JUMP-1 メモリ ・コマンド仕様 (ver. 2.1)

JUMP-1 設計グループ

Revision History

0.9	'95 07/15	H. Nakashima	第一次仕様 (一部未完) 作成
1.0	'95 07/27	H. Nakashima	第一次仕様作成
1.1	'95 09/14	H. Nakashima	主に Control Space 関する修正
2.0	'97 08/29	H. Nakashima	実装に基づく修正 (一部未完)
2.1	'97 09/10	H. Nakashima	実装に基づく修正

Revision History

ver. 1.0 1995.7.27 by H. Nakashima

- メモリ ・コマンド割付けの大幅修正。
- 未完の節の作成。

ver. 1.1 1995.9.14 by H. Nakashima

- 2 の OWCH の記述を全面変更。
- 2 の L2CCTL の記述に，MAXWM を追加。
- 3 の cachability に関する記述中の，同一アドレスに対する異なるメモリ ・コマンドによるアクセスの順序保存について，書込後に STBAR を行なわねばならない理由を詳述。
- 4.2(3) の INJLEV に関する記述に，値の設定に関する簡単な指針を追加。
- self-cleanup をコヒーレンス無制御の一種とし，その用途に関する記述を全面変更。
- 6 の q-write に関する記述に，プロセッサ間での q-write の逐次化に関する記述を追加。
- 7 の Block FIFO queue について，出力ポートの構成，dequeue の方法，出力ポート ・アクセス権のプロセッサ間移動，入力ポートの連続アクセスに関する制限，入力ポート ・アクセス権のプロセッサ間移動，bq-check による dequeue, についての記述を追加 / 修正。
- 8 の prefetch について，2 次キャッシュの遅延書込機能に関する記述を追加。また cflush について，CTYPE $\neq 3$ での動作に関する記述を修正。
- 9 を中心に，2 次キャッシュ資源に関する記述を大幅に修正。

ver. 2.0 1997.8.29 by H. Nakashima

- 最終的な実装に基づく修正。ただし control space などについての修正は未完。

ver. 2.1 1997.9.10 by H. Nakashima

- ver. 2.1 の未修正箇所を修正。

目次

1 概要 **1**

2 メモリ ・コマンドを決定する情報 **1**

3 メモリ ・コマンド一覧 **2**

4 コヒーレンス ・プロトコル **6**
4.1 Invalidate 系プロトコル 6
4.2 Update 系プロトコル 6
4.3 コヒーレンス無制御 7

5 I-structure **7**

6 FIFO queue **8**

7 Block FIFO queue **9**

8 プリフェッチとキャッシュ ・ラインのフラッシュ **10**

9 2 次キャッシュの内部資源 **11**

10 共有バスからの割込 **20**

11 アクセス例外 **20**
11.1 例外報告 20
11.2 例外に関する 2 次キャッシュ内部資源 21
11.3 例外情報の獲得 24

12 初期化 **24**

1 概要

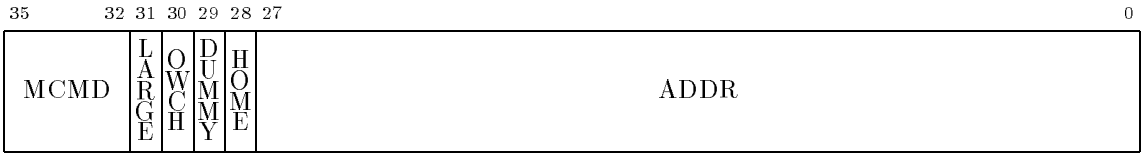
本仕様は，JUMP-1 の PE が実行する load/store などのメモリ・アクセス命令が，どのようなメモリ・コマンドとして解釈／実行されるかを定めたものである。

なお，本仕様は MBPL による実装に基づくものであり，本仕様の元となった『MBP』に関する仕様については資料 [1, 2] を参照されたい。

2 メモリ・コマンドを決定する情報

メモリ・コマンドは，以下の情報により決定される。

(1) 物理アドレス (PA[35:0])
SuparSparc の MMU が論理アドレスから変換して生成する，36 ビットの物理アドレスであり，以下のフォーマットによって解釈される。



- MCMD[3:0] (= PA[35:32])
  メモリ・コマンドの基本的な動作を定める。
- LARGE (= PA[31])
  0 であれば 4 KB/page のスモール・ページであることを，1 であれば 256 KB/page のラージ・ページであることを示す。このビットは，ページ・テーブルの構造と一致していなければならない[1]。
- OWCH (= PA[30])
  1 であれば，クラスタ・メモリへのアクセスのたびに MBPL のプログラムが実行される。従って通常は 0 であり，デバッグや評価データの収集の際にのみ 1 にする。
- DUMMY (= PA[29])
  0 であればページがクラスタ上に存在する通常のページであることを，1 であれば実体は存在しないダミー・ページであることを示す。なおダミー・ページであっても 1 次／2 次キャッシュにキャッシュすることができる。
- HOME (= PA[28])
  0 であれば他のクラスタが所有するページのコピーであることを，1 であれば自クラスタが所有するホーム・ページであることを示す。但し，DUMMY が 1 である場合にはホーム・ページであるか否かではなく，単にアドレスの一部であると解釈される。
- ADDR[27:0] (= PA[27:0])
  クラスタ・メモリ内のアドレス。MBPL ではクラスタ・メモリ容量が 32 MB であるので，クラスタに実体が存在するページについては ADDR[27:25] は 0 である。ADDR[27:0] が等しく MCMD が異なるアクセスは，同一の物理アドレスに対して行なわれる（物理的 aliasing と呼ぶ）。

(2) メモリ・アクセス命令の種類
ロード (LDxx), ストア (STxx), ロード／ストア (LDSTUBx, SWAPx) のいずれであるかにより，例えばロードとストアの意味が非対称である場合がある。

[1] MBPL では無視されるが，正しく設定しておくことが望ましい

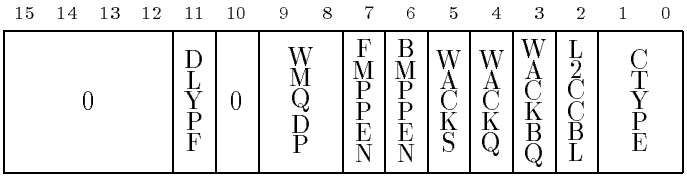
(3) メモリ ・アクセス幅
メモリ ・アクセスの幅，即ち byte (B), halfword (H), word (W), doubleword (D) により，MCMD の解釈が異なることがある。

(4) ストア ・データ
一部のメモリ ・コマンドでは，ストアされるデータがコマンドの意味を修飾することがある。

(5) キャッシュ可否
ページ ・テーブル ・エントリのビット C (PTE[7]) が定める，1 次キャッシュへのキャッシュ可否 (1 ならば可)。ほとんどの場合には 1 次キャッシュと 2 次キャッシュへのキャッシュ可否は等しいが，一部のメモリ ・コマンドでは 1 次キャッシュ不可であっても 2 次キャッシュにキャッシュすることができる (5 節，6 節参照)。

(6) Address Space Identifier (ASI)
LDA, STA など alternate space に対するアクセス命令の ASI フィールドが 0x02 であるとき，論理アドレスをそのままアドレスとする空間 (Control Space) 中に存在する，2 次キャッシュの内部資源をアクセスすることができる (9 節参照)。なお 2 次キャッシュの内部資源の一部は，通常のアドレス空間にもマップされており，ユーザモードでアクセスすることができる。

(7) 2 次キャッシュ制御レジスタ L2CCTL (ASI=0x02, VA=0x10000080)
Control Space 中の制御レジスタであり，以下のフォーマットを持つ。



メモリ ・コマンドに関連するビットは以下のものである。他のビットについては [4] 参照。

- WACKS = L2CCTL[5] (With Acknowledge for s-write)
  1 であれば，s-write (5 節参照) 操作完了を STBAR で確認することができる。通常は s-write の完了を確認する必要はなく，性能上は 0 であることが望ましい。
- WACKQ = L2CCTL[4] (With Acknowledge for q-write)
  1 であれば，q-write (6 節参照) 操作完了を STBAR で確認することができる。通常は q-write の完了を確認する必要はなく，性能上は 0 であることが望ましい。
- WACKBQ = L2CCTL[3] (With Acknowledge for bq-write)
  1 であれば，bq-write (7 節参照) 操作完了を STBAR で確認することができる。通常は bq-write の完了を確認する必要はなく，性能上は 0 であることが望ましい。
- L2CCBL = L2CCTL[2] (L2C Cachable for Special Commands)
  1 であれば，1 次キャッシュ不可とされた一部のメモリ ・コマンドを，2 次キャッシュ可で実行する (5 節，6 節参照)。
- CTYPE = L2CCTL[1:0] (Command Type)
  4 種類 (0 ~ 3) のメモリ ・コマンド ・セットのいずれを用いるかを定める。

3 メモリ ・コマンド一覧

表 1 にメモリ ・コマンドの一覧を示す。各欄について簡単に説明する。

表 1: メモリ ・コマンド一覧

MCMD	CTYPE	cache	protocol	Load				Store				Swap
				D	W	H	B	D	W	H	B	
0000	0–3	no		*	*	*	*	*	*	*	*	*
0001	0–3	no		—	—	—	RG	M	T	—	RG	M
0010	0–3	no(S)	read-inv	Q	Q	QC	Q	Q	Q	Q	Q	—
0011	0–3	no(S)	read-inv	P	P	QC	P	Q	Q	Q	Q	—
0100	0,1	no(S)	update-drct	S0	S0	F0	S0	S0	S0	PF	—	—
	2	no		S0	S0	F0	S0	S0	S0	PF	—	—
	3	y/n(S)	update-drct	S0	S0	S0	S0	S0	S0	PF	—	—
0101	0,1	no(S)	update-drct	S1	S1	F1	S1	S1	S1	PF	—	—
	2	no		S1	S1	F1	S1	S1	S1	PF	—	—
	3	y/n(S)	update-drct	S1	S1	S1	S1	S1	S1	PF	—	—
0110	0,1	no		P	P	QC	P	Q	Q	Q	Q	—
	2	no(S)	allread-up-drct	S0	S0	F0	S0	S0	S0	PF	—	—
	3	y/n(S)	update-drct	F0	F0	F0	F0	S0	S0	PF	—	—
0111	0,1	y/n	allread-up	*	*	*	*	*	*	PF	*	*
	2	no(S)	allread-up-drct	S1	S1	F1	S1	S1	S1	PF	—	—
	3	y/n(S)	update-drct	F1	F1	F1	F1	S1	S1	PF	—	—
1000	0–3	yes	read-inv	BQ	BQ	BQ	BQ	BQ	BQ	BQ	BQ	—
1001	0–2	y/n	invalidate	*	*	*	*	*	*	*	*	*
	3	no(Q)	read-inv	BC	BC	BC	BC	—	—	—	—	—
1010	0–3	y/n	invalidate	*	*	*	*	*	*	PF	*	*
1011	0–3	y/n	read-inv	*	*	*	*	*	*	PF	*	*
1100	0–3	y/n	no-coherent	*	*	*	*	*	*	*	*	*
	3	y/n	no-coherent	*	*	*	*	*	*	PF	*	*
1101	0–3	y/n	rdinv-up	*	*	*	*	*	*	PF	*	*
1110	0–3	y/n	update-drct	*	*	*	*	*	*	*	*	*
1111	0–3	y/n	update	*	*	*	*	*	*	*	*	*

(1) MCMD
MCMD[3:0] の値。

(2) CTYPE
L2CCTL.CTYPE[1:0] の値。

(3) cache
1 次 / 2 次のキャッシュ可否を示す。

- no ..... キャッシュ不可でなければならない。
- yes ..... キャッシュ可でなければならない。
- y/n ..... キャッシュ可否はどちらでも良く，指定に従って適切に処理される。
- y/n(S) .... キャッシュ可否はどちらでも良いが，可否それぞれについて s-read/s-write/f-read に関する例外的な解釈がなされる (5 節参照)。
- no(S) ..... キャッシュ不可でなければならないが，2 次キャッシュのみを例外的にキャッシュ可にすることができる (5, 6 節参照)。
- no(Q) .... キャッシュ不可でなければならないが，無条件に 2 次キャッシュのみ可と解釈される。

なお，キャッシュ可否のどちらかが定められたものに対して，それに反する指定を行なってアクセスすると，bus-error 例外が発生する (11 節参照)。

またキャッシュ可の二つのコマンド C_1 と C_2 に関し，同一アドレスに対して C_1 による書込を行なった後で C_2 による読出を行うと，書込結果が反映されていないことがある。つまり $A_1[31:0] = A_2[31:0]$, $A_1[35:32] = C_1$, $A_2[35:32] = C_2$ であるような二つの物理アドレス A_1, A_2 について，JUMP-1 のメモリ・システムはこれらを同一アドレスと解釈するが，SuperSparc 内部では異なるアドレスと解釈する。また SuperSparc の 1 次キャッシュはセット連想方式であるので，A_1 と A_2 は 1 次キャッシュの異なるエントリにマップされる可能性がある。従って A_2 が 1 次キャッシュ上に存在する状況では，SuperSparc のキャッシュ機構のみでは A_1 への書込が 1 次キャッシュ上の A_2 に反映されることはない。

この問題の対策として 2 次キャッシュ制御機構では，A_1 への書込を検知すると直ちに 1 次キャッシュ上の A_2 を無効化し，引き続く A_2 の読出が 2 次キャッシュに対して行なわれるようにしている。しかし，SuperSparc には書込をバッファリングする機構があるため，A_1 への書込が 2 次キャッシュに伝えられる以前に，A_2 の読出が実行されてしまうことがある。

そこで，同一アドレスに対して C_1 による書込を行なった後で C_2 による読出を行う場合には，書込と読出の間で STBAR (store barrier) を実行し，A_1 への書込が完了したこと (即ち A_2 が 1 次キャッシュから消滅したこと) を確認しなければならない。また 2 次キャッシュは一般に，1 次キャッシュ中に同一アドレスのコピーが一つしか存在しないように制御するので，同一アドレスに対して MCMD が異なる操作を互い違いに反復して行なうと，性能に悪影響を及ぼすことがある。

またキャッシュ不可のコマンドで書込んだ値を，キャッシュ可のコマンドで読出した場合，書込値が反映されていないことがある。このような操作が行なわれることは通常ないが，万一実施する場合には，書込む前にキャッシュをフラッシュしておかなければならない。

(4) protocol
二次キャッシュの特殊機能と，コヒーレンス・プロトコルを示す。

- s/*protocol* ..... I-structure の操作 s-read/f-read/s-write を二次キャッシュでサポートする (5 参照)。
- q/*protocol* ..... FIFO queue の操作 q-read/p-read/q-check/q-write を二次キャッシュでサポートする (6 節参照)。

- bq/*protocol* ..... Block FIFO queue の操作 bq-read/bq-write/bq-check を二次キャッシュでサポートする (7 節参照)。
- invalidate ..... invalidate プロトコル (4.1(1) 節参照)。
- read-inv ..... キャッシュ・リード・ミス時にも他のコピーを無効化する invalidate プロトコル (4.1(2) 節参照)。
- update ..... update プロトコル (4.2(1) 節参照)。
- update-drct ..... home page でのシリアライズを行なわない update プロトコル (4.2(2) 節参照)。
- rdinv-up ..... キャッシュ・ミス時に他のコピーを無効化する update プロトコル (4.2(4) 節参照)。
- allread-up ..... キャッシュ・ミス時に injection を行なう update プロトコル (4.2(3) 節参照)。
- no-coherent ..... コヒーレント管理を一切行なわない (4.3(1) 節参照)。

(5) Load/Store
load/store 命令とそのバイト長によって，どのような動作をするかを示す。

- * ..... 通常の load/store を，指定された cachability とコヒーレンス・プロトコルに基づいて行なう。
- S0/S1 ..... full/empty ビットの極性を empty = 0/1 として，I-structure 操作 s-read/s-write を行なう (5 節参照)。
- F0/F1 ..... full/empty ビットの極性を empty = 0/1 として，I-structure 操作 f-read を行なう (5 節参照)。
- Q ..... FIFO queue 操作 q-read/q-write を行なう (6 節参照)。
- P ..... FIFO queue 操作 p-read を行なう (6 節参照)。
- QC ..... FIFO queue 操作 q-check を行なう (6 節参照)。
- BQ ..... Block FIFO 操作 bq-read/bq-write を行なう (7 節参照)。
- BC ..... Block FIFO 操作 bq-check を行なう。
- M ..... MBPL 起動要求 m-write を行なう (資料 [1, 2, 3] 参照)。
- T ..... Memory Based Signal の発行要求 t-write を行なう (資料 [1, 2, 3] 参照)。
- RG ..... 二次キャッシュのレジスタをアクセスする (9 節参照)。
- PF ..... プリフェッチまたはキャッシュ・ライン・フラッシュを行なう (8 節参照)。
- — ..... 指定されたバイト長での load/store は禁止されており，bus-error 例外が発生する (11 節参照)。

(6) Swap
swap 系の命令 (SWAPx, LDSTUBx) による動作を示す。

- * ..... 通常の swap 系の動作を，指定された cachability とコヒーレンス・プロトコルに基づいて行なう。
- M ..... 不可分メモリ・アクセスを行なう (資料 [1, 2, 3] 参照)。なお，この操作は SWAPx にのみ許され，LDSTUBx を行なうと bus-error 例外が発生する (11 節参照)。
- — ..... 操作が禁止されており，bus-error 例外が発生する (11 節参照)。

4　コヒーレンス・プロトコル

JUMP-1 のキャッシュ・コヒーレンスは，代表的な invalidate と update プロトコルに加え，それらを変形した数多くのプロトコルによって管理することができる。従って，アプリケーションやデータの性質に応じて，最適なプロトコルを選択して使用することができる。

4.1　Invalidate 系プロトコル

(1)　invalidate

invalidate プロトコルでは，共有データに書込を行なう前に，全てのコピーが無効化される。従って，あるプロセッサが更新した値を他のプロセッサが初めて読出す際に，必ずキャッシュミスが起こり，プロセッサ / クラスタ間でのデータ転送が行なわれるという欠点がある。

その反面，あるキャッシュ・ラインを連続的に更新する場合でも，プロセッサ間 / クラスタ間の通信は一度だけで済む。また必要最小限のコピーしか存在しないことも，メリットととらえることができる。

(2)　read-inv

読出でキャッシュ・ミスした際にも，全てのコピーを無効化する機能を付加した invalidate プロトコルである。例えば，参照と更新がペアになっているような共有データに対して read-inv プロトコルを用いると，参照時の通信（データ読出）と，書込時の通信（コピー無効化）を，1 回で済ませることができる。

また read-inv プロトコルは，FIFO queue のように参照者が 1 つでなければならない機能の実現にも用いられる。

4.2　Update 系プロトコル

(1)　update

update プロトコルでは，書込の際にコピーを保持している全てのキャッシュについて，更新後のデータが送られる。従って，あるプロセッサが更新した値を，他のプロセッサは自分のキャッシュから読むことができる。

その反面，全てのコピーに対して（正確にはコピーを保持していると想定されるキャッシュに対して）書込のたびにデータが送られるので，プロセッサ間 / クラスタ間の通信が多発する傾向にある。但し，時間的に近接した同一キャッシュ・ラインへの書込は，2 次キャッシュによってマージされるので，多くの場合には通信回数は最小化される。なおこのマージのためのバッファは，2 次キャッシュ内部の（仮想的）レジスタ WBFLSH へ任意の値を書くことにより，フラッシュされて確実にプロセッサ間 / クラスタ間の書込要求が出される（9 節参照）。

また 2 次キャッシュの容量が大きいため，参照しなくなったキャッシュ・コピーが残存し，それに対しても無駄な更新要求が送られると言う問題もある。

(2)　update-drct

update プロトコル（及び全ての invalidate 系プロトコル）では，クラスタ間で共有されているデータへの書込要求が，一度ホーム・クラスタへ送られて順序付けられる。この結果，同一アドレスに対して複数のプロセッサが同期を取らずに書込を行なった場合にも，全てのコピーの値が同一であることが保証される[2]。

しかし，このような同期なしの書込競合は通常のプログラムでは行なわれず，ホーム・クラスタでの順序づけはある意味で無駄である。そこで update-drct プロトコルでは，書込を行なったプロセッサが所属するクラスタから，各コピーに対して要求が直接送出される。

[2] 遷移的な矛盾は観測されうる。

(3) allread-up

キャッシュ・ミスが起こると，通常はミスしたプロセッサに関するキャッシュにだけ，データが送られる。allread-up プロトコルではこれを拡張し，ミスしたプロセッサと「近縁関係」にあるプロセッサやクラスタに対して，データを「注入」する injection を行なう。従って，あるプロセッサと同じデータが必要な他のプロセッサは，キャッシュ・ミスを起こすことなくデータを入手できる可能性が高くなる。

この injection に関する戦略として，二つの要素が考えられる。一つは「近縁関係」プロセッサ / クラスタの選択であり，これは MBPL のプログラム設計に依存する。但し，ミスしたプロセッサと同じクラスタに属するプロセッサは，近縁関係にあると考えて良い。

もう一つの要素は，実際に injection を行なうか否かであり，例えば対応するキャッシュ・ラインに既に他のデータがある場合にも実施するか否かの判断は，必ずしも明白ではない。そこで 2 次キャッシュに関しては，injection の実行条件を定めるレジスタ INJLEVが設けられ（9 節参照），その値によって injection の可否が以下のように判断される。

- INJLEV = 0 ..... 該当するラインが無効であり，かつ直前に有効であった時には同じアドレスのデータが存在した場合に実行。
- INJLEV = 1 ..... 該当するラインが無効であれば実行。
- INJLEV = 2 ..... 該当するラインが無効であるか，既に存在するデータをクラスタ・メモリへ書き戻す必要がない場合に実行。
- INJLEV = 3 ..... 無条件に実行。

即ち，INJLEVの値が大きいほど injection が行なわれる確率が高くなる。従って，他のプロセッサがラインをアクセスする確度が高ければ，INJLEVを大きくしてラインがアクセス時にキャッシュされているようにすればよい。逆に確度が低ければ，injection によるオーバヘッドを避けるために，INJLEVを小さくすればよい。

なお，クラスタ・メモリ（3 次キャッシュ）に関する injection 条件は MBPL のプログラム設計に依存する。また，injection は allread-up 以外のプロトコルを使用している場合にも，MBPL のプログラムにより陽に実行可能であるが，その場合も 2 次キャッシュでの実行条件は INJLEVが定める。

(4) rdinv-up

Invalidate 系の read-inv と同様，キャッシュ・ミス時（読出 / 書込とも）に他のコピーを無効化する機能を付加した update プロトコルである。あるデータを共有する全てのプロセッサがこのプロトコルを使用するのは当然無意味であり，一部のプロセッサのみが使用したり，プログラムの特定の局面で使用したりするのが普通である。

4.3 コヒーレンス無制御

(1) no-coherent

no-coherent プロトコル（という表現は不適切であるが）では，キャッシュ・コヒーレンスに関する操作を一切行なわない。従って共有データに対して常にこの方式を使うのは当然無意味であり，局面に応じて限定的かつ適切に行なわねばならない。なお共有データを一時的に書換える場合など，他のコピーに更新データを反映させる必要がない場合には，no-coherent を用いると無駄な通信を抑止することができる。

5 I-structure

JUMP-1 のメモリの全ての doubleword には，同期ビットと呼ばれるビットが付加されており，データの有効性を表現することができる。I-structure 操作のためのメモリ・コマンド s-read/f-read/s-write は，この同期ビットの状態に応じた読出や，同期ビットの変更を伴う書込を行なう。

同期ビットは，0 を empty (データ無効) とみなすか，1 を empty とみなすか，どちらにでも用いることができる。また s-read/f-read/s-write は doubleword 以外のアクセス幅でも使用できるが，この場合には例えば対象の word が属する doubleword に付加された同期ビットが処理の対象となる。

(1) s-read

s-read は，データが full (empty でない) であればそのデータを返し，empty であれば undefined-error 例外を発生する。SuperSparc を用いた通常のシステムでは，この例外の発生は深刻な例外事象の発生を意味するが，JUMP-1 では続行可能な例外として取り扱われる。例えば，この例外処理ルーチンでスレッドの実行を中断し，full になった時に再開できるように s-enqueue (資料 [1, 2] 参照) によって再開アドレスを登録するなどの処理が行なえる。

なおこの節の (4) で述べるように，s-read が (1 次キャッシュに) キャッシュ可であるページに対して発行された場合には，empty であっても例外を発生させず，単に 0 を返す。

(2) f-read

f-read は，データが empty であれば 0 を返し，full であればデータの値に関わらず -1 (unsigned の場合には $2^n - 1$) を返す。

(3) s-write

s-write は，同期ビットを full にしながらデータを書込む。その際に書込を待っているスレッドが s-enqueue によって登録されていれば，そのスレッドの再開等の処理が MBPL によって行なわれる。

なお s-write がキャッシュ可であり，かつダミーであるようなページに対して発行された場合には，(通常は) 実際にキャッシュにデータがアロケートされない。この特例は，キャッシュ可のページへの書込が，キャッシュ不可のページへの書込よりも高速であることを利用するために設けられている。

また L2CCTL のフラグ WACKS が 0 であれば，s-write の完了を確認するための処理が一切行なわれない。s-write による書込を s-read で読出す限り，s-write の完了を確認する必要はなく，WACKS は 0 であるのが普通であり，かつ完了確認処理が省かれるので性能も良い。一方 WACKS が 1 であれば，s-write の完了を STBAR で確認することができる。

(4) キャッシュ可否の特例

I-Structure 操作のコマンドは，キャッシュ不可のページに対して発行されても，2 次キャッシュに関してはキャッシュ可にすることができる。この選択は Control Space 中の制御レジスタ L2CCTL のフラグ L2CCBL により定められ，0 であれば 2 次キャッシュについても不可，1 であれば 2 次キャッシュについては可となる。

また 1 次キャッシュ可のページに対して発行された場合には，前述のように s-read が empty の場合に 0 を返すという変更がある。

6 FIFO queue

JUMP-1 の各 doubleword は，doubleword を要素とする FIFO queue とすることができる。この queue の生成などは MBPL のプログラムを実行することにより行なうが，queue へのアクセスは，q-read/p-read/q-check/q-write によって行なうことができる。

なおこれらのコマンドを，doubleword 以外の幅でアクセスすることも可能であるが，それらは doubleword 単位の操作をデータだけを部分的にして行なったものと解釈される。例えば q-read を上位の word と下位の word に対して行なうと，queue の先頭から二つの doubleword が取り出され，前のものの上位と，後のものの下位が得られる (先頭の doubleword の上位と下位が得られるのではない)。

また queue が存在する 32 B ブロックに対して，queue 操作以外のコマンドを発行した場合，結果は保証されない。従って，queue を含む物理ページに対して queue 操作以外のコマンドを行なうページを物理的 aliasing するのは避けるべきである。

(1) q-read

queue の先頭要素 (出力端の要素) を返し，その要素を queue から取り除く。もし queue に要素が一つもなければ，(s-read と同じように) undefined-error 例外が発生する。例外処理ルーチンの中でスレッド実行を中断して，要素が得られた時点で再開するには，q-enqueue (資料 [1, 2] 参照) を使用する。

(2) p-read

queue の先頭要素 (出力端の要素) を返し，その要素を queue から取り除く。もし queue に要素が一つもなければ，(例外を発生せずに) 0 を返す。

(3) q-check

queue に要素が一つもなければ 0 を，一つでもあれば -1 を (unsigned であれば $2^n - 1$) 返す。queue の状態は変更しない。

(4) q-write

queue の末尾 (入力端) に，データを付加する。

L2CCTL のフラグ WACKQ が 0 であれば，q-write の完了を確認するための処理が一切行なわれない。q-write による書込を q-read などで読出す限り，q-write の完了を確認する必要はなく，WACKQ は 0 であるのが普通であり，かつ完了確認処理が省かれるので性能も良い。一方 WACKQ が 1 であれば，q-write の完了を STBAR で確認することができる。

あるプロセッサが q-write を実行中に，別のプロセッサが同じアドレスに対する q-write を行なうと，enqueue される順序は非決定的である。従って，プロセス・マイグレーションなどの際に queue のアクセス権を移動する場合には，q-write の完了を確認する必要があるが，WACKQ が 0 である時には STBAR によって確認することはできない。このために，q-write の完了を確認するための m-write が用意される [3]。

(5) キャッシュ可否の特例

FIFO queue 操作のコマンドは，キャッシュ不可のページに対して発行されなければならないが，2 次キャッシュに関してはキャッシュ可にすることができる。この選択は Control Space 中の制御レジスタ L2CCTL のフラグ L2CCBL により定められ，0 であれば 2 次キャッシュについても不可，1 であれば 2 次キャッシュについては可となる。キャッシュ可とすると，queue の先頭の要素を 2 次キャッシュにプリフェッチする機構が働く。

7 Block FIFO queue

前述の FIFO queue とは別に，32 B ブロックを要素とする Block FIFO queue を定義することができる。この Block FIFO queue は 1 次キャッシュまでキャッシュ可であり (逆にキャッシュ不可であってはならない)，またプリフェッチ機構も doubleword の FIFO queue よりも強力である。

Block FIFO queue の生成などは MBPL のプログラムを実行することにより行なうが，queue へのアクセスは bq-read/bq-write/bq-check によって行なうことができる。queue には先頭要素を取り出すための読出ポートと，末尾に要素を付加するための複数の書込ポートとがあり，読出ポートと書込ポートは別のアドレスに割り当てられる。読出ポートは 128 B 境界にアラインされた 128 B のブロックを占有し，4 ブロックのプリフェッチ用循環バッファとして用いられる。また書込ポートは 32 B 境界にアラインされた 32 B のブロックを占有する。

読出ポートが存在する 128 B ブロックや書込ポートが存在する 32 B ブロックに対して，queue 操作以外のコマンドを発行した場合，結果は保証されない。従って，Block FIFO queue の読出 / 書込ポートを含む物理ページに対して，queue 操作以外のコマンドを行なうページを物理的 aliasing するのは避けるべきである。

(1) bq-read

queue の先頭ブロックの取り出しと，そのブロック中の個々のデータの読出を行なう。bq-read が dequeue を伴うか否かは，そのアドレス A と，直前に行なった bq-read/bq-check のアドレス A'との関係で定まる。即ち A と A'が読出ポートの同じ 32 B ブロックに含まれていれば (i.e. $A[7{:}6] = A'[7{:}6]$)，単なる読出である。一方，$A[7{:}6] = A'[7{:}6] + 1 \ (mod 4)$ であれば，後に行なわれた bq-read によって dequeue が行なわれる。その際，queue に要素が一つもなければ undefined-error 例外が発生する。

読出ポートは通常一つのプロセッサに占有されるが，プロセス・マイグレーションなどによってアクセス権がプロセッサを移動しても構わない。しかし複数のプロセッサが同時にアクセス権を所有してはならず，あるプロセッサが bq-read/bq-check を実行中に，別のプロセッサが同じ queue に対して bq-read/bq-check を行なうと，bus-error となることがある。また読出ポートのブロックのアクセス順が不正である場合にも，bus-error となることがある。

(2) bq-write

queue の末尾に付加すべき要素中の個々のデータの書込と，実際の付加とを行なう。bq-write が enqueue を伴うか否かは，書込ポートの 32 B ブロック中のアドレスにより定まる。即ちアドレスがブロックの末尾バイトを含む場合には，enqueue が行なわれる。

一つの書込ポートに対する，末尾バイトの書込による enqueue 操作のための store 命令 S_1 と，次に付加するブロックを生成するための store 命令 S_2 は，その書込ポート以外のアドレスへの store または STBAR により分離されていなければならない。この制限を最も簡単，かつ大きな性能ダメージをもたらすことなくクリアするには，S_1 の直後に 2 次キャッシュの内部レジスタ L2DMY へ任意の値を書込む STH を挿入すれば良い (9 節参照)。なおこの制限に違反し，かつその違反が致命的である場合には bus-error 例外が発生する。

L2CCTL のフラグ WACKBQ が 0 であれば，q-write の完了を確認するための処理が一切行なわれない。bq-write による書込を bq-read で読出す限り，bq-write の完了を確認する必要はなく，WACKBQ は 0 であるのが普通であり，かつ完了確認処理が省かれるので性能も良い。一方 WACKBQ が 1 であれば，bq-write の完了を STBAR で確認することができる。

書込ポートは通常一つのプロセッサに占有されるが，プロセス・マイグレーションなどによってアクセス権がプロセッサを移動しても構わない。しかし複数のプロセッサが同時にアクセス権を所有してはならず，あるプロセッサが bq-write を実行中に，別のプロセッサが同じ queue に対して bq-write を行なうと，結果を予期することができない。従って，アクセス権を移動する際には bq-write の完了を確認する必要があるが，WACKBQ が 0 である時には STBAR によって確認することはできない。このために，bq-write の完了を確認するための m-write が用意される [3]。

(3) bq-check

queue に要素が一つもなければ 0 を，一つでもあれば -1 を (unsigned であれば $2^n - 1$) 返す。bq-read と同様の規則で，dequeue 操作を伴う。

8 プリフェッチとキャッシュ・ラインのフラッシュ

あらかじめデータを 2 次キャッシュに読み込んでおく prefetch と，キャッシュ・ラインの無効化を伴う (あるいは伴わない) ホーム・ページへの書き戻しを行なう cflush は，実際には書込を伴わない halfword の store によって行なわれる。

二つの操作の区別，及び操作を 2 次キャッシュと MBPL のどちらが行なうかは，ストアされるデータによって以下のように定まる。

- PC (= DATA[15]) ..... PC = 0 であれば prefetch を，PC = 1 であれば cflush を行なう。
- PARAM (= DATA[14:0]) ..... PARAM[14:8] = 0 であれば，ADDR で定まるキャッシュ・ラインの 2 次キャッシュへのプリフェッチ (prefetch)，あるいはキャッシュ・ラインの無効化 (cflush) を行なう。

  prefetch については，獲得したラインを直ちに 2 次キャッシュに書込まず，そのラインをプロセッサがアクセスするまでバッファに蓄えておく機能がある。この機能により，プリフェッチしたラインの読出のための 2 次キャッシュ・アクセスを省略することができる。この機能は L2CCTL のフラグが 1 の時に有効である[3]。

  cflush については，プロセッサが所有する 1 次 / 2 次キャッシュの無効化と，必要に応じたクラスタ・メモリへの書き戻しだけを行なう。

  PARAM[14:8] $\neq$ 0 であれば，prefetch/cflush 操作は MBPL が行なう。その際の PARAM の解釈は資料 [3] を参照。

なお，prefetch/cflush には以下の制限があるので，注意すること。

(a) ADDR は doubleword 境界にアラインされていなければならない。即ち ADDR の下位 3 ビットは 0 でなければならない。

(b) prefetch/cflush が実行される以前に最後に実行された store 系命令 S_Pと，prefetch/cflush が実行された後に最初に実行される store 系命令 S_Sは，ともに **prefetch/cflush** とは異なる **32 B** ブロックをアクセスするものでなければならない。但し，prefetch/cflush と S_P/S_Sの間に STBAR があれば S_P/S_Sに関する制限はない。

この制限を最も簡単，かつ大きな性能ダメージをもたらすことなくクリアするには，prefetch/cflush の直前と直後で，2 次キャッシュの内部レジスタ L2DMY へ任意の値を書込む STH を挿入すれば良い (9 節参照)。

なお，制限 (a) に違反した場合は bus-error 例外が発生し，制限 (b) に違反した場合には当該違反が致命的である (誤った動作をする) 場合に限って bus-error 例外が発生する。

9 2 次キャッシュの内部資源

2 次キャッシュの内部資源は，表 2 に示すように，32 ビットのアドレス空間中にマップされている。表の cat は，以下に示す資源のアクセスに関する制限を示したものである。

- (A) ..... MCMD = 0001 の LDH/STH によるアクセスを，ユーザに開放することを想定したもの。
- (B) ..... MCMD = 0001 の LDH/STH によるアクセスを，ユーザに開放しないことを想定したもの。但し OS の判断によって，開放することもできる。
- (Cd,Ci,Cx) .. LDA/STA など，Control Space をアクセスできる特権命令によってのみアクセスできるもの。ASI は 0x02 であり，VA は表 2 に示したアドレスである。Cd に属する資源は LDA/STA で直接ロード / ストアが可能である。Ci に属する資源に (LDA ではなく) STA を行なうと，資源の値が Cd に属するレジスタ CSADR に格納される。また Ci に属する資源のアドレスに 0x10000000 を加えたものに対して STA を行なうと，CSADR の値が資源に格納される。Cx に属する資源のアクセスには特殊な手順が必要である。

表に示すように，2 次キャッシュの内部資源には 8 B (doubleword) ごとにアドレスが付されており，doubleword 境界以外のアドレスを指定した場合の結果は結果は保証されない。定義されていないアドレスへの書込は無効であり，読出したデータの値は不定である。

[3] bq-read/bq-check でのプリフェッチでも，この機能が使用される。

表 2: 2 次キ ャッシ ュ内部資源一覧 (1/5)

cat	ADDR	r/w	width	name	note
A	00000000	w	0	EBAPRV	EB Approve
	00000008			(reserved)	
	00000010	w	0	EBPREQ	EB Pre-Request
	00000018			(reserved)	
	00000020			(reserved)	
	00000028			(reserved)	
	00000030	r	1	EBRREQ	EB Real-Request
	00000038			(reserved)	
	00000040	w	0	WBFLSH	Write Buffer Flush
	00000048			(reserved)	
	00000050			(reserved)	
	00000058			(reserved)	
	00000060	w	0	L2DMY	Dummy Register
	00000068			(reserved)	
	⋮				
	00000FF8			(reserved)	
B	00001000	r/w	8	EBCAP	EB Approve Counter
	00001008			(reserved)	
	00001010	r/w	8	EBCPR	EB Pre-Request Counter
	00001018			(reserved)	
	00001020	r/w	8	EBCCP	EB Completed Request Counter
	00001028			(reserved)	
	00001030	w	0	EBCLR	EB Clear
	00001038			(reserved)	
	00001040	r/w	2	INJLEV	Injection Level
	00001048			(reserved)	
	00001050	r	2	PEID	Processor Element ID
	00001058			(reserved)	
	00001060			(reserved)	
	⋮				
	00001FF8			(reserved)	
	00002000			(reserved)	
	⋮				
	0FFFFFF8			(reserved)	

表 2: 2 次キャッシュ内部資源一覧 (2/5)

cat	ADDR	r/w	width	name	note
Cd	10000000	r/w	8	CSADR[31:24]	Cont Space Access Data
	10000008	r/w	8	CSADR[23:16]	
	10000010	r/w	8	CSADR[15:8]	
	10000018	r/w	8	CSADR[7:0]	
	10000020			(reserved)	
	⋮				
	10000078			(reserved)	
	10000080	r	4	BMPRDR[11:8]	Backmap Read Data
	10000088	r	8	BMPRDR[7:0]	
	10000090	r/w	4	L2CCTL[11:8]	L2C Control Reg
	10000098	r/w	8	L2CCTL[7:0]	
	100000A0	r/w	2	L2CFRZ	L2C Freeze
	100000A8			(reserved)	
	100000B0	r	1	L2CERR	L2C Error
	100000B8			(reserved)	
	100000C0	r/w	6	ERRST[13:8]	Access Error Status
	100000C8	r/w	8	ERRST[7:0]	
	100000D0	r	7	FMPEAR[14:8]	Frotmap Error Address
	100000D8	r	8	FMPEAR[7:0]	
	100000E0			(reserved)	
	100000E8			(reserved)	
	100000F0	r	8	INTST	Bus Interrupt Status
	100000F8			(reserved)	
	10000100	w	8	INTRST	Reset Interrupt Status
	10000108			(reserved)	
	10000110	w	8	INTSET	Set Interrupt Status
	10000118			(reserved)	
	10000120	w	2	SNDINV	SNDBUF invalidate
	10000128			(reserved)	
	10000130	w	2	RCVINV	RCVBUF invalidate
	10000138			(reserved)	
	10000140	w	1	EMMTRP	Emergency Trap to pseudo MBP
	10000148			(reserved)	
	10000150	w	1	L2CRDY	Make CPU accessible to L2C
	10000158			(reserved)	
	10000160	w	4	ECSAID	External Cont Space Access ID
	10000168			(reserved)	
	10000170			(reserved)	
	⋮				
	3FFFFFF8			(reserved)	

表 2: 2 次キ ャッシ ュ内部資源一覧 (3/5)

cat	ADDR	r/w	width	name	note
Cd	40000000	r/w	64	DARRAY[0]	L2C Data Array
	⋮				
	400FFFF8	r/w	64	DARRAY[0xFFFF8]	
	40100000			(reserved)	
	⋮				
	4FFFFFF8			(reserved)	
	50000000			(reserved)	
	⋮				
	5FFFFFF8			(reserved)	
	60000000	w	0	INV[0]	Invalidate Cache Entry
	60000020	w	0	INV[1]	
	⋮				
	600FFFE0	w	0	INV[0x7FFF]	
	60100000			(reserved)	
	⋮				
	6FFFFFF8			(reserved)	
	70000000			(reserved)	
	⋮				
	7FFFFFF8			(reserved)	

表 2: 2 次キ ャッシ ュ内部資源一覧 (4/5)

cat	ADDR	r/w	width	name	note
Ci	80000000			(reserved)	
	⋮				
	80000038			(reserved)	
	80000040	r/w	32	RCVDIR[0][63:32]	RCVBUF directory
	⋮				
	80000058	r/w	32	RCVDIR[3][63:32]	
	80000060	r/w	32	RCVDIR[0][31:0]	
	⋮				
	80000078	r/w	32	RCVDIR[3][31:0]	
	80000080	r/w	32	SNDDIRA[0][63:32]	SNDBUF directory (part-A)
	⋮				
	80000098	r/w	32	SNDDIRA[3][63:32]	
	800000A0	r/w	32	SNDDIRA[0][31:0]	
	⋮				
	800000B8	r/w	32	SNDDIRA[3][31:0]	
	800000C0	r/w	8	SNDDIRB[0][39:32]	SNDBUF directory (part-B)
	⋮				
	800000D8	r/w	8	SNDDIRB[3][39:32]	
	800000E0	r/w	32	SNDDIRB[0][31:0]	
	⋮				
	800000F8	r/w	32	SNDDIRB[3][31:0]	
	80000100	r/w	14	REQRCVQ	Request Receive Queue
	80000108	r/w	14	REPRCVQ	Reply Receive Queue
	80000110	r/w	14	REQSNDQ	Request Send Queue
	80000118	r/w	14	REPSNDQ	Reply Send Queue
	80000120			(reserved)	
	80000128	r/w	32	RETBUF	Retry Buffer
	80000130	r/w	4	SDWORK[35:32]	SNDDIR Work Register
	80000138	r/w	32	SDWORK[31:0]	
	80000140	r/w	32	TOCTR	Timeout Counter
	80000148	r/w	32	TOLMT	Timeout Limit
	80000150	r/w	16	WMCTR	Write Merge Timeout Counter
	80000158	r/w	16	WMLMT	Write Merge Timeout Limit
	80000160	r/w	24	PFCTR	Prefetch Counters
	80000168	r/w	12	PFLMT	Prefetch Limit
	80000170			(reserved)	
	80000178			(reserved)	
	80000180	r/w	19	MESPATB	Measurement Pattern for Bus Event
	80000188	r/w	19	MESMSKB	Mask for MESPATB
	80000190	r/w	32	MESCTRB	Counter for MESPATB
	80000198			(reserved)	
	800001A0	r/w	16	MESPATP	Measurement Pattern for Proc Event
	800001A8	r/w	16	MESMSKP	Mask for MESPATP
	800001B0	r/w	32	MESCTRP	Counter for MESPATP
	800001B8			(reserved)	
	800001C0			(reserved)	
	⋮				
	8FFFFFF8			(reserved)	

表 2: 2 次キ ャッシ ュ内部資源一覧 (5/5)

cat	ADDR	r/w	width	name	note
Ci	90000000				for store 80000000–8FFFFFF8
	⋮				
	9FFFFFF8				
	A0000000	r/w	28	FMP[0]	L2C Cache Tag Front Map
	A0000020	r/w	28	FMP[1]	
	⋮				
	A00FFFE0	r/w	28	FMP[0x7FFF]	
	A0100000			(reserved)	
	⋮				
	AFFFFFF8			(reserved)	
	B0000000				for store A0000000–AFFFFFF8
	⋮				
	BFFFFFF8				
	C0000000			(reserved)	
	⋮				
	DFFFFFF8			(reserved)	
Cx	E0000000	w	0	RBMP[0]	Read Cache Tag Back Map
	E0000020	w	0	RBMP[1]	
	⋮				
	E01FFFE0	w	0	RBMP[0x7FFF]	
	E0200000			(reserved)	
	⋮				
	EFFFFFF8			(reserved)	
	F0000000	w	0	WBMP[0](0)	Write 0 to L2C Cache Tag Back Map
	F0000020	w	0	WBMP[1](0)	
	⋮				
	F00FFFE0	w	0	WBMP[0x7FFF](0)	
	F0100000	w	0	WBMP[0](1)	Write 1 to L2C Cache Tag Back Map
	F0100020	w	0	WBMP[1](1)	
	⋮				
	F01FFFE0	w	0	WBMP[0x7FFF](1)	
	F0200000	w	0	WBMP[0](1)	
	⋮				
	FFEFFFE0	w	0	WBMP[0x7FFF](0xFE)	
	FFF00000	w	0	WBMP[0](0xFF)	Write 0xFF to L2C Cache Tag Back Map
	FFF00020	w	0	WBMP[1](0xFF)	
	⋮				
	FFFFFFE0	w	0	WBMP[0x7FFF](0xFF)	

以下，各資源について簡単に説明する。詳細については [4] 参照。

- EBAPRV
  任意の値を書くと Elastic Barrier の同期点到達を報告する。なお将来，その同期が成立したことを確認してはならない。

- EBPREQ
  任意の値を書くと Elastic Barrier の同期点到達を報告する。なお将来，その同期が成立したことを確認しなければない。

- EBRREQ
  Elastic Barrier の未確認同期点の中で，最も古いものが成立していれば -1, 成立していなければ 0 を返す。

- WBFLSH
  任意の値を書くと，書込データのマージのためのバッファに残っている全ての書込要求について，バスへの送出を開始する。書込要求のバスへの送出が完了したことは，STBAR で間接的に確認することができる。

- L2DMY
  任意の値を書いても，システムの状態を一切変更しないダミー・レジスタ。bq-write や prefetch/cflush に関する制限を確実に守るために使用することができる (7, 8 節参照)。

- EBCAP
  Elastic Barrier の未成立同期点の内，EBAPRV で報告したものの数を保持するカウンタ。

- EBCPR
  Elastic Barrier の未成立同期点の内，EBPREQ で報告したものの数を保持するカウンタ。

- EBCCP
  Elastic Barrier の成立した同期点の内，EBRREQ で確認していないもの数を保持するカウンタ。

- EBCLR
  Elastic Barrier に関するカウンタ EBCAP, EBCPR, EBCCP，並びに関連する上位階層が持つ全てのカウンタをクリアする。クリア動作の完了は，EBRREQ で確認しなければならない。

- INJLEV
  Injection を実行する条件を定める (4.2(3) 節参照)。

- PEID
  クラスタ内のどの PE であるかを，$0 \sim 3$ の値で返す。

- CSADR
  表 2 の cat が Ci であるような 2 次キャッシュ資源の，読出 / 書込に用いる。

- BMPRDR
  共有バスのスヌープ操作のための 2 次キャッシュ・タグのコピーである，バックマップの読出に使用する。

- L2CCTL
  2 次キャッシュの動作モードを定める (2 節参照)。

- L2CFRZ
  例外情報獲得などのために，2 次キャッシュをフリーズする (11 節参照)。

- L2CERR
  2 次キャッシュがメモリ・アクセス例外を検出したことを示す (11 節参照)。
- ERRST
  2 次キャッシュが検出したメモリ・アクセス例外の詳細情報を保持する (11 節参照)。
- FMPEAR
  パリティエラーが生じた FMP のアドレスを保持する。
- INTST, INTRST, INTSET
  共有バスからの割込に関する情報の参照と，そのリセット / セットを行なう (10 節参照)。
- SNDINV
  n をストアすると SNDDIR[n] を無効化する。危険な操作であるが，timeout-error 例外発生時に実施する必要がある。
- RCVINV
  n をストアすると RCVDIR[n] を無効化する。書込は危険であり，ハードウェアの診断などにしか用いられない。
- EMMTRP
  MBPL に対して緊急トラップを発行する。このトラップはメモリ機構をバイパスして伝えられる。
- L2CRDY
  1 を書込むと CPU によるメモリ・システムのアクセスが可能になる。リセット時には 0 であり，CPU のリセット割込ルーチンの実行を抑止している。2 次キャッシュを含むメモリ・システムの初期化後，MBPL などから L2CRDY を 1 にすることにより，リセット割込ルーチンの実行を開始することができる。0 の書込は極めて危険な操作で，実行される状況は想定されておらず，また実行の結果は全く保証されない。
- ECSAID
  ECSAID に下表に示す値をストアすると，その次に行なわれる Control Space へのストアが，他のプロセッサあるいは MBPL へのアクセスとして解釈される。

ECSAID	function
0x0	(reserved)
⋮	⋮
0x7	(reserved)
0x8	store PE0 resource
0x9	store PE1 resource
0xA	store PE2 resource
0xB	store PE3 resource
0xC	store MBPLresource
0xD	(reserved)
0xE	(reserved)
0xF	store PE0-3, MBPLresources

- DARRAY[n]
  2 次キャッシュのデータ・アレイ。書込は危険であり，ハードウェアの診断などにしか用いられない。
- INV[n]
  任意の値を書くと，2 次キャッシュのブロック n を無条件に無効化し，必要であればクラスタ・メモ

リへの書き戻しを行う。この操作の対象ブロックに対して他のプロセッサがアクセスを並行して行なうと，不正な結果が生じることがある。書込は危険であり，ハードウェアの初期化や診断などにしか用いられない。

- RCVDIR[n]
  共通バスから入力したパケットを処理するための情報を保持する。書込は危険であり，ハードウェアの診断などにしか用いられない。

- SNDDIRA[n], SNDDIRB[n]
  SNDBUF[n] のディレクトリ。timeout-error 例外発生時に原因を知るために読出す必要がある。書込は危険であり，ハードウェアの診断などにしか用いられない。。

- REQRCVQ, REPRCVQ, REQSNDQ, REPSNDQ, RETBUF, SDWORK
  2 次キャッシュ内部の各種キュー / バッファの制御情報。書込は危険であり，ハードウェアの診断などにしか用いられない。

- TOCTR
  メモリ・コマンドのタイムアウトを監視するタイマー。書込は危険であり，ハードウェアの診断などにしか用いられない。

- TOLMT
  メモリ・コマンドがタイムアウトする時間を定める。タイムアウトの検出は，TOLMT の値を L としたとき

  - CPU が発行してからの時間が L に満たないメモリ・コマンドは，タイムアウトとはならない。
  - CPU が発行してからの時間が $4 \times L$ を超えるメモリ・コマンドは，必ずタイムアウトとなる。

  という規則により行なわれる。書込は危険であり，ハードウェアの初期化や診断などにしか用いられない。

- WMCTR, WMLMT
  共有バスへの書込要求が，書込データのマージのためのバッファに滞在する時間を計測するタイマーと，滞在時間の限界値。書込は危険であり，ハードウェアの初期化や診断などにしか用いられない。

- PFCTR, PFLMT
  プリフェッチ要求に対する応答が，バッファに滞在する時間を計測するタイマーと，滞在時間の限界値。書込は危険であり，ハードウェアの初期化や診断などにしか用いられない。

- MESPATB, MESMSKB, MESCTRB
  共有バスに流れたパケットの種類が MESPATB に等しければ，MESCTRB がインクリメントされる。なおパケット種別の比較の際に，MESMSKB によるマスクが施される。

- MESPATP, MESMSKP, MESCTRP
  プロセッサから受けとった MCMD, その時点でのキャッシュ・ラインの状態などが MESPATP に等しければ，MESCTRP がインクリメントされる。なおパケット種別の比較の際に，MESMSKP によるマスクが施される。

- FMP[n]
  2 次キャッシュ・タグ。書込は危険であり，ハードウェアの初期化や診断などにしか用いられない。

- RBMP, WBMP[n](d)
  共有バスのスヌープ操作のための 2 次キャッシュ・タグのコピーである，バックマップの書込と読出。書込は危険であり，ハードウェアの初期化や診断などにしか用いられない。

10 共有バスからの割込

2 次キャッシュの内部レジスタ INTST (ASI=0x02, VA=0x100000F0) は，クラスタ内の他のプロセッサまたは MBPL が発行した割込要求を保持する。INTST は 8 ビット (byte) のレジスタであり， INTST[7] (MSB) がレベル 15 の割込 (マスク不能) に，INTST[6] がレベル 14 の割込 (マスク可能) に，以下順に INTST[i] ($i \geq 8$) がレベル i の割込 (マスク可能) に対応する。なお，INTST[15] は 2 次キャッシュが検出した非同期的なアクセス例外によっても 1 になる (11 節参照)。またレベル 7 以下の割込が報告されることはない。

INTST[7:0] のいずれかが 1 であると，1 であるものの中でもっともレベルが高い割込がプロセッサに通知される。また割込の通知は対応する INTST のビットを 0 にするまで，継続的に行なわれる。なお既に 1 であるビットに対して重複して割込要求がなされても，重複割込の通知などの特別な操作は一切行なわれない。

INTST の更新は，疑似レジスタ INTRST/INTSET へのストアにより行なう。INTRST へのストアは，通常の意味のストアではなく，ストア ・データの論理否定と INTST の値との論理積を INTST に書込むことを意味する。つまり INTST[i] を 0 にする (レベル $i+8$ の割込要因消去) は，2^iを INTRST にストアすることによって実現される。

同様に INTSET へのストアは，ストア ・データと INTST の値との論理和を INTST に書込むことを意味する。また同一クラスタに属する他のプロセッサ PEn への割込要求は，9 節で述べた ECSAID による他プロセッサの 2 次キャッシュ資源のアクセス機能を用いて，PEn の INTSET へのストアによって実現できる。

11 アクセス例外

11.1 例外報告

2 次キャッシュあるいは MBPL が検出したメモリ ・アクセスに関する例外は，以下の 4 種の例外 / 割込により報告される。

- data_store_error ..... CPU が (通常) 実行してしまった store 系の命令が，アクセス例外を引き起こした場合に生起する。
- instruction_access_exception ..... CPU が行なった命令フェッチが，アクセス例外を引き起こした場合に生起する。
- data_access_exception ..... CPU が行なった load 系の命令が，アクセス例外を引き起こした場合に生起する[4]。
- interrupt_level_15 ..... CPU が行なった命令フェッチやデータ ・アクセスが，その完了と同期しないようなアクセス例外を引き起こした場合に生起する。

なお 2 次キャッシュに関する限り instruction_access_exception と data_access_exception が同時に生起することはないが，data_store_error, instruction/data_access_exception, interrupt_level_15 は同時に生起する可能性がある。

また CPU 内部では，例外の原因を以下の資源に保持する。

(1) MFSR (ASI=0x04, VA=0x00000300)
interrupt-level-15 を除くメモリ ・アクセス例外の原因などが保持される。2 次キャッシュなどが検出する例外に関連するビットは以下の通り。

- SB (MFSR[15]) ..... data-store-error が生起したことを示す。

[4] アドレス変換に関する例外など，store 系の命令が CPU 内部で検出される同期的な例外を起こした場合にも，data-access-exeption となる。

- UD (MFSR[14]) ..... s-read, q-read, bq-read が empty のデータまたは空の queue をアクセスしたことを示す。この事象は例外として報告されるが，エラーではない。
- UC (MFSR[13]) ..... 命令フェッチまたは load 系の命令の実行時に，2 次キャッシュ・タグのパリティ・エラーあるいはクラスタ・メモリの 2 ビット・エラーが発生したことを示す[5]。
- TO (MFSR[12]) ..... 命令フェッチまたは load 系の命令を実行中に，L2C がタイムアウトを検出したことを示す。命令フェッチ / load 系命令自体がタイムアウトしたとは限らないことに注意されたい。
- BE (MFSR[11]) ..... 不正なメモリ・コマンドの実行などにより，bus-error を起こしたことを示す。
- FT (MFSR[4:2]) ..... 例外の原因を示し，6 であれば「確実に」2 次キャッシュで検出されたものである。但し，6 以外であっても 2 次キャッシュが例外を検出していることがある。

なお UD/UC/TO/BE は排他的であり，どれか一つだけが 1 である。2 次キャッシュが複数の例外を検出した場合には，後述する 2 次キャッシュ内部資源 ERRST に例外の種類が非排他的に保持される。

(2) MFAR (ASI=0x04, VA=0x00000400)
data_access_exception を起こした論理アドレスが保持される。なお instruction_access_exception の場合の論理アドレスは MFAR ではなく，トラップ発生時の PC/nPC として%l1, %l2 に保持される。

(3) SBCNTL (ASI=0x32)
CPU 内部の store buffer の状態を保持する。data_store_error が起こった時には，SBCNTL を参照して store buffer の先頭 (出力端) がどのエントリであるかを知り，store buffer tag (ASI=0x30) や store buffer data (ASI=0x31) を参照することにより，例外を起こしたアクセスの物理アドレスなどを知ることができる。

11.2 例外に関する 2 次キャッシュ内部資源

2 次キャッシュの内部資源の中で，以下のものがアクセス例外に関係する。

(1) L2CFRZ (ASI=0x02, VA=0x100000A8)
L2CFRZ のビット 0 が 1 である時，2 次キャッシュの動作が凍結される。これは共有バスに関する送受信を行なわないだけでなく，Control Space を除く全ての CPU からのアクセスを拒否することも意味している (アクセスがあると bus-error となる)。2 次キャッシュが保持するエラー情報を正しく参照するためには，L2CFRZ を 1 にする必要があるが，参照後 L2CFRZ を 0 にするまでは

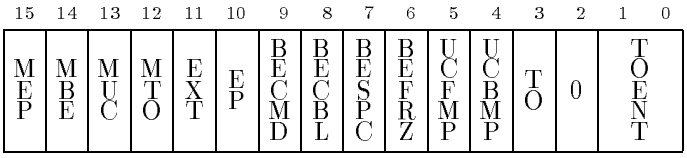
- 1 次キャッシュのロックされた領域の命令のみを実行する。
- CPU の内部資源および Control Space を除く全てのメモリ・アクセス命令を実行しない。

ことが必要である。

(2) L2CERR (ASI=0x02, VA=0x10000030)
2 次キャッシュ内部で例外を検出すると 1 になる。L2CERR が 1 である間は，完了していない書込操作が存在していても，全て完了したかのように見える。従って，例外検出から例外処理ルーチンへ移行するまでの間に STBAR 命令が実行されると，実際には完了していない書込を完了したと誤認することに注意されたい。なお L2CERR を 0 にすれば，例外検出以前に行なった書込を含めて，完了検出が正しく行なわれるようになる。

(3) ERRST (ASI=0x02, VA=0x10000038)

[5] クラスタ・メモリに関するエラーについては，MBPL の実装に依存する。



- MEP = ERRST[15] (Multiple Packet Error)
  ERRST が不正パケット受信が生起したことを示している間に (即ち EP=1)，再び不正パケットを受信したことを示す。
- MBE = ERRST[14] (Multiple Bus Error)
  ERRST がある原因による bus-error が生起したことを示している間に (即ち BExxx=1)，同じ原因による bus-error が生起したことを示す。このような状況は通常，2 次キャッシュが CPU に例外を通知して CPU が trap routine に移行するまでの間に行なわれる store buffer の掃き出しが，新たな例外を起こしたことを意味する。なお別の原因による bus-error が重複して生起した場合には，MBE は 1 にはならず，単に二つの BExxx が 1 になるだけである。
- MUC = ERRST[13] (Multiple Uncorrectable Error)
  ERRST が uncorrectable-error が生起したことを示している間に (即ち UCxxx=1)，同じ原因による uncorrectable-error が生起したことを示す。この状況では最後の uncorrectable-error に関する情報のみが保存されている。なお別の原因による Uncorrectable-error が重複して生起した場合には，MUC は 1 にはならず，単に二つの UCxxx が 1 になるだけである。
- MTO = ERRST[12] (Multiple Timeout Error)
  ERRST が time-out が生起したことを示している間に (即ち TO=1)，再び time-out が生起したことを示す。この状況では最後の time-out に関する情報のみが保存されている。
- EXT = ERRST[11] (External Asynchronous Error)
  INTST[7] を 1 にするような割込要求を，CPU, クラスタ内の他のプロセッサ，または MBPL が行なったことを示す。EXT は他の例外ビットと同時に 1 になることがある。
- EP = ERRST[10] (Error Packet)
  不正なパケットあるいは *error_reply* を受信したことを示す。*error_reply* に対応する CPU のアクセスが read または swap であれば，エラーの通知は uncorrectable-error によって同期的に行なわれ，それ以外の場合は NMI によって非同期的に行なわれる。

以下の 4 ビットは bus-error の原因を示す。なお，特に非同期的と記されたものは NMI により通知される。通常は排他的であるが，MBE の項で述べた原因によって二つのビットが 1 になることがある。また他の例外ビットと同時に 1 になることもある。

- BECMD = ERRST[9] (Bus Error on Illegal Command)
  定義されていないメモリ・コマンドを実行しようとしたことを示す。
- BECBL = ERRST[8] (Bus Error on Illegal Cachability)
  キャッシュ可否に関する制限に反して，メモリ・コマンドを実行しようとしたことを示す。
- BESPC = ERRST[7] (Bus Error on Command Specific Exception)
  以下に示すような，個々のメモリ・コマンドに特有の例外が発生したことを示す。
  - **prefetch/cflush** を doubleword 境界以外のアドレスに対して行なった。
  - **rg-read/rg-write** を doubleword 境界以外のアドレスに対して行なった。
  - **prefetch/cflush** に関する同一ブロックへの連続書込禁止規則が破られ，かつそれが致命的である。通知は非同期的である。
  - アドレス変換の過程で **s-read/f-read** を実行しようとした。通知は非同期的である。

- **q-read/p-read/q-check** の実行時に，キャッシュ・ブロックが他のプロセッサ / クラスタに共有されていることが判明した。通知は非同期的である。
- **bq-read/bq-check/bq-write** の実行時に，キャッシュ・ブロックが他のプロセッサ / クラスタに共有されていることが判明した。通知は非同期的である。
- **bq-wirte** に関する同一ブロックへの連続書込禁止規則が破られ，かつそれが致命的である。通知は非同期的である。
- **bq-read/bq-check** の実行過程で，不正な読出権移動が生じた。

- BEFRZ = ERRST[6] (Bus Error on Freezed L2C Access)
  L2CFRZ.BUSFRZ が 1 である間に，CPU が通常のメモリ空間をアクセスしようとしたことを示す。この例外はほとんどの場合致命的であり，CPU の watchdog reset を引き起こす可能性が高い。

以下の 2 ビットは，uncorrectable-error の原因を示す。

- UCFMP = ERRST[5] (Uncorrectable Error on Frontmap Bad Parity)
  フロントマップのパリティエラーが原因で，uncorrectable-error が生起したことを示す。なお，フロントマップ・パリティエラーは同期的（即ち命令フェッチや load 系命令の結果として）にも非同期的にも発生し，いずれの場合も UCFMP が 1 となる。CPU が受理したトラップが NMI であれば非同期的であり，そうでない場合は同期的である。またエラーを起こしたアドレスは FMPEAR に保持される。
- UCBMP = ERRST[4] (Uncorrectable Error on Backmap Bad Parity)
  バックマップのパリティエラーが原因で，uncorrectable-error が生起したことを示す。なお，バックマップ・パリティエラーは必ず非同期的に発生する。

以下のビットは，time-out の原因を示す。

- TO = ERRST[3] (Timeout Error)
  time-out が生起したことを示す。なお，time-out は同期的（即ち命令フェッチや load 系命令の結果として）にも非同期的にも発生し，いずれの場合も TO が 1 となる。CPU が受理したトラップが interrupt-level-15 であれば非同期的であるが，そうでない場合に同期的であるとは限らず，CPU 内部に保存された例外原因アドレス（論理アドレス）と，後述する SNDDIR に保存されたアドレス（物理アドレス）とを比較して判断する必要がある。
- TOENT = ERRST[1:0] (Timeout Entry of SNDDIR)
  time-out を起こした物理アドレスを含む情報は，8 エントリの SNDDIR のいずれかに保存されている。TOENT は情報が保存されている SNDDIR のエントリ番号を示す。

(4) SNDINV (ASI=0x02, VA=10000120)
timeout-error を起こした物理アドレスを含む情報は，SNDDIR の TOENT が示すエントリに保存されている。このエントリを保存状態にしたままにしておくと，2 次キャッシュが正常に動作できなくなるため，情報を獲得した後に速やかに解放しなければならない。この解放は，SNDINV にエントリ番号を書込むことにより行なうことができる。

(5) SNDDIRA (ASI=0x02, VA=80000080 ~ 800000B8)
timeout-error を起こした物理アドレスが，SNDDIRA[TOENT][35:0] に保存されている。SNDDIRA のアクセスは，L2CFRZ が 1 の状態で行なわなければならない。 SNDDIRA を読出して得られるその他の情報については，[4] を参照。

11.3 例外情報の獲得

アクセス例外に関する trap routine, 即ち data_store_error, instruction_access_exception, data_access_exception, interrupt_level_15 の trap routine は，'必ず」以下の方法によって 2 次キャッシュから例外情報を獲得しなければならない。なお， MFSR の FT が 6 (access bus error) でなくとも，2 次キャッシュが例外を検出していることがあるため，いずれにせよ必ず例外情報を獲得しなければならない。これを怠ると，2 次キャッシュの正常な動作が妨げられることがある。なお，2 次キャッシュが例外を検出すると書込完了検出が正しく行なわれないので，全ての trap routine では，WBFLSH を行なった後に STBAR を実行し，その後に INTST[15] が 1 であれば同様の例外情報獲得処理を行なうべきである。

1. L2CFRZ を 1 にする。これ以降 L2CFRZ を 0 にするまでは，1 次キャッシュのロックされた領域の命令のみを実行し，CPU の内部資源および Control Space を除く全てのメモリ・アクセス命令を実行しないようにしなければならない。
2. L2CERR を読出す。0 であれば 2 次キャッシュはエラーを検出していないので，L2CFRZ を 0 にして終了する。
3. ERRST を読出し，その後で ERRST に 0 を書込む。この読出とクリアがアトミックであることは，L2CFRZ が 1 であることにより保証される。
4. 読出した ERRST の TO が 1 であれば，SNDDIR[TOENT] を読出し，その後で SNDINV に TOENT を書込んで SNDBUF のエントリを解放する。
5. この時点で全てのエラー情報が獲得できたので，L2CFRZ を 0 にする。これ以後は命令領域やメモリ・アクセスに関する制限はない。
6. TO が 1 であれば，MBPL の書込完了確認機構が正しく動作しない可能性が高いので，MBPL に対して書込完了確認機構の正常化を指示し，その応答を待って L2CERR を 0 にする[6]。TO が 0 であれば無条件に L2CERR を 0 にする。
7. これ以後，2 次キャッシュは通常の動作ができる。

12 初期化

JUMP-1 システムの初期化手順が進行し，CPU の reset trap routine が起動される時点で，以下の初期化が完了している。

(1) 2 次キャッシュの内部資源の内，CPU 起動以前に初期化が必要なものの初期化。具体的な資源とその初期状態については [4] を参照。

(2) CPU の初期化プログラムのクラスタ・メモリへのロード。

(3) CPU の内部資源 MCNTL (ASI=0x04, VA=0x0) の初期化。主なビットの状態は以下の通り。

- MCNTL[15] : AC = 0
  1 次 / 2 次キャッシュを使用しない。
- MCNTL[13] : BT = 1
  命令フェッチのためのアドレス変換は，PA[35:28]=0xFF, PA[27:0]=VA[27:0] により行なう。
- MCNTL[10] : SB = 0
  store buffer を使用しない。

[6] 具体的な方法は，MBPL の実装に依存する。

- MCNTL[9] : IE = 0
  命令キャッシュ（1 次キャッシュ）を使用しない。
- MCNTL[8] : DE = 0
  データ・キャッシュ（1 次キャッシュ）を使用しない。
- MCNTL[7] : PSO = 0
  全ての store 系命令が書込完了を待つ。
- MCNTL[0] : EN = 0
  データ・アクセスのためのアドレス変換は，PA[35:32]=0, PA[31:0]=VA[31:0] により行なう。

従って，CPU の reset trap routine は，PA[35:0]=0xFF0000000 より実行を開始し (MCMD = 1111)，データ・アクセスの際の PA[35:32]（即ち MCMD）は 0 である。また命令 / データともにキャッシュ不可でアクセスされる。MCMD = 0000 および MCMD = 1111 をキャッシュ不可でアクセスした時の動作は，単純な読出 / 書込であるので，初期化プログラムは危険なメモリ・コマンド（例えば q-read による命令フェッチ）を発行することなく実行できる。

一方，命令フェッチの PA[31:28] は 0xF であり，LARGE, OWCH, DUMMY, HOME は全て 1 となっている。この内 DUMMY が 1 であることが若干問題であり，初期化プログラムがダミー・ページ中の命令を実行しようとしていると解釈される。この誤った解釈を避けるために，MBPL は PA[27:12]=0 の物理ページ（物理ページ番号 0）は DUMMY=1 であってもダミー・ページではないと解釈する。但しこの特別な解釈を行なうのは MBPL プログラムであるため，命令フェッチに長い時間を要する。

そこで，初期化プログラムはできるだけ速やかに，PA[31:28]=0xF の状態から脱出することが要求される。具体的には初期化プログラムの最初の命令を

```
sta    %g0, [%g0]0x04
```

として，MCNTL に 0 を書込むことが求められる。この操作により MCNTL.BT が 0 になり（他のビットは初期化時に全て 0），命令フェッチの PA[31:28] が VA[31:28]（即ち 0）となる。

なお 2 次キャッシュの資源の中には，CPU によって初期化されなければならないものがある。具体的な資源と，その求められる初期状態については [4] を参照。

参考文献

[1] 松本，CL-BUS と MBP の PE インターフェイス Version 0.1，JUMP-1 設計グループ内部資料

[2] 松本，JUMP-1 における Memory-Based Processor および Strategic Memory System Version 1.0，JUMP-1 設計グループ内部資料

[3] JUMP-1 設計グループ，JUMP-1MBPL 仕様書（未完）

[4] JUMP-1 設計グループ，JUMP-1 2 次キャッシュ内部資源仕様書 (ver 2.0)，JUMP-1 設計グループ内部資料