MMC の構成について

東京大学工学部 田中研究室 佐藤充

1 はじめに

MMC(Main Memory Controller) は, JUMP-.5 用に開発されたメモリコントローラである. MMC の概要, および外部仕様については 「JUMP-1 メモリコントローラ仕様説明書」を参考のこと.

本書では, MMC の内部構成, および内部レジスタについて説明する.

2 MMC の内部構成

MMC は以下の 7 つのユニット, および外部バッファによって構成される.

- SBUSCTL(Shared-BUS ConTroLler)

  クラスタ共有バスとのやりとりを行なうユニット. クラスタ共有バスからパケットを受け取り, 入力バッファに保存する. また, 出力バッファにたまったパケットをクラスタ共有バスへ出力する.

  SBUSCTL ではパケットを受け取ると, 簡単なデコードを行ない,

  - Request Packet か Reply Packet か
  - MMC が受け取るパケットかそうでないパケットか
  - 無条件に割り込みを発生するパケットかそうでないか
  - アドレスがロックアドレス (後述) と一致するか

  を判別する.

  パケットが MMC のハードウェア処理が必要なパケットであると判定されると, MMC はそのパケット情報を MTC に送る.

- MTC(Memory Timing Controller)

  MTC は, SBUSCTL から送られてきたハードウェア処理が必要なパケットの情報を受け取り, メモリアクセスを行なう.

  MTC は SBUSCTL からパケット情報を受け取ると, ADRCTL, SRAMCTL, DRAMCTL(必要があれば) を起動し, メモリへのアクセスを始める. 同時に, パケットをデコードし,

  - Read か Write か
  - SRAM-TAG, DRAM-TAG, ECC に対するアクセスか
  - 入力バッファ, 出力バッファに対するコントロールアクセスか
  - ブロック転送か

を判別する.

SRAM-TAG からタグ情報が到着すると, MTC はパケットの種類と TAG の内容に応じて DSP に対する割り込みを発生する.

- ADRCTL

  ADRCTL はアドレス線を制御する. MMC では物理的ピン数の制限から, SRAM と DRAM でアドレス線を共有している. そのため, メモリアクセスのタイミングに応じて, アドレス線の内容を変える必要がある.

  メモリアクセスが始まると, ADRCTL はパケットのアドレスにしたがって, SRAM のアドレスおよび DRAM の Row アドレスを出力する. しばらくすると, DRAMCTL から Col アドレス出力要求が来るので, ADRCTL は Col アドレスを出力する.

  また, リセット時には DRAM のモードレジスタ書き込み, リフレッシュ時には両バンク選択アドレスを出力する.

- SRAMCTL

  SRAMCTL は, SRAM アクセスのコントロールを行なう. 通常のメモリアクセスでは, SRAM は読みだし専用なので, SRAMCTL は OE の制御を行ない, SRAM からのデータを受け取る.

  SRAM に対する書き込み要求がくると, SRAMCTL は SRAM の OE, WE の制御を行ない, セットされたデータを出力する.

- DRAMCTL

  DRAMCTL は Jump-.5 のメインメモリである Synchronous DRAM を制御する. S-DRAM のシーケンスに応じて, ADRCTL にどのアドレス (Row または Col) を出力するかを指示し, データを受け取ったり送り出したりする. また読み出しの場合は, ECC のチェックを行ない, エラーを検出するとそれを通知する. 書き込みの場合は自動的に ECC を生成し, データとともに書き込む.

  また DRAM には DTAG が付随しており, この DTAG の読み書きも DRAMCTL が担当する.

- DSPIF

  DSPIF は DSP とのインタフェースである MCIF とのやりとりを行なう. DSP からの Memory Read/Memory Write を control_read/control_write のパケットの形にし, SBUSCTL へ送る.

  DSP は 32 ビット単位のアクセスを行なうので, write 時には read/modify/write を行ない, 32 ビット単位のアクセスを可能にしている.

  また, MMC には多くの内部資源が存在するが, それらに対するアクセスも (基本的に)DSPIF が行なう.

- REFRESH_COUNTER

  クロックをカウントし, 一定値になると DRAMCTL に対してリフレッシュのリクエストを行なう.

3 共用バッファへのアクセス

共用バッファは, 同一チップ内に収められる MMC, Bus-Chip, Cache Buffer が共用して利用する RAM である. 詳しい仕様については 「JUMP-1 共用バッファ仕様書」を参考のこと.

本節では, MMC がどのように共用バッファに対してアクセスするかを述べる.

3.1 入力バッファ(BUFX)

クラスタ共有バスから受け取ったパケットのデータ・フリットは, 入力バッファ(BUFX) に収められる. データは BUSA 上を流れるので, MMC は BUFX の Write Address および Write Enable を制御して, 適切なタイミングで BUFX が BUSA からデータを取り込めるようにする. この制御は SBUSCTL が行なう.

入力バッファに蓄えられたデータは, 基本的には DRAM に出力される. DRAM は BUSB 上に接続しているので, MMC はメモリに対するアクセスタイミングに合わせて BUFX の Read Address をコントロールする. この制御は MTC が行なう.

3.2 出力バッファ(BUFY)

出力バッファには, クラスタ共有バスへ出力すべきパケットのデータフリットが収められる. 基本的に, MMC がクラスタ共有バスに出力するパケットのデータには, メモリから読み出されたデータを載せる. したがって, MMC はメモリの読みだしタイミングに合わせて BUFY の Write Address および Write Enable をコントロールし, データを BUFY に収める.

出力バッファに蓄えられたデータは, SBUSCTL によって, クラスタ共有バスにパケットを出力する時に読み出される.

3.3 一時バッファ(BUFZ)

一時バッファは, DSP からのアクセス時に使用される. 一時バッファには 2 通りのデータが収められる.

1. DSP が読みだしたメモリ・データ

   DSP がメモリアクセスを発生すると, MMC はメモリから読み出されたデータを, 出力バッファ(BUFY) の代わりに一時バッファ(BUFZ) に書き込む. この選択は MTC が行なう. MTC が書き込んだメモリ・データは, DSPIF によって読み出され, DSP への Read Reply として出力される.

2. DSP が書き込むデータ

   DSP がメモリに対する書き込みを行なおうとする時は, まず当該メモリを読みだし, そのデータを一時バッファに蓄える. その後, DSPIF は DSP が書き込みを行なうデータ (32 ビット単位) のみを上書きし, MTC に対して書き込み要求を出す. MTC は上書きされたデータをメモリに書き込む.

一時バッファは, MTC, DSPIF 両者によって読み書きされる. しかし, 一時バッファは DSP が行なう処理のためのバッファであるから, 同時にアクセスが発生することはあり得ない.

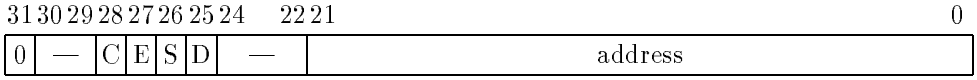
4 MMC の内部レジスタ

MMC には内部資源として, 数多くのレジスタを持つ. レジスタへのアクセスは, DSP からのコントロールアクセスという形でアクセスできる. コントロールアクセスかどうかは, メモリアクセスのアドレスによって決定される (「JUMP-1 メモリコントローラ仕様説明書」を参照)(図 1).

4.1 コントロールアドレス

内部レジスタは, コントロールアドレス内にマップされている. 有効なアドレスは 22 ビットあり, うち上位 3 ビット (b21 – b19) がブロックを表している (表 1).

内部レジスタの種類とアドレスを, 表 2に示す.



C: CNTL, E: ECC, S: S/D, D: D/T

図 1: DSP address map

表 1: Blocks in MMC

b21:b20:b19			block name
0	0	0	Shared Buffer(SB)
0	0	1	Cluster Bus Controller(CBC)
0	1	0	Memory Timing Controller(MTC)
0	1	1	Address Controller(ACTL)
1	0	0	SRAM Controller(SCTL)
1	0	1	DRAM Controller(DCTL)
1	1	0	DSP Interface(DSPIF)
1	1	1	others

4.2 共用バッファ(SB)

4.2.1 BUFX

Address = 0x000000 – 0x00003F, Read Only

offset	31 ... 0
00	BUFX(00)[63:32]
01	BUFX(00)[31: 0]
02	BUFX(01)[63:32]
03	BUFX(01)[31: 0]
...	
3F	BUFX(1F)[31: 0]

共用バッファの BUFX. MMC ではクラスタ共有バスからの入力パケットのデータを保持するのに使われる. BUFX は 64 ビット幅なので, DSP からは上位 32 ビットと下位 32 ビットの 2 word のように見える.

MMC では, BUFX は 4 word ごとに使用される. BUFX(00)–BUFX(03) は入力バッファ0, BUFX(04)–BUFX(07) は入力バッファ1, ..., 以下, 入力バッファ7 まで定義されている. 入力バッファ0–3(BUFX(00)–BUFX(0F)) は, Reply Packet 用, 入力バッファ4–7(BUFX(10)–BUFX(1F)) は, Request Packet 用である.

入力バッファに有効なデータが入っているかどうかは, recv_buf_state(Address = 0x080200) によって示される. ただし, バッファを埋めているパケットがデータなしのパケット (packet length = 1) である場合には, 入力バッファの内容は意味がない.

表 2: Internal Registers

Block	Address(Hex)	Name	Description	RW
SB	000000-00003F	BUFX	BUFX	R
	000040-00007F	BUFY	BUFY	W
	000080-000087	BUFZ	BUFZ	RW
CBC	080000-080047	recv_adr_reg	入力パケットのアドレス部	R
	080080-0800C7	recv_wwe_reg	入力パケットの WWE 部	R
	080100-08013F	send_adr_reg	出力パケットのアドレス部	R
	080180-0801BF	send_wwe_reg	出力パケットの WWE 部	R
	080200	recv_buf_state	入力バッファの状態	RW
	080208	send_buf_state	出力バッファの状態	RW
	080210	MTC req. state	MTC へのリクエスト状態	RW
	080218	Bus signals	クラスタ共有バスの信号線	R
	080300	lock address	ロックアドレス	RW
	080308	lock adr. clr.	ロックアドレスのクリア	W
	080310	int. cause reg.	割り込み原因となったバッファの番号	R
	080400-08041F	PEND counters	現在処理中のパケット数	RW
	080420-08043F	dec. PEND	PEND カウンタを-1 する	W
	080500	IRR reg.	バス入力拒否レジスタ	RW
	080508	OSR reg.	バス出力停止レジスタ	RW
MTC	100000-100002	proc_adr_flit	処理中のアドレスフリット	R
	100008	proc_buf_num	処理中のバッファ番号	R
	100010-100013	reply_adr_flit	リプライ用アドレスフリット	R
	100018	wbuf_num	リプライを書き込むバッファ番号	R
	100020	decoded result	デコード結果	R
	100028	interrupt	割り込み状態	R
	100030	dece_reg	DRAM のエラーチェック ・イネーブル	RW
ACTL	180000	adr_latch	アドレス	R
	180008	mode reg.	DRAM 用のモードレジスタ	RW
SCTL	200000	r_data	SRAM データ	R
	200008	w_data	書き込み用データ	R
DCTL	280000-28003	r_data	DRAM データ	R
	280008	r_ecc	ECC データ	R
	280010	r_tag	タグデータ	R
	280018	r_syn	シンドローム ・データ	R
	280020	r_parity	パリティ ・データ	R
	280028	error	エラー	R
	280030	error num.	エラー発生ワード番号	R
	280038	refresh	リフレッシュ	R

表 3: Internal Registers (続き)

Block	Address(Hex)	Name	Description	RW
DSPIF	300000-300001	dsp_adr	アドレス	R
	300020-300027	dbuf	書き込み用データバッファ	R
	300028	dbuf_fe	データバッファfull/empty ビット	R
その他	380000	int. src.	割り込み原因レジスタ	R
	380008	int. reset	割り込みリセット	W
	380010	to_refresh	リフレシュ参照カウンタ	RW
	380018	refr. count	リフレッシュ ・カウンタ	R

4.2.2 BUFY

Address = 0x000040 – 0x00007F, Write Only

offset	31	0
00	BUFY(00)[63:32]	
01	BUFY(00)[31: 0]	
02	BUFY(01)[63:32]	
03	BUFY(01)[31: 0]	
...		
3F	BUFY(1F)[31: 0]	

共用バッファの BUFY. MMC ではクラスタ共有バスへ出力するパケットのデータを保持するのに使われる. BUFY は 64 ビット幅なので, DSP からは上位 32 ビットと下位 32 ビットの 2 word のように見える.

MMC では, BUFY は 4 word ごとに使用される. BUFY(00)–BUFY(03) は出力バッファ0, BUFY(04)–BUFY(07) は出力バッファ1, ..., 以下, 出力バッファ5 まで定義されている (BUFY(18)–BUFY(1F) は未使用). 出力バッファ0–3(BUFY(00)–BUFY(0F)) は, MMC がハードウェア的に生成する Reply Packet 用, 出力バッファ4(BUFY(10)–BUFY(13)) は, DSP が生成する Reply Packet 用, 出力バッファ5(BUFY(14)–BUFY(17)) は, DSP が生成する Request Packet 用である[1].

出力バッファに有効なデータが入っているかどうかは, send_buf_state(Address = 0x080208) によって示される. ただし, バッファを埋めているパケットがデータなしのパケット (packet length = 1) である場合には, 出力バッファの内容は意味がない.

DSP がデータつきパケットを送出する時には, まず, このアドレスを用いて BUFY に有効なデータを書き込む必要がある. DSP から出力バッファ4 または出力バッファ5 にデータをセットする際には, send_buf_state(Address = 0x080208) を読み出して, 出力バッファ4 番または 5 番が使用中でないことを確認すること. また, 出力アドレスレジスタ (Address = 0x080180) にデータを書き込むと, 出力バッファステート (Address = 0x080208) の内容がセットされるので, DSP からデータつきパケットを書き込む場合には, 出力アドレスレジスタより先に, 出力バッファにデータを書き込むこと.

[1] MMC はハードウェア的に Request Packet を生成することはない

4.2.3 BUFZ

Address = 0x000080 – 0x000087, Read Write

offset 31	0
00	BUFZ(00)[63:32]
01	BUFZ(00)[31: 0]
02	BUFZ(01)[63:32]
03	BUFZ(01)[31: 0]
04	BUFZ(02)[63:32]
05	BUFZ(02)[31: 0]
06	BUFZ(03)[63:32]
07	BUFZ(03)[31: 0]

共用バッファの BUFZ. MMC では DSP がメモリ処理する場合の一時バッファとして使われる. BUFY は 64 ビット幅なので, DSP にからは上位 32 ビットと下位 32 ビットの 2 word のように見える.

DSP がメモリから read を行なうと, MMC はメモリから読み込んだデータを一旦 BUFZ にため, BUFZ から DSP へデータの転送を行なう. DSP がメモリへ write を行なう場合は, まずメモリからデータを BUFZ に書き込み, 必要な部分を上書きした後, BUFZ の内容をメモリに書き込む.

一般には, DSP は BUFZ の内容を読み書きする必要はない.

4.3 クラスタバスコントローラ (CBC)

4.3.1 recv_adr_reg: 入力アドレスレジスタ

Address = 0x080000 – 0x080047, Read Only

offset 31	2 1 0
00	address(0)[63:32]
01	address(0)[31: 0]
02	— | WWE
03	—
04	—
...	
07	—
08	address(1)[63:32]
09	address(1)[31: 0]
0A	— | WWE
0B	—
...	
40	address(8)[63:32]
41	address(8)[31: 0]
42	— | WWE
43	—
...	
47	—

クラスタ共有バスから受け取ったパケットのアドレスフリットを保持しているレジスタ. アドレスフリットは WWE フィールドを含めて 66 ビットなので, DSP からはアドレス部上位 32 ビット, アドレス部下位

32 ビット, WWE フィールドの 3 word のように見える.

入力アドレスレジスタは, 9 つのアドレスフリットを保持することができる. address(0), address(1), ..., address(7) はそれぞれ, 入力バッファ(BUFX) の 0~7 に対応している. address(8) は, DSP からメモリをアクセスする時に用いる特殊レジスタであり, データは一時バッファ(BUFY) に蓄えられる.

入力アドレスレジスタの内容を見ることで, DSP は MMC がクラスタ共有バスから受け取ったパケットのアドレスフリットを検査することができる.

なお, MMC 内部では address(0)–address(3) は Reply Packet 用, address(4)–address(7) は Request Packet 用, address(8) は DSP 用である.

4.3.2 recv_wwe_reg: 入力 WWE レジスタ

Address = 0x080080 – 0x0800C7, Read Only

offset	31 ... 8	7 ... 0
00	—	WWE(0)(0:3)
01	—	
02	—	
03	—	
04	—	
...		
07	—	
08	—	WWE(1)(0:3)
09	—	
0A	—	
0B	—	
...		
3F	—	

クラスタ共有バスから受け取ったパケットのデータ部 WWE フィールドを保持しているレジスタ. 1 word 内に 4 データフリットの WWE フィールドが保持されている. そのビット配置は, 以下のようになっている.

7 6	5 4	3 2	1 0
WWE 3	WWE 2	WWE 1	WWE 0

入力アドレスレジスタは, 8×4 つの WWE フィールドを保持することができる. WWE(0), WWE(1), ..., WWE(7) はそれぞれ, 入力バッファ(BUFX) の 0~7 に対応している.

入力 WWE レジスタの内容を見ることで, DSP は MMC がクラスタ共有バスから受け取ったパケットの WWE フィールドを検査することができる.

なお, MMC 内部では WWE(0)–WWE(3) は Reply Packet 用, WWE(4)–WWE(7) は Request Packet 用である.

4.3.3 send_adr_reg: 出力アドレスレジスタ

Address = 0x080100 – 0x08013F, Read Only*

offset	31 … 2	1	0
00	address(0)[63:32]		
01	address(0)[31: 0]		
02	—		WWE
03	—		
04	—		
…			
07	—		
08	address(1)[63:32]		
09	address(1)[31: 0]		
0A	—		WWE
0B	—		
…			
38	address(7)[63:32]		
39	address(7)[31: 0]		
3A	—		WWE
3B	—		
…			
3F	—		

*: address(4), address(5) のみ Read Write

クラスタ共有バスへ送り出すパケットのアドレスフリットを保持しているレジスタ. アドレスフリットは WWE フィールドを含めて 66 ビットなので, DSP からはアドレス部上位 32 ビット, アドレス部下位 32 ビット, WWE フィールドの 3 word のように見える.

出力アドレスレジスタは, 8 つのアドレスフリットを保持することができる. address(0), address(1), …, address(7) はそれぞれ, 出力バッファ(BUFY) の 0~7 に対応している.

出力アドレスレジスタの内容を見ることで, DSP は MMC がクラスタ共有バスへ送り出すパケットのアドレスフリットを検査することができる. また, DSP から共有バスへパケットを送り出す時は, このレジスタの address(4)(Reply Packet の場合), address(5)(Request Packet の場合) に, 送り出そうとするパケットのアドレスフリットをセットしなくてはならない. DSP から address(4) または address(5) にアドレスをセットする際には, send_buf_state(Address = 0x080208) を読み出して, 出力バッファ4 番または 5 番が使用中でないことを確認すること. また, 出力アドレスレジスタにデータを書き込むと, 出力バッファステート (Address = 0x080208) の内容がセットされるので, DSP からパケットを書き込む場合には, 出力アドレスレジスタを最後に書き込むこと.

なお, MMC 内部では address(0)–address(3) は MMC がハードウェア的に送り出す Reply Packet 用, address(4) は DSP が送り出す Reply Packet 用, address(5) は DSP が送り出す Request Packet 用, address(6), address(7) は未使用である.

4.3.4 send_wwe_reg: 出力 WWE レジスタ

Address = 0x080180 – 0x0801BF, Read Only*

offset	31 — 8	7 — 0
00	—	WWE(0)(0:3)
01	—	
02	—	
03	—	
04	—	
...		
07	—	
08	—	WWE(1)(0:3)
09	—	
0A	—	
0B	—	
...		
3F	—	

*: WWE(4), WWE(5) のみ Read Write

クラスタ共有バスへ送り出すパケットのデータ部 WWE フィールドを保持しているレジスタ. 1 word 内に 4 データフリットの WWE フィールドが保持されている. そのビット配置は, 以下のようになっている.

7 — 6	5 — 4	3 — 2	1 — 0
WWE 3	WWE 2	WWE 1	WWE 0

出力アドレスレジスタは, 8×4 つの WWE フィールドを保持することができる. WWE(0), WWE(1), ..., WWE(7) はそれぞれ, 出力バッファ(BUFY) の 0~7 に対応している.

出力 WWE レジスタの内容を見ることで, DSP は MMC がクラスタ共有バスへ送り出すパケットの WWE フィールドを検査することができる. また, DSP から共有バスへパケットを送り出す時は, このレジスタの WWE(4)(Reply Packet の場合), WWE(5)(Request Packet の場合) に, 送り出そうとするパケットの WWE フィールドをセットしなくてはならない. DSP から WWE(4) または WWE(5) にアドレスをセットする際には, send_buf_state(Address = 0x080208) を読み出して, 出力バッファ4 番または 5 番が使用中でないことを確認すること. また, 出力アドレスレジスタ (Address = 0x080180) にデータを書き込むと, 出力バッファステート (Address = 0x080208) の内容がセットされるので, DSP からデータつきパケットを書き込む場合には, 出力アドレスレジスタより先に, 出力 WWE レジスタにデータを書き込むこと.

なお, MMC 内部では WWE(0)–WWE(3) は MMC がハードウェア的に送り出す Reply Packet 用, WWE(4) は DSP が送り出す Reply Packet 用, WWE(5) は DSP が送り出す Request Packet 用, WWE(6), WWE(7) は未使用である.

4.3.5 recv_buf_state: 入力バッファステート

Address = 0x080200, Read Write

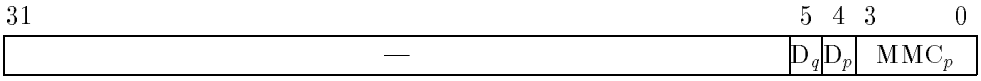
31 — 9	8	7 — 4	3 — 0
—	D	Req.	Rep.

入力バッファ(BUFX) の状態を表しているレジスタ. 1 の時には当該バッファが使用中 (パケットが入っている状態) であることを示し, 0 の時には未使用 (パケットが入っていない状態) であることを示す.

下位ビットから順に, 入力バッファの 0 番, 1 番, ..., 8 番に対応している. MMC では, 入力バッファ0~3番がクラスタ共有バスからの Reply Packet 用, 4~7 番がクラスタ共有バスからの Request Packet 用, 8 番が DSP 用となっている.

4.3.6 send_buf_state: 出力バッファステート

Address = 0x080208, Read Write



出力バッファ(BUFY) の状態を表しているレジスタ. 1 の時には当該バッファが使用中 (パケットが入っている状態) であることを示し, 0 の時には未使用 (パケットが入っていない状態) であることを示す.

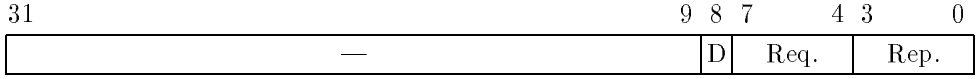
下位ビットから順に, 出力バッファの 0 番, 1 番, ..., 5 番に対応している. MMC では, 出力バッファ0~3番が, MMC がハードウェア的に生成するクラスタ共有バスへ出力する Reply Packet 用[2], 4 番が DSP がクラスタ共有バスへ出力する Reply Packet 用, 5 番が DSP がクラスタ共有バスに出力する Request Packet 用となっている.

DSP からクラスタ共有バスへパケットを出力する時には, 出力アドレスレジスタにデータを書き込むことによって, 自動的に出力バッファステートがセットされる. したがって, 一般には DSP は出力バッファステートにデータを書き込む必要はない.

DSP からクラスタ共有バスへ出すパケットが実際に出力されたかどうかは, このレジスタを検査することによって知ることができる. また, DSP から出力バッファ(BUFY) にデータを書き込む前には, このレジスタを検査して, 以前書いたデータが実際にクラスタ共有バスへ出ていったかどうかを確認すること.

4.3.7 mtc_req_state: MTC 要求ステート

Address = 0x080210, Read Write



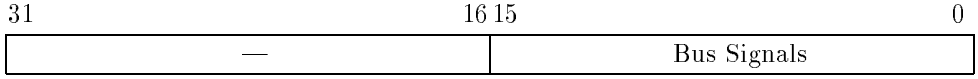
入力バッファ(BUFX) からのメモリアクセス要求の状態を表しているレジスタ. 1 の時には当該バッファからメモリアクセス要求が出ていることを示し, 0 の時には当該バッファにはパケットが入っていないか, メモリアクセス要求が出ないパケットが入っていることを示す.

MTC は, このレジスタから出るメモリアクセス要求に応じて, メモリアクセスを処理する.

下位ビットから順に, 入力バッファの 0 番, 1 番, ..., 8 番に対応している. MMC では, 入力バッファ0~3番がクラスタ共有バスからの Reply Packet 用, 4~7 番がクラスタ共有バスからの Request Packet 用, 8 番が DSP 用となっている.

4.3.8 bus_signals: クラスタ共有バス信号

Address = 0x080218, Read Only



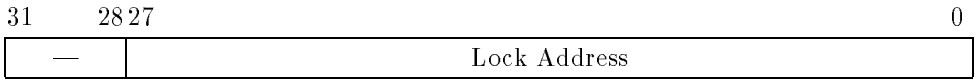
クラスタ共有バスの信号線をモニタするレジスタ. 以下のようなバス信号をモニタすることができる. 信号線の意味については, 「JUMP-1 共通バス仕様書」を参照のこと.

[2] MMC がハードウェア的に生成するパケットは Reply Packet のみである

bit	signal name
0	X_BREQ0
1	X_BREQ1
2	X_BBUSY0
3	X_BBUSY1
4	X_ASTB_I
5	X_ASTB_O
6	X_BGRT
7	X_BRETRY_O
8	X_BRETRY_I
9	X_CRPLY_I
10	WWE0
11	WWE1
13	X_PEND0
14	X_PEND1
14	X_PEND2
14	X_PEND3

4.3.9 lock_adr_reg: ロックアドレスレジスタ

Address = 0x080300, Read Write



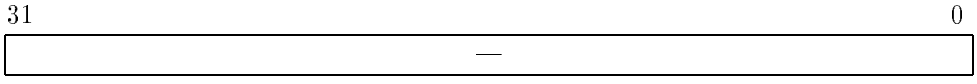
クラスタ共有バスからのパケットを reject するためのロックアドレスを保持するレジスタ. このレジスタに設定したアドレスと同じアドレスを持つパケットは, MMC によって retry される. したがって, ロックアドレスレジスタに設定したアドレスのメモリに対して DSP が単一操作を行なうことができる.

実際には, ロックをかけるまでに受け取ったパケットの中にロックアドレスレジスタと同じアドレスを持つパケットが入っている可能性があるので, DSP で単一操作を行なう場合には, ロックアドレスレジスタにアドレスを設定した後, MTC 要求ステート (Address = 0x080210) を検査し入力バッファの中にメモリアクセス要求を出しているパケットがないことを確認してから, メモリ操作を行なうこと.

なお, MMC ではロックアドレスはひとつしか設定できない.

4.3.10 lock_adr_clr: ロックアドレスクリアレジスタ

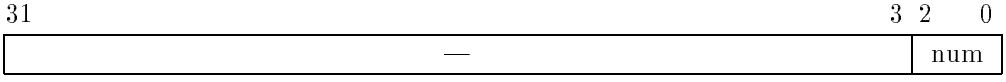
Address = 0x080308, Write Only



ロックアドレスレジスタ (Address = 0x080220) で設定したアドレスをクリアする. このレジスタに任意の値を書き込むと (書き込み動作を行なうと), ロックアドレスをクリアする.

4.3.11 int_cause_reg: 割り込み原因バッファ番号レジスタ

Address = 0x080310, Read Only



クラスタ共有バスから受け取ったパケットが原因で割り込みが起こった場合に, どの入力バッファに取り込まれたパケットによって割り込みが引き起こされたかを表すレジスタ. ビット 2∼ビット 0 は, 入力バッファの番号を示している.

クラスタ共有バスから受け取ったパケットによって割り込みが生じた場合, DSP はまず割り込み原因レジスタ (Address = 0x380000) によって割り込みが CBC で生じたものであることを確認する. その後に割り込み原因バッファ番号レジスタによって, 割り込みの元となったパケットがバッファのどこに格納されているかを調べ, 入力アドレスレジスタ (Address = 0x080000), 入力 WWE レジスタ (Address = 0x080080), 入力バッファ(BUFX: Address = 0x000000) によってパケットデータを参照する.

4.3.12 PEND_counter: PEND カウンタ

Address = 0x080400, Read Write

offset	31 ... 4	3 ... 0
00	—	PEND0
01	—	
02	—	
...		
07	—	
08	—	PEND1
09	—	
0A	—	
...		
0F	—	
10	—	PEND2
11	—	
...		
17	—	
18	—	PEND3
19	—	
...		
1F	—	

SPARC に送る PEND 信号を生成するためのカウンタ. クラスタ共有バスからメモリアクセス要求を発生するパケットが到着すると, このカウンタが+1 され, メモリアクセスが終了するとこのカウンタが自動的に−1 される. PEND カウンタのカウントアップ, ダウンの詳しい条件に関しては, 「UMP-1 メモリコントローラ仕様説明書」を参照のこと.

PEND0 が CC0 からのパケットに対するカウンタ, PEND1 は CC1, PEND2 は CC2, PEND3 は CC3 に対応する.

DSP が処理しなくてはならないメモリアクセスに対しては, MMC は PEND カウンタをカウントアップするが, カウントダウンはしない. したがって, メモリアクセス終了時に, DSP は明示的に PEND カウンタをカウントダウンする必要がある. PEND カウンタをカウントダウンする方法は, PEND カウンタに直接値を書き込む方法と, PEND ダウンレジスタ (Address = 0x080420) を用いて PEND カウンタを−1 する方法のふたつの方法がある. しかし, PEND カウンタに値を直接書き込むと, MMC による自動カウントダウンと重なった場合に, カウンタの値が実際のメモリアクセスの数とずれてしまう可能性がある. したがって, DSP から PEND カウンタをカウントダウンする場合には, MTC 要求ステートでメモリアクセス要求

がないことを確認した上でメモリアクセスを処理するか, PEND ダウンレジスタ (Address = 0x080420) を用いて, ひとつずつカウントダウンするか, どちらかの方法をとらなければならない.

4.3.13 PEND_decrement: PEND ダウンレジスタ

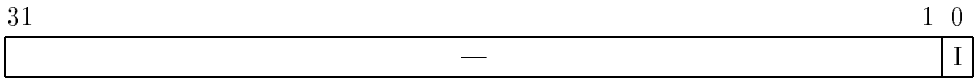
Address = 0x080420, Write Only

PEND カウンタをダウンカウントするためのレジスタ. このレジスタに任意の値を書き込むと (書き込み動作を行なうと), 対応する PEND カウンタを−1 する. PEND カウンタに値を直接書き込む方法と違って, この PEND ダウンレジスタを用いると, MMC による自動カウントダウンと重なっても正しく動作する (同一クロックに MMC による自動カウントダウンと PEND ダウンレジスタに対するアクセスが重なると, PEND カウンタは−2 される).

offset 0 が PEND0(CC0 用の PEND カウンタ), offset 8 が PEND1(CC1 用), offset 10 が PEND2(CC2 用), offset 18 が PEND3(CC3 用) に対応する.

4.3.14 irr_reg: バス入力拒否レジスタ

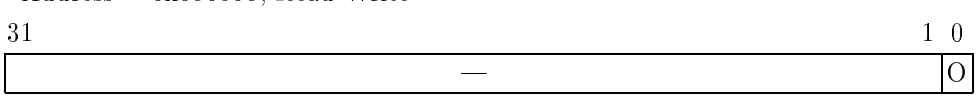
Address = 0x080500, Read Write



バス入力拒否レジスタ (Input Reject Register) は, クラスタ共有バスからのパケットの受け取りをコントロールするレジスタである. I=0 の時, MMC はクラスタ共有バスからのパケットに対して通常の受け取り動作を行なう. I=1 の時, MMC は入力バッファ(BUFX) に空きがあってもクラスタ共有バスに対して Busy を出し, それ以上パケットを受け取らないようにする.

4.3.15 osr_reg: バス出力停止レジスタ

Address = 0x080508, Read Write



バス出力停止レジスタ (Output Stop Register) は, クラスタ共有バスへのパケットの送出をコントロールするレジスタである. O=0 の時, MMC はクラスタ共有バスに対して通常の送出動作を行なう. O=1 の時, MMC は出力バッファ(BUFY) にパケットが残っていてもクラスタ共有バスに対して Request を出さず, それ以上パケットを送出しないようにする.

4.4 メモリタイミングコントローラ (MTC)

4.4.1 proc_adr_flit: 処理中アドレスフリット

Address = 0x100000 – 0x10002, Read Only

offset	31 … 2	1	0
00	address[63:32]		
01	address[31: 0]		
02	—	W	WE

現在 MTC で処理中のパケットのアドレスフリットを保持しているレジスタ. MTC はこのアドレスフリットを元に, さまざまなメモリアクセスのタイミングを生成する.

MTC で割り込みが発生した場合は, 割り込みの状態が保存される. したがって, MTC による割り込み発生時には, 処理中アドレスフリットには, 割り込みの元となるパケットのアドレスフリットが保存されていることになる.

4.4.2 proc_buf_num: 処理中バッファ番号

Address = 0x100008, Read Only

31 … 3	2 … 0
—	num

現在 MTC で処理中のパケットの, 入力バッファ(BUFX) の番号を保持しているレジスタ.

MTC で割り込みが発生した場合は, 割り込みの状態が保存される. したがって, MTC による割り込み発生時には, 処理中バッファ番号には, 割り込みの元となるパケットの入力バッファ(BUFX) の番号が保存されていることになる.

4.4.3 reply_adr_flit: リプライアドレスフリット

Address = 0x100010 – 0x10012, Read Only

offset	31 … 2	1	0
00	address[63:32]		
01	address[31: 0]		
02	—	W	WE

現在 MTC で処理中のパケットに対応するリプライパケットのアドレスフリットを保持しているレジスタ. 処理中のパケットのアドレスフリット (proc_adr_flit: Address = 0x100000) と SRAM Tag のデータから, MTC によって生成される. メモリアクセスが終了すると, MMC はこのレジスタに保存されているアドレスフリットを持つリプライを, クラスタ共有バスに送出する.

処理中のパケットが, リプライを必要としないパケットの場合, このレジスタのデータは意味のないデータである.

4.4.4 wbuf_num: リプライ書き込みバッファ番号

Address = 0x100018, Read Only

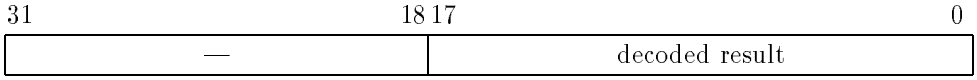
31 … 3	2 … 0
—	num

現在 MTC で処理中のパケットに対応するリプライパケットの, 出力バッファ(BUFY) の番号を保持しているレジスタ. メモリアクセスが終了すると, MTC はこのレジスタに保持されている出力バッファ(BUFX) にリプライパケットのアドレス, WWE およびデータを書き込む.

処理中のパケットが, リプライを必要としないパケットの場合, このレジスタのデータは意味のないデータである.

4.4.5 decoded_result: MTC デコード結果レジスタ

Address = 0x100020, Read Only

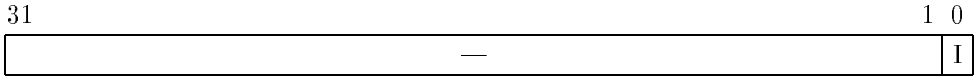


proc_adr_flit(Address = 0x100000) をデコードした結果を保持しているレジスタ. 以下のような情報が含まれている.

bit	signal name
0	write_or_read (1: write, 0: read)
1	sdram_access (1: access to DRAM, 0: not access to DRAM)
2	from_dram (1: Data read from DRAM Data)
3	from_sram (1: Data read from SRAM Tag)
4	from_dtag (1: Data read from DRAM Tag)
5	from_ecc (1: Data read from DRAM ECC)
6	from_bufz (1: DSP memory Access)
7	to_dram (1: Data write to DRAM DATA)
8	to_sram (1: Data write to SRAM Tag)
9	to_dtag (1: Data write to DRAM Tag)
10	ctrl_bufx (1: control read from BUFX)
11	ctrl_bufy (1: control write to BUFY)
12 – 14	ctrl_buf_adr (control read/write buffer number)
15	block_or_not (1: block access, 0: double word access)
16 – 17	start_low_bits (block offset)

4.4.6 interrupt: MTC 割り込みレジスタ

Address = 0x100028, Read Only



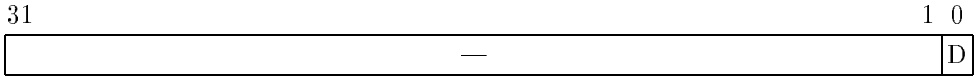
MTC による割り込みを表すレジスタ. このレジスタの I の値が 1 ならば, MTC 内部が割り込み状態であることを示す. MTC が割り込み状態になる原因には, 以下のようなものがある.

- SRAM Tag によるトラップ
- block, double word 以外のサイズの書き込みアクセス
- DCTL でのエラー発生

MTC の割り込み状態は, 割り込み解除レジスタ (Address = 0x380008) により解除することができる.

4.4.7 dece_reg: DRAM エラー検査レジスタ

Address = 0x100030, Read Write

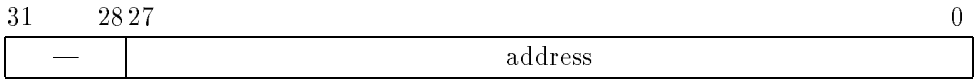


DRAM DATA に付随している ECC によるエラーチェックを有効 (D = 1)/無効 (D = 0) にするためのレジスタ. リセット時には無効 (D = 0) になっている.

4.5 アドレスコントローラ (ACTL)

4.5.1 adr_latch: アドレスラッチ

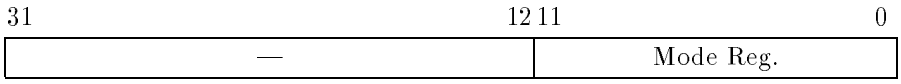
Address = 0x180000, Read Only



現在処理中のメモリアドレスを保持しているレジスタ. 実際には, SRAM, DRAM のアドレス線が重複しているので, この値がそのまま外部に出力されるわけではないことに注意.

4.5.2 mode_reg: MODE レジスタ

Address = 0x180008, Read Write



Synchronous DRAM にセットするモードレジスタを保持しているレジスタ. モードレジスタの各ビットには, 次のような意味がある.

bit	description
2 – 0	Burst Length (通常は “010”(burst length = 4) に設定)
3	Wrap Type (通常は ‘0’(sequential) に設定)
6 – 4	Latency MODE (通常は “010”(CAS Latency = 2) に設定)
11 – 7	“00000” に固定

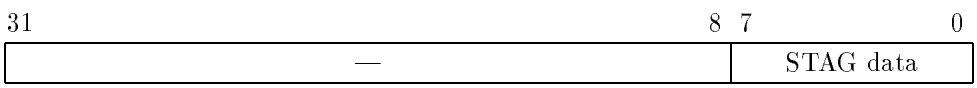
モードレジスタの詳細については, Synchronous DRAM(NEC: μPD4516821G5 - A10) のマニュアルを参照のこと.

ただし, 現在の MMC の実装では, リセット時に Synchronous DRAM のモードレジスタに “00000010010”(CAS Latency = 2, Wrap Type = Sequential, Burst Length = 4) の値を書き込むのみであり, このレジスタに値を書き込んでもそれは Synchronous DRAM に対して反映されない.

4.6 SRAM コントローラ (SCTL)

4.6.1 r_data: STAG データ

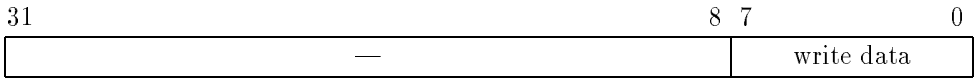
Address = 0x200000, Read Only



SRAM から読み込まれた Tag データ. データの意味については 「JUMP-1 メモリコントローラ仕様説明書」を参照のこと.

4.6.2 w_data: STAG write データ

Address = 0x200008, Read Only

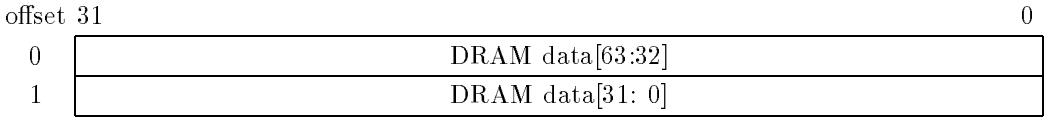


SRAM に書き込んだ (またはこれから書き込む) データを保存しているレジスタ.

4.7 DRAM コントローラ (DCTL)

4.7.1 r_data: DRAM データ

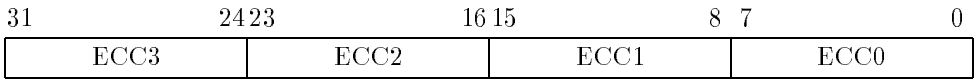
Address = 0x280000 – 0x280001, Read Only



DRAM から読み込まれたデータ. 最後に読んだ Double Word が保存してある.

4.7.2 r_ecc: ECC データ

Address = 0x280008, Read Only



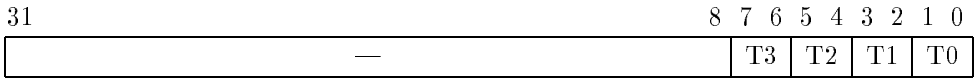
DRAM Data に付随する ECC データ. 最後に読み込みを行なった block の内容が保存されている. 下位から 8 ビットごとに, word 0 の ECC データ, word 1 の ECC データ, word 2 の ECC データ, word 3 の ECC データという順に並んでいる.

エラー発生時には, DCTL は停止状態になるので, エラーの状態が保存される.

MMC では, ハードウェアによるエラー訂正機能は持っていないため, ECC エラーが発生した場合には, DSP はこの ECC データまたは Syndrome データ (Address = 0x280018) を元にエラー訂正を行なわなくてはならない.

4.7.3 r_tag: DTAG データ

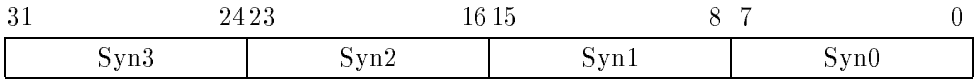
Address = 0x280010, Read Only



DRAM Data に付随する Tag データ. 最後に読み込みを行なった block の内容が保存されている. 下位から 2 ビットごとに, word 0 の Tag データ, word 1 の Tag データ, word 2 の Tag データ, word 3 の Tag データという順に並んでいる.

4.7.4 r_syn: Syndrome データ

Address = 0x280018, Read Only



DRAM Data およびそれに付随する ECC データから生成される Syndrome データ. 最後に読み込みを行なった block の内容が保存されている. 下位から 8 ビットごとに, word 0 の Syndrome データ, word 1 の Syndrome データ, word 2 の Syndrome データ, word 3 の Syndrome データという順に並んでいる.

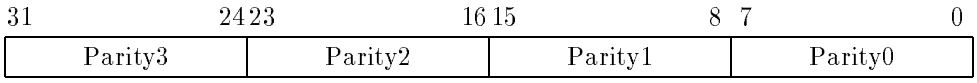
Syndrome データは, 正常な場合は常に 0 になる. エラーが発生すると, エラービットの状態に応じて 1 が立つ. 1 であるビット数が 3 である場合には, 1 ビットのエラーを表す (訂正可能なエラー).

エラー発生時には, DCTL は停止状態になるので, エラーの状態が保存される.

MMC では, ハードウェアによるエラー訂正機能は持っていないため, ECC エラーが発生した場合には, DSP はこの Syndrome データまたは ECC データ (Address = 0x280008) を元にエラー訂正を行なわなくてはならない.

4.7.5 r_parity: Parity データ

Address = 0x280020, Read Only

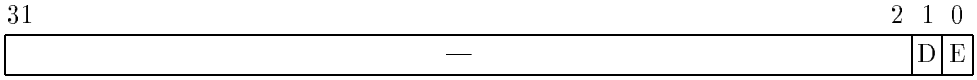


DRAM Data から生成される Parity データ. 最後に読み込みを行なった block の内容が保存されている. 下位から 8 ビットごとに, word 0 の Parity データ, word 1 の Parity データ, word 2 の Parity データ, word 3 の Parity データという順に並んでいる.

エラー発生時には, DCTL は停止状態になるので, エラーの状態が保存される.

4.7.6 error: DCTL エラーステートレジスタ

Address = 0x280028, Read Only



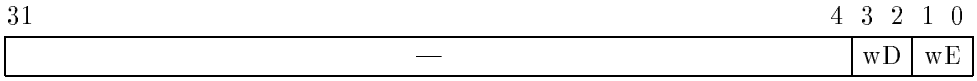
DRAM Data の ECC エラー検出を表すレジスタ. DSP に割り込みがかかった時に, このレジスタの E ビットが 1 の場合は ECC エラーによる割り込みであることを表す.

D ビットはダブルエラービットであり, このビットが 1 であると, ECC エラー検出状態でさらに ECC エラーを検出した状態を表す.

ECC エラー検出状態は, 割り込み解除レジスタ (Address = 0x380008) により解除することができる.

4.7.7 error_num: DCTL エラーワードレジスタ

Address = 0x280030, Read Only

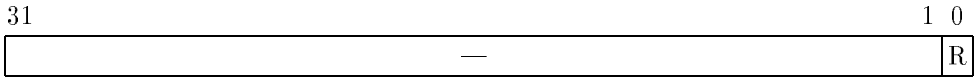


DRAM Data の ECC エラーがどこで発生したかを表すレジスタ. wE には, ECC エラーが検出されたワード番号が入っている.

wD はダブルエラーワード番号であり, ECC エラー検出状態でさらに ECC エラーが検出されたワード番号が入っている.

4.7.8 refresh: リフレッシュインジケータ

Address = 0x280038, Read Only



DRAM のリフレッシュ状態を表すレジスタ. R ビットが 1 である場合は, DRAM がリフレッシュ状態であることを示す. リフレッシュの間隔は, リフレッシュ参照レジスタ (Address = 0x380010) により設定することができる.

4.8 DSP インタフェース (DSPIF)

4.8.1 adr_flit: アドレスレジスタ

Address = 0x300000, Read Only

31	0
Address	

DSP からのアクセスアドレスを保持しているレジスタ.

4.8.2 dbuf: DSP データバッファ

Address = 0x3000008 – 0x30000F, Read Only

offset	31 0
0	dbuf(0)
1	dbuf(1)
2	dbuf(2)
3	dbuf(3)
4	dbuf(4)
5	dbuf(5)
6	dbuf(6)
7	dbuf(7)

DSP からの書き込みデータを一時保存しているバッファ. DSP からは 32 ビット単位でデータアクセスし, ブロックモード時も途中中断の可能性があるので, DSPIF では内部にデータバッファを持ち, メモリに書き込みアクセスする場合には Read-Modify-Write を行なう.

4.8.3 dbuf_fe: DSP データバッファfull/empty

Address = 0x300010, Read Only

31	8	7	0
—		dbuf_fe	

DSP データバッファの Full/Empty 状態を保持しているレジスタ. 1 が該当する dbuf に有効なデータが入っていることを示し, 0 が該当するデータは無効なデータであることを示す. DSPIF では, DSP からのメモリ書き込みアクセス時の Modify の時にこのビットを参照する.

ビット 0 が dbuf(0) に, ビット 1 が dbuf(1) に, . . ., ビット 7 が dbuf(7) に対応している.

4.9 その他

4.9.1 int_src: 割り込み原因レジスタ

Address = 0x380000, Read Only

31	3	2	1	0
—		D	M	S

DSP に対して割り込みが発生した時に, その原因となるブロックを示すレジスタ.

S ビットが 1 の場合は, クラスタバスコントローラ (CBC) 内部で割り込みが発生したことを表す. この時には, 割り込み原因バッファ番号レジスタ (Address = 0x080310) にその原因となるパケットが収められている入力バッファ(BUFX) の番号が格納されている. クラスタバスコントローラ (CBC) で割り込みが発生する要因には

- 割り込みを発生するパケットを受信した
- Reply パケットのアドレスがロックアドレスと一致した

がある. 割り込みを発生するパケットの詳細については, 「JUMP-1 メモリコントローラ仕様説明書」を参照のこと.

M ビットが 1 の場合は, メモリタイミングコントローラ (MTC) 内部で割り込みが発生したことを表す. この時には, 処理中アドレスフリット (Address = 0x100000) に割り込み発生の原因となるパケットのアドレスフリット, 処理中バッファ番号 (Address = 0x100008 に割り込み発生の原因となるパケットが格納されている入力バッファ(BUFX) の番号, STAG データ (Address = 0x200000) にそのアドレスの STAG データが格納されている. メモリタイミングコントローラで割り込みが発生する要因には

- SRAM Tag によるトラップ
- block, double word 以外のサイズの書き込みアクセス

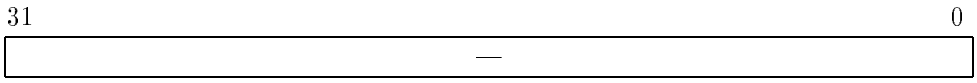
がある. SRAM Tag によるトラップ条件の詳細については, 「JUMP-1 メモリコントローラ仕様説明書」を参照のこと.

D ビットが 1 の場合は, DRAM コントローラ (DCTL) 内部で ECC エラーを検出したことを表す. この時には, DCTL エラーワードレジスタ (Address = 0x280030) の wE に, エラーが検出されたワード番号, ECC データ (Address = 0x280008) に ECC, Syndrome データ (Address = 0x280018) に Syndrome, Parity データ (Address = 0x280020) に Parity が格納されている.

MMC の割り込み状態は, 割り込み解除レジスタ (Address = 0x380008) により解除することができる.

4.9.2 int_reset: 割り込み解除レジスタ

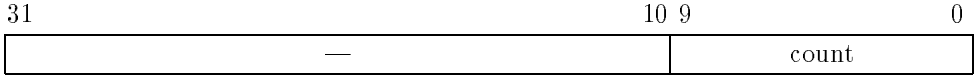
Address = 0x380008, Write Only



MMC の割り込み状態を解除する. このレジスタに任意の値を書き込むと (書き込み動作を行なうと), MMC 内部の全ての割り込み状態を解除する.

4.9.3 to_refresh: リフレッシュ参照レジスタ

Address = 0x380010, Read Write



DRAM リフレッシュの間隔を設定するレジスタ. リフレッシュカウンタ (Address = 0x380018) がここで設定した値に達すると, DRAM コントローラに対してリフレッシュ要求が出る.

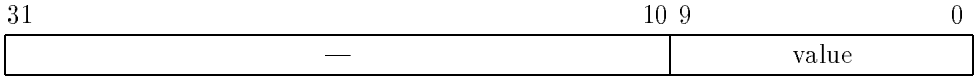
μPD4516821G5 - A10 は, 32ms 以内に 2048row をリフレッシュしなくてはならない. したがって, リフレッシュの間隔は 15.6μs になる. MMC の動作周波数は 50MHz であるから, 15.6μs は 781 クロックに相

当する. したがって, 50MHz 動作の場合のリフレッシュ参照レジスタは, 少なくとも 771 以下でなくてはならない (DRAM コントローラがリフレッシュ要求を受けた時に動作中である可能性を考えると, 少なくとも 10 クロックは余裕を見なくてはならない).

リフレッシュ参照レジスタの初期値は 0x300(=768) である.

4.9.4 refresh_counter: リフレッシュカウンタ

Address = 0x380018, Read Only



現在のリフレッシュカウンタの値. この値がリフレッシュ参照レジスタ (Address = 0x380010) で設定された値に達すると, DRAM コントローラ (DCTL) に対してリフレッシュ要求が出る.

5 DSP からのアクセス手順

ここでは, DSP から MMC の内部資源を用いたアクセスを行なう場合の手順について, いくつかの例を示す.

5.1 クラスタ共有バスにパケットを出力する手順

クラスタ共有バスへ DSP からパケットを送り出す時は, 以下のような手順を用いる.

1. データつきパケットの場合
   (a) 出力バッファ(BUFY: Address = 0x000040) の 4 番 (Reply Paket の場合) または 5 番 (Request Packet の場合) に, データを書き込む.
   (b) 出力 WWE レジスタ (Address = 0x080180) の 4 番 (Reply Paket の場合) または 5 番 (Request Packet の場合) に, WWE データを書き込む.
   (c) 出力アドレスレジスタ (Address = 0x080100)4 番 (Reply Paket の場合) または 5 番 (Request Packet の場合) に, 送出すべきパケットのアドレスフリットを書き込む.
   (d) MMC が割り込み状態でなければ, パケットが自動的に送出される.
   (e) 確実に送出されたかどうかの確認は, 出力バッファステート (Address = 0x080208) の該当するビットを見て判断する
2. データなしパケットの場合
   (a) 出力アドレスレジスタ (Address = 0x080100)4 番 (Reply Paket の場合) または 5 番 (Request Packet の場合) に, 送出すべきパケットのアドレスフリットを書き込む.
   (b) MMC が割り込み状態でなければ, パケットが自動的に送出される.
   (c) 確実に送出されたかどうかの確認は, 出力バッファステート (Address = 0x080208) の該当するビットを見て判断する

5.2 割り込み発生時の手順

MMC から DSP に対して割り込みが発生した場合は, 以下のような手順を用いて解決する.

1. 割り込み原因レジスタ (Address = 0x380000) を検査して, どのブロックからの割り込みかを調べる.
   (a) クラスタバスコントローラ (S=1) の場合.
      i. 割り込み原因バッファ番号レジスタ (Address = 0x080310) を見て, どのバッファにあるパケットが割り込みを発生したのかを調べる.
      ii. 入力アドレスレジスタ (Address = 0x080000), 入力 WWE レジスタ (Address = 0x080080) および入力バッファ(BUFX: Address = 0x000000) からパケットデータを読み出す.
      iii. 必要があれば, ロックアドレス (Address = 0x080300) を設定する.
      iv. 必要があれば, MTC 要求ステート (Address = 0x080210) を調べて, 処理中のパケットがなくなるまで待つ.
      v. 割り込み解除レジスタ (Address = 0x380008) をアクセスし, 割り込みを解除する.
      vi. DSP によりパケットの処理を行なう.
      vii. ロックアドレスクリアレジスタ (Address = 0x080308) をアクセスし, ロックアドレスを解除する.
      viii. 必要があれば, PEND ダウンレジスタ (Address = 0x080420) をアクセスし, PEND カウンタを−1 する.
      ix. 必要があれば, クラスタ共有バスに対してパケットを送出する (クラスタ共有バスにパケットを出力する手順を参照).
   (b) メモリタイミングコントローラ (M=1) の場合.
      i. 処理中アドレスフリット (Address = 0x1000000), 処理中バッファ番号 (Address = 0x100008) により, 割り込みを起こしたパケットを調べる.
      ii. 必要があれば, 入力 WWE レジスタ (Address = 0x080080) および入力バッファ(BUFX: Address = 0x000000) からパケットデータを読み出す.
      iii. 必要があれば, ロックアドレス (Address = 0x080300) を設定する.
      iv. 必要があれば, バス入力拒否レジスタ (Address = 0x080500) を設定し, クラスタ共有バスからのパケットを受け取らないようにする.
      v. 割り込み解除レジスタ (Address = 0x380008) をアクセスし, 割り込みを解除する.
      vi. 必要があれば, MTC 要求ステート (Address = 0x080210) を調べて, 処理中のパケットがなくなるまで待つ.
      vii. DSP によりパケットの処理を行なう.
      viii. ロックアドレスクリアレジスタ (Address = 0x080308) をアクセスし, ロックアドレスを解除する.
      ix. バス入力拒否レジスタ (Address = 0x080500) を設定し, 再びクラスタ共有バスからのパケットを受け取るようにする.
      x. 必要があれば, PEND ダウンレジスタ (Address = 0x080420) をアクセスし, PEND カウンタを−1 する.
      xi. 必要があれば, クラスタ共有バスに対してパケットを送出する (クラスタ共有バスにパケットを出力する手順を参照).

(c) DRAM コントローラ (D=1) の場合.

i. DCTL エラーワードレジスタ (Address = 0x280030) で, エラー検出が生じたワードアドレスを調べる.

ii. ECC データ (Address = 0x280008), Syndrome データ (Address = 0x280010), Parity データ (Address = 0x280020) を読みだし, それらのデータを元にソフトウェアによるエラー訂正を行なう.

iii. リプライアドレスフリット (Address = 0x100010), DTAG データ (Address = 0x280010) により, 返信しなくてはならないリプライパケットのアドレスフリット, WWE データを調べる.

iv. クラスタ共有バスに対して, 訂正したデータと, 上で得られたアドレスフリットおよび WWE データを持つリプライパケットを送出する (クラスタ共有バスにパケットを出力する手順を参照).

v. 割り込み解除レジスタ (Address = 0x380008) をアクセスし, 割り込みを解除する.

5.3 メモリに対する単一操作の手順

メモリの特定ブロックに対して単一操作を行なう場合は, 以下のような手順を用いる.

1. バス入力拒否レジスタ (Address = 0x080500) を設定し, クラスタ共有バスからパケットを受け取らないようにする.
2. ロックアドレス (Address = 0x080300) を設定する.
3. MTC 要求ステート (Address = 0x080210) を調べて, 処理中のパケットがなくなるまで待つ.
4. DSP によりメモリに対する単一操作を行なう.
5. バス入力拒否レジスタ (Address = 0x080500) を設定し, 再びクラスタ共有バスからのパケットを受け取るようにする.
6. ロックアドレスクリアレジスタ (Address = 0x080308) をアクセスし, ロックアドレスを解除する.

6 ECC コード

MMC では, DRAM Data に対して ECC コードを付加する. この ECC コードは 2 ビットエラー検出, 1 ビットエラー訂正コードである. MMC には自動エラー訂正機能はないので, エラーが検出された場合, DSP がソフトウェアによりエラー訂正しなくてはならない. 本節では, DSP がエラー訂正を行なうための ECC コードについて説明する.

6.1 ECC

MMC で用いられる ECC は, MIPS R4000 のデータ用 ECC と同じものである. この ECC コードには以下のような特徴がある.

- 1 ビット訂正, 2 ビット検出.
- 64 ビットデータに対して 8 ビットの ECC コード.

- パリティビットは最低 27 入力 XOR が必要.
- パリティジェネレータ用のパリティツリー内にバイトパリティが含まれる.
- 1 ビットエラーは 3 つの 1 を含むシンドロームで示されるか, 5 つの 1 を含むシンドローム (この場合, シンドロームのビット 0~3, またはビット 4~7 はすべて 1 である) で示される.
- 2 ビットエラーは偶数個の 1 を含むシンドロームで示される.

6.2 パリティチェックマトリクス

MMC で用いられる ECC コードのパリティチェックマトリクスを, 表 4に示す.

参考資料

- 「UMP-1 メモリコントローラ仕様説明書」
- 「UMP-1 共通バス仕様書」
- 「UMP-1 共用バッファ仕様書」

表 4: ECC Parity Matrix

ECC	Data	7	6	5	4	3	2	1	0	# of 1
	63	1	1	0	0	1	0	0	0	3
	62	1	1	0	0	0	1	0	0	3
	61	1	1	0	0	0	0	1	0	3
	60	1	1	0	0	0	0	0	1	3
	59	1	1	1	1	0	1	0	0	5
	58	1	0	0	0	1	1	1	1	5
4		0	0	0	1	0	0	0	0	1
3		0	0	0	0	1	0	0	0	1
	57	1	1	1	0	0	0	0	0	3
	56	1	0	1	1	0	0	0	0	3
	55	0	0	0	0	1	1	1	0	3
	54	0	0	0	0	1	0	1	1	3
	53	1	1	1	1	0	0	1	0	5
	52	0	0	0	1	1	1	1	1	5
5		0	0	1	0	0	0	0	0	1
2		0	0	0	0	0	1	0	0	1
	51	1	0	0	0	1	1	0	0	3
	50	0	1	0	0	1	1	0	0	3
	49	0	0	1	0	1	1	0	0	3
	48	0	0	0	1	1	1	0	0	3
	47	0	0	1	1	1	0	0	0	3
	46	0	0	1	1	0	1	0	0	3
	45	0	0	1	1	0	0	1	0	3
	44	0	0	1	1	0	0	0	1	3
	43	1	0	1	0	1	0	0	0	3
	42	1	0	1	0	0	1	0	0	3
	41	1	0	1	0	0	0	1	0	3
	40	1	0	1	0	0	0	0	1	3
	39	1	0	0	1	1	0	0	0	3
	38	1	0	0	1	0	1	0	0	3
	37	1	0	0	1	0	0	1	0	3
	36	1	0	0	1	0	0	0	1	3
	35	0	1	0	1	1	0	0	0	3
	34	0	1	0	1	0	1	0	0	3
	33	0	1	0	1	0	0	1	0	3
	32	0	1	0	1	0	0	0	1	3
	31	1	0	0	0	1	0	1	0	3
	30	0	1	0	0	1	0	1	0	3
	29	0	0	1	0	1	0	1	0	3
	28	0	0	0	1	1	0	1	0	3
	27	1	0	0	0	1	0	0	1	3
	26	0	1	0	0	1	0	0	1	3
	25	0	0	1	0	1	0	0	1	3
	24	0	0	0	1	1	0	0	1	3
	23	1	0	0	0	0	1	0	1	3
	22	0	1	0	0	0	1	0	1	3
	21	0	0	1	0	0	1	0	1	3
	20	0	0	0	1	0	1	0	1	3
	19	1	0	0	0	1	1	0	0	3
	18	0	1	0	0	1	1	0	0	3
	17	0	0	1	0	1	1	0	0	3
	16	0	0	0	1	1	1	0	0	3
	15	0	1	1	0	1	0	0	0	3
	14	0	1	1	0	0	1	0	0	3
	13	0	1	1	0	0	0	1	0	3
	12	0	1	1	0	0	0	0	1	3
	11	1	1	1	1	1	0	0	0	5
	10	0	1	0	0	1	1	1	1	5
7		1	0	0	0	0	0	0	0	1
0		0	0	0	0	0	0	0	1	1
	9	0	1	1	1	0	0	0	0	3
	8	1	1	0	1	0	0	0	0	3
	7	0	0	0	0	0	1	1	1	3
	6	0	0	0	0	1	1	0	1	3
	5	1	1	1	1	0	0	0	1	5
	4	0	0	1	0	1	1	1	1	5
6		0	1	0	0	0	0	0	0	1
1		0	0	0	0	0	0	1	0	1
	3	1	0	0	0	0	0	1	1	3
	2	0	1	0	0	0	0	1	1	3
	1	0	0	1	0	0	0	1	1	3
	0	0	0	0	1	0	0	1	1	3