビットで内部 I/O レジスタにより異なります。内蔵周辺モジュールアクセスタイミングを図 4.3、端子状態を図 4.4 に示します。

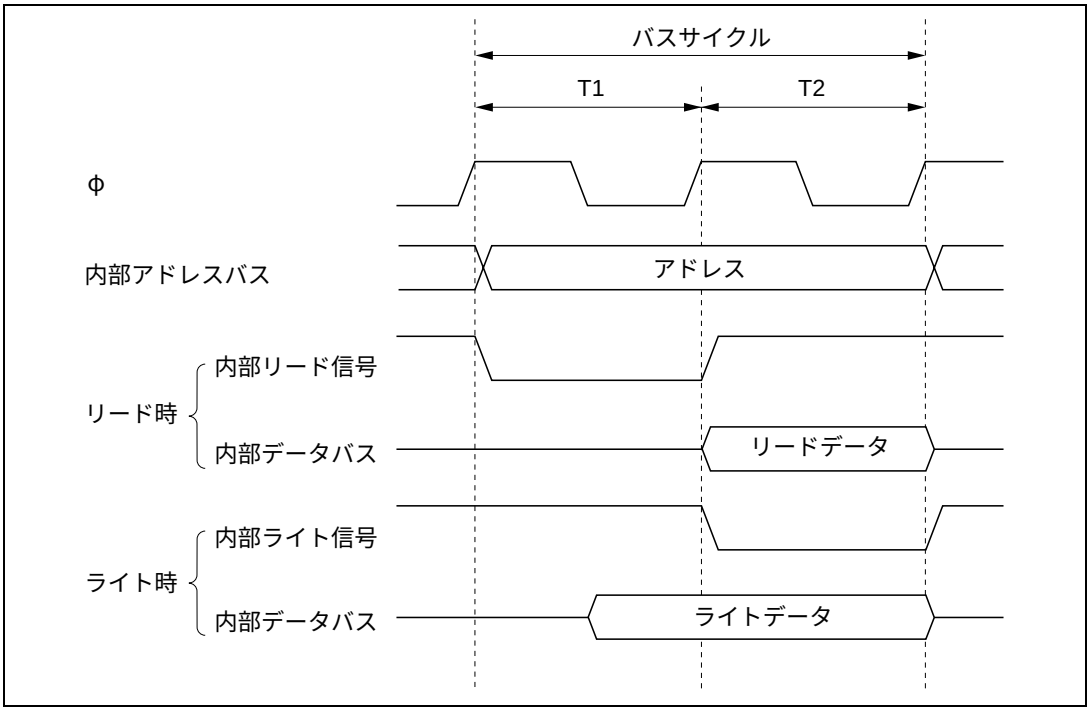


図 4.3 内蔵周辺モジュールアクセスタイミング

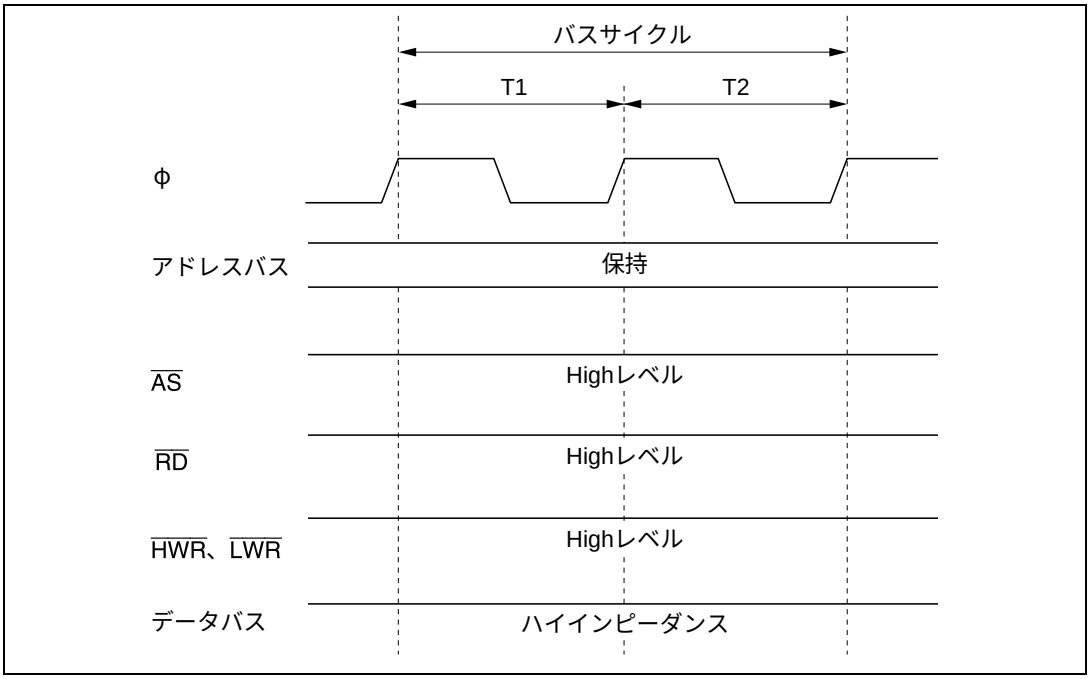


図 4.4　内蔵周辺モジュールアクセス時の端子状態

4.4　外部アドレス空間アクセスタイミング

外部アドレス空間のアクセスを行うときのデータバス幅は 8 ビットまたは 16 ビット、バスサイクルは 2 ステートまたは 3 ステートです。図 4.5 に 2 ステートアクセスおよび 3 ステートアクセスのリードタイミングを、また図 4.6 に 2 ステートアクセスおよび 3 ステートアクセスのライトタイミングを示します。3 ステートアクセスではウェイトステートを挿入することができます。詳細は当該製品のハードウェアマニュアルを参照してください。

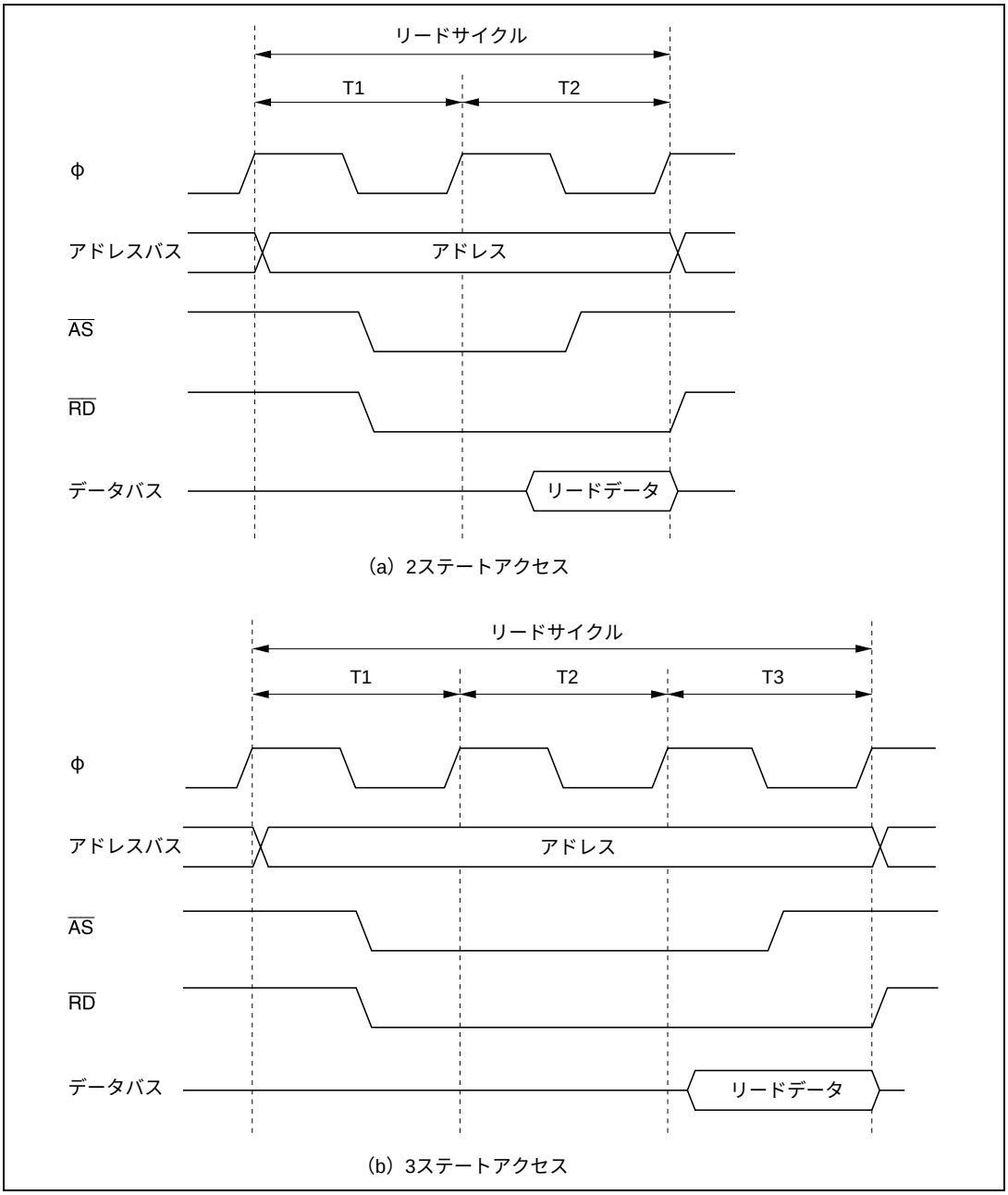


図 4.5 外部デバイスアクセスタイミング（リードタイミング）

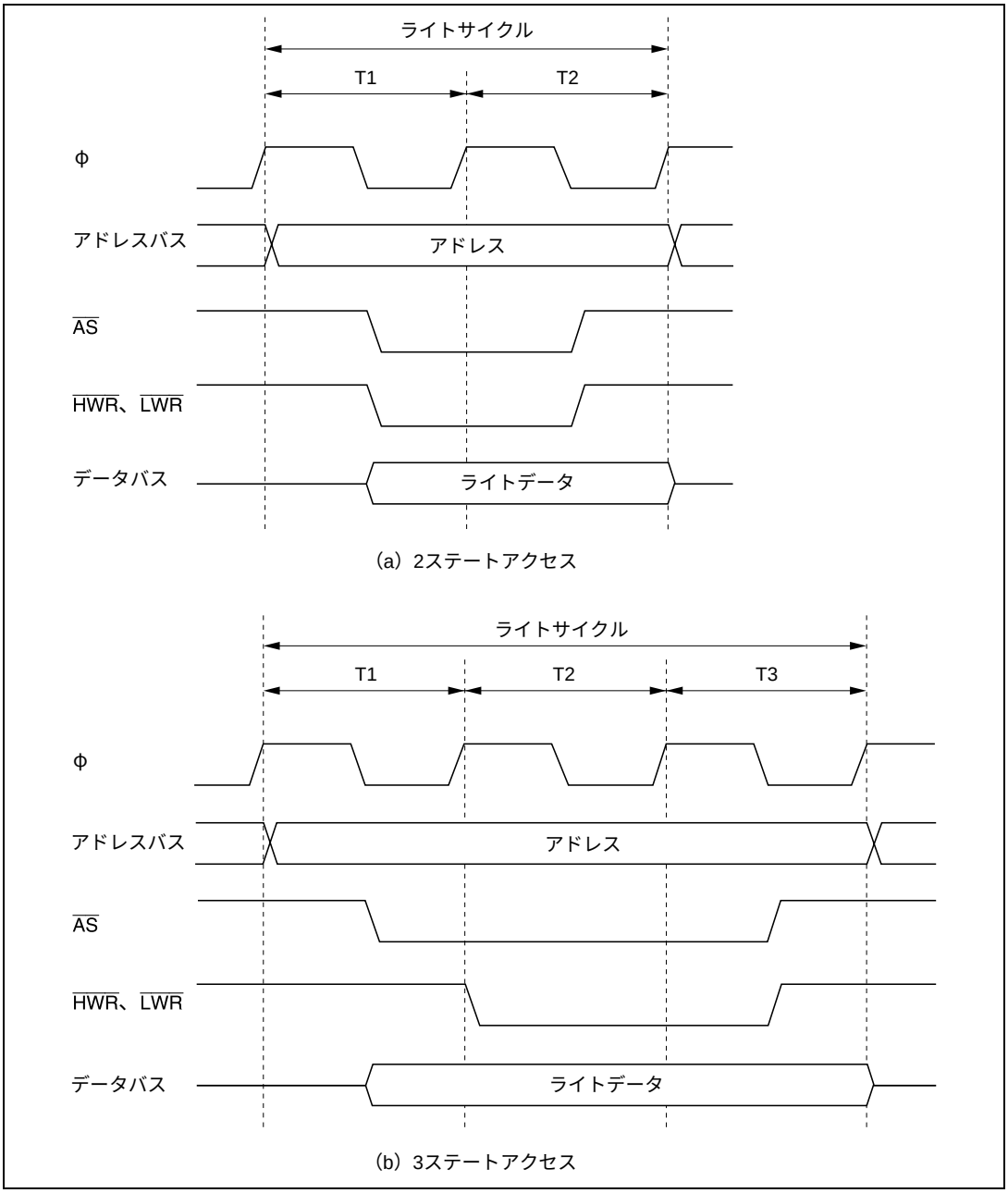


図 4.6 外部デバイスアクセスタイミング（ライトタイミング）

ルネサス16ビットシングルチップマイクロコンピュータ
ソフトウェアマニュアル
H8S/2600シリーズ、H8S/2000シリーズ

発行年月　1995 年 3 月　　第 1 版
　　2006 年 2 月 22 日　Rev.5.00
発　　行　株式会社ルネサス テクノロジ 営業企画統括部
　　〒100-0004 東京都千代田区大手町 2-6-2
編　　集　株式会社ルネサスソリューションズ
　　グローバルストラテジックコミュニケーション本部
　　カスタマサポート部

株式会社ルネサス テクノロジ　営業企画統括部　〒100-0004 東京都千代田区大手町2-6-2 日本ビル

営業お問合せ窓口
株式会社ルネサス販売

http://www.renesas.com

本社	〒100-0004	千代田区大手町2-6-2 (日本ビル)	(03) 5201-5350
京浜支社	〒212-0058	川崎市幸区鹿島田890-12 (新川崎三井ビル)	(044) 549-1662
西東京支社	〒190-0023	立川市柴崎町2-2-23 (第二高島ビル2F)	(042) 524-8701
東北支社	〒980-0013	仙台市青葉区花京院1-1-20 (花京院スクエア13F)	(022) 221-1351
いわき支店	〒970-8026	いわき市平小太郎町4-9 (平小太郎ビル)	(0246) 22-3222
茨城支店	〒312-0034	ひたちなか市堀口832-2 (日立システムプラザ勝田1F)	(029) 271-9411
新潟支店	〒950-0087	新潟市東大通1-4-2 (新潟三井物産ビル3F)	(025) 241-4361
松本支社	〒390-0815	松本市深志1-2-11 (昭和ビル7F)	(0263) 33-6622
中部支社	〒460-0008	名古屋市中区栄4-2-29 (名古屋広小路プレイス)	(052) 249-3330
関西支社	〒541-0044	大阪市中央区伏見町4-1-1 (明治安田生命大阪御堂筋ビル)	(06) 6233-9500
北陸支社	〒920-0031	金沢市広岡3-1-1 (金沢パークビル8F)	(076) 233-5980
広島支店	〒730-0036	広島市中区袋町5-25 (広島袋町ビルディング8F)	(082) 244-2570
鳥取支店	〒680-0822	鳥取市今町2-251 (日本生命鳥取駅前ビル)	(0857) 21-1915
九州支社	〒812-0011	福岡市博多区博多駅前2-17-1 (ヒロカネビル本館5F)	(092) 481-7695

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：コンタクトセンタ　E-Mail: csc@renesas.com

H8S/2600 シリーズ、H8S/2000 シリーズ
ソフトウェアマニュアル

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ09B0143-0500