

CONFIDENTIAL

TWL
プログラミングマニュアル
V1.5



任天堂株式会社発行

このドキュメントの内容は、機密情報である
ため、厳重な取り扱い、管理を行ってください。

改訂履歴

バージョン	更新日時	改訂内容
V1.5	2012/4/16	<ul style="list-style-type: none"> 21 章カメラの画角情報を変更。注意事項を追加
V1.4	2009/1/29	<ul style="list-style-type: none"> 図 3-1,図 3-5,図 3-9 の転送シーケンスの誤記を修正
V1.3	2008/12/19	<ul style="list-style-type: none"> 図 1-1 の ARM7 バスと外部メモリインタフェースを繋ぐ線を追加 表 3-6 の VRAM からメインメモリへの DMA 転送速度の誤記を修正 図 3-1 のサイクル数を修正 図 3-3,図 3-5,図 3-7,図 3-9,図 3-11 にターミネーションが入るサイクルを追加 図 3-4,図 3-8,図 3-12 (ワーストケースの転送シーケンス図) を新規追加 6 章表示コントロールレジスタの表示選択フラグの設定から切り替わりまでのラグについて注意事項を追加 7 章ジオメトリコマンドの実行サイクル数について補足説明を追加 21 章カメラの画角情報を追加。「焦点距離」を「撮影可能範囲」に変更 23 章リード時のウェイトに関する注意事項を追加
V1.2	2008/10/30	<ul style="list-style-type: none"> Nintendo ロゴを一部削除 図 1-2 及び図 1-3 のアドレス表記を修正 付録 C のパレットデータを修正、注意書きを追加
V1.1	2008/10/27	<ul style="list-style-type: none"> Nintendo ロゴを修正 6.5 節 カラーデータフォーマットを修正、注意事項を追加 17.1.4 節 マイクの入力値に関する記述を修正 22 章 DSP の「コマンド」を表す略称を「COM」から「CMD」に修正
V1.0	2008/10/1	<ul style="list-style-type: none"> 初版発行

プログラミングマニュアルでの表記について

■レジスタ表記

上段を細かい区分に分け、大まかな区分は下段に表記します。

文字が入りきらない場合には、下図「イネーブルフラグ」のように表記します。

また、特定のビットを表す際、例えば 16 ビットレジスタの最上位ビットを「d15」と表記します。

表記例) TWL レジスタ

名称 TWL アドレス 0x04000??? 属性 R/W 初期値 0x0000

15	8	7	0
E1	E0		
		BLUE	GREEN
			RED
画像モード			
カラー			

イネーブルフラグ

■ビット長

バイト、ハーフワード、ワードのビット長を次のように定義します。

8bit : バイト

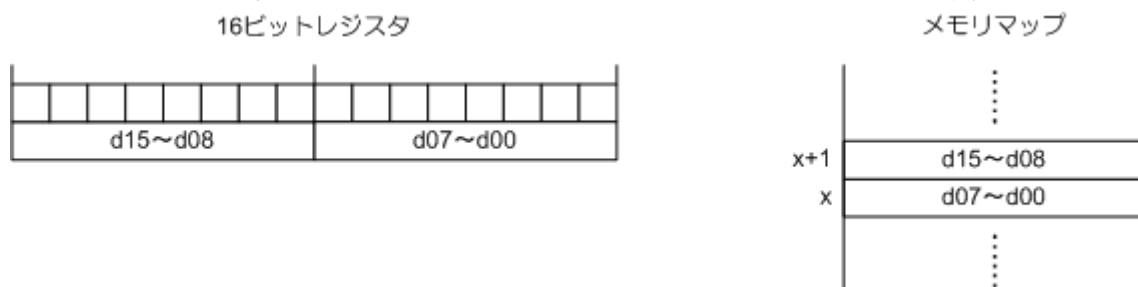
16bit : ハーフワード

32bit : ワード

■エンディアン

TWL はリトルエンディアンです。

例えば 16 ビットレジスタの場合、d15~d08 は d07~d00 よりも一つ大きいアドレスになります。



目 次

1. システム	13
1.1 システム概要	13
1.1.1 TWL プロセッサ	14
1.1.2 メインメモリ	16
1.1.3 LCD	16
1.1.4 デジタルキー	16
1.1.5 タッチパネル	16
1.1.6 マイク	16
1.1.7 RTC	16
1.1.8 ワイヤレス通信	17
1.1.9 TWL / NITRO カード	17
1.1.10 カートリッジ	17
1.1.11 SD メモリカード	17
1.1.12 NAND フラッシュメモリ	17
1.1.13 カメラ	17
1.1.14 DSP(Digital Signal Processor)	17
1.2 メモリマップ	18
1.3 サブプロセッサへ接続されているデバイスへのアクセス	20
1.4 起動モード	20
1.4.1 TWL モード	20
1.4.2 NITRO 互換モード	20
1.5 仕向地	20
2. NITRO との違い	21
2.1 新たに追加されたモジュール	21
2.2 カートリッジスロット	21
2.3 変更のあったモジュール	21
2.4 システム設定	22
2.4.1 システム ROM	22
2.4.2 新規ブロックのクロック制御	23
2.4.3 新規ブロックのリセット	23
2.4.4 拡張機能の制御	24
2.4.4.1 拡張される機能	25
2.4.4.2 修正される機能	26
2.4.5 新規共有 WRAM	26
3. メモリ	27
3.1 外部メモリ	30
3.1.1 メインメモリ	31
3.1.1.1 先読みバッファ	31
3.1.1.2 バーストモード	31
3.1.1.2.1 バーストアクセスの条件	31
3.1.1.2.2 ウェイトが追加される条件	31
3.1.1.2.3 メインメモリの DMA 転送サイクル	32

3.1.1.2.3.1	メインメモリからワーク RAM への 32 ビット DMA 転送サイクル	32
3.1.1.2.3.2	ワーク RAM からメインメモリへの 32 ビット DMA 転送サイクル	33
3.1.1.2.3.3	メインメモリから VRAM(16 ビット幅)への 32 ビット DMA 転送サイクル	34
3.1.1.2.3.4	VRAM(16 ビット幅)からメインメモリへの 32 ビット DMA 転送サイクル	35
3.1.1.2.3.5	メインメモリから VRAM(32 ビット幅)への 32 ビット DMA 転送サイクル	36
3.1.1.2.3.6	VRAM(32 ビット幅)からメインメモリへの 32 ビット DMA 転送サイクル	37
3.2	TWL プロセッサ内メモリ	38
3.2.1	VRAM	38
3.2.2	ワーク RAM	47
3.2.2.1	WRAM-0/1	47
3.2.2.2	WRAM-A	49
3.2.2.3	WRAM-B	51
3.2.2.4	WRAM-C	54
3.2.3	TWL-SDK におけるワーク RAM の設定	57
3.2.4	I/O レジスタ	57
3.3	カードブート時のメモリマップ	58
4.	メインプロセッサコア (ARM946E-S)	60
4.1	プロテクションユニット	61
4.2	TCM (Tightly-coupled Memory)	62
4.2.1	命令 TCM	62
4.2.2	データ TCM	62
4.3	キャッシュメモリ	63
4.3.1	命令キャッシュ	64
4.3.1.1	ヒット/ミス判定	64
4.3.2	データキャッシュ	65
4.3.2.1	ヒット/ミス判定	65
4.3.3	キャッシュに対するオペレーション	67
4.3.4	キャッシュの最適化	68
4.4	ライトバッファ	69
4.4.1	ライトバッファに関するオペレーション	69
4.5	コヒーレンシの確保	70
4.5.1	ライトバックモード時	70
4.5.2	ライトスルーモード時	71
5.	表示	72
5.1	表示システム	72
5.2	LCD	74
5.2.1	LCD コントローラ仕様	74
5.3	表示ステータス	75
5.4	表示制御	77
5.4.1	上画面 LCD / 下画面 LCD 出力先切り換え	77
5.4.2	2D グラフィックスエンジン A の表示制御	78
5.4.3	2D グラフィックスエンジン B の表示制御	80
5.4.4	表示モード	81
5.4.4.1	グラフィックス表示モード	82
5.4.4.2	VRAM 表示モード	84
5.4.4.3	メインメモリ表示モード	86
5.5	表示キャプチャ	87

5.6	マスター輝度.....	90
6.	2D グラフィックス.....	91
6.1	2D 表示制御	92
6.2	BG	94
6.2.1	BG モード	94
6.2.1.1	2D グラフィックスエンジン A	94
6.2.1.2	2D グラフィックスエンジン B	95
6.2.1.3	BG タイプ別の基本的な特徴.....	96
6.2.1.4	BG タイプ別仕様一覧.....	97
6.2.2	BG コントロール.....	98
6.2.2.1	スクリーンサイズと表示画面	102
6.2.2.1.1	テキスト BG	102
6.2.2.1.2	アフィン BG	103
6.2.3	キャラクタ BG	104
6.2.3.1	BG データの VRAM マップ	105
6.2.3.2	テキスト BG	107
6.2.3.2.1	スクリーンデータフォーマット	107
6.2.3.2.2	スクリーンデータのアドレスマッピング	108
6.2.3.2.3	キャラクタデータフォーマット	110
6.2.3.2.3.1	16 色モード	110
6.2.3.2.3.2	256 色モード	111
6.2.3.3	アフィン BG	112
6.2.3.3.1	スクリーンデータフォーマット	112
6.2.3.3.2	スクリーンデータのアドレスマッピング	113
6.2.3.3.3	キャラクタデータフォーマット	116
6.2.3.4	256 色×16 パレットキャラクタ BG (アフィン拡張 BG)	117
6.2.3.4.1	スクリーンデータフォーマット	117
6.2.3.4.2	キャラクタデータフォーマット	118
6.2.4	ビットマップ BG	119
6.2.4.1	256 色ビットマップ BG (アフィン拡張 BG)	119
6.2.4.1.1	ピクセルデータフォーマット	119
6.2.4.1.2	ピクセルデータの VRAM マップ	119
6.2.4.2	ダイレクトカラービットマップ BG (アフィン拡張 BG)	119
6.2.4.2.1	ピクセルデータフォーマット	119
6.2.4.2.2	ピクセルデータの VRAM マップ	119
6.2.4.3	大画面 256 色ビットマップ BG	120
6.2.4.3.1	ピクセルデータフォーマット	120
6.2.4.3.2	ピクセルデータの VRAM マップ	120
6.2.5	BG スクロール	121
6.2.6	BG の回転拡大縮小 (アフィン変換)	122
6.3	OBJ	124
6.3.1	OBJ 表示制御	126
6.3.2	OAM	127
6.3.2.1	メモリマップ	127
6.3.2.2	OAM データフォーマット	128
6.3.2.3	OBJ の回転拡大縮小 (アフィン変換)	133
6.3.3	キャラクタ OBJ	134
6.3.3.1	キャラクタデータフォーマット	136
6.3.3.1.1	16 色モード	136
6.3.3.1.2	256 色モード	137
6.3.3.2	キャラクタ OBJ データ・マッピングモード	138
6.3.3.2.1	2 次元マッピング	138
6.3.3.2.2	1 次元マッピング	139
6.3.4	ビットマップ OBJ	141
6.3.4.1	ビットマップ OBJ データ	143
6.3.4.2	BG とのブレンドリング	143

6.3.4.3	ビットマップ OBJ データ・マッピングモード	144
6.3.4.3.1	横 128 ドットで 2 次元マッピング	144
6.3.4.3.2	横 256 ドットで 2 次元マッピング	145
6.3.4.3.3	1 次元マッピング	146
6.4	バックドロップ	147
6.5	カラーパレット	148
6.5.1	標準パレット	148
6.5.2	拡張パレット	149
6.5.2.1	BG 拡張パレット	149
6.5.2.2	OBJ 拡張パレット	152
6.6	ウィンドウ	154
6.6.1	ウィンドウの優先順位	156
6.7	カラー特殊効果	157
6.8	モザイク	160
6.9	表示優先順位	161
7.	3D グラフィックス	162
7.1	3D 表示制御	164
7.2	ジオメトリエンジン	166
7.2.1	概要	166
7.2.2	座標系	166
7.2.3	座標変換	167
7.2.4	射影変換	169
7.2.5	デプスバッファリング	170
7.2.6	ジオメトリコマンド	173
7.2.7	レンダリングエンジン参照データのスワップ	183
7.2.8	ビューポート	184
7.2.9	行列	185
7.2.9.1	カレント行列の操作	185
7.2.9.2	行列スタック	189
7.2.9.3	カレント行列の読み出し	191
7.2.10	ライト	192
7.2.11	マテリアル	193
7.2.11.1	ライティング（照光処理）	197
7.2.12	ポリゴン属性	198
7.2.13	ポリゴン	201
7.2.14	テクスチャマッピング	206
7.2.14.1	テクスチャ座標変換	211
7.2.15	テスト	214
7.2.16	ステータス	216
7.2.16.1	ポリゴンリスト RAM と頂点 RAM のデータ格納数	219
7.2.17	計算精度の注意点	222
7.3	レンダリングエンジン	223
7.3.1	概要	223
7.3.2	レンダリング方式	225
7.3.2.1	ラインバッファレンダリング	225
7.3.2.2	レンダリングエンジン内のバッファ	225
7.3.2.3	ブランク期間	226
7.3.2.4	1 ラインで描画可能なポリゴン数	227
7.3.3	レンダリングバッファ初期化	228

7.3.3.1	クリアレジスタによる初期化	228
7.3.3.2	クリアイメージによる初期化	229
7.3.4	ラスタライズ	231
7.3.4.1	不透明ポリゴン	231
7.3.4.2	半透明ポリゴン	231
7.3.4.2.1	1 ≤ α ≤ 30 のポリゴン	231
7.3.4.2.2	半透明テクスチャを貼ったポリゴン	232
7.3.4.3	ワイヤーフレーム	232
7.3.4.4	シャドウポリゴン	233
7.3.4.5	トゥーン/ハイライト シェーディング	236
7.3.4.5.1	トゥーンシェーディング	237
7.3.4.5.2	ハイライトシェーディング	238
7.3.5	テクスチャ	239
7.3.5.1	テクスチャブレンディング	239
7.3.5.1.1	テクスチャイメージのサンプリング	239
7.3.5.1.2	デカルモード	241
7.3.5.1.3	モジュレーションモード	242
7.3.5.2	テクスチャフォーマット	243
7.3.5.2.1	テクスチャイメージ	243
7.3.5.2.1.1	4 色パレットテクスチャ	243
7.3.5.2.1.2	16 色パレットテクスチャ	244
7.3.5.2.1.3	256 色パレットテクスチャ	244
7.3.5.2.1.4	4x4 テクセル圧縮テクスチャ	245
7.3.5.2.1.5	A3I5 半透明テクスチャ	248
7.3.5.2.1.6	A5I3 半透明テクスチャ	248
7.3.5.2.1.7	ダイレクトカラーテクスチャ	249
7.3.5.2.2	テクスチャパレット	250
7.3.6	α テスト	253
7.3.7	α ブレンディング	253
7.3.7.1	3D の α ブレンディング	254
7.3.7.2	2D と 3D の α ブレンディングの前処理	254
7.3.8	エッジマーキング	254
7.3.9	フォグブレンディング	255
7.3.9.1	3D のフォグ	255
7.3.9.1.1	フォグ濃度計算式	256
7.3.9.2	2D へのフォグの前処理	257
7.3.10	アンチエイリアシング	258
7.3.11	ステータス	261
7.4	レンダリング後の 3D 面に適用できる 2D グラフィックス機能	262
7.4.1	ラスタスクロール	262
7.4.2	2D 面との表示優先順位	262
7.4.3	ウィンドウ	263
7.4.4	カラー特殊効果	264
7.4.4.1	2D 面との α ブレンディング	264
7.4.4.2	輝度アップ/ダウン	265
8.	DMA	266
8.1	NITRO 互換 DMA	266
8.2	新 DMA	271
9.	タイマ	276
10.	割り込み	278
10.1	割り込みマスタイネーブルレジスタ	278
10.2	割り込みイネーブルレジスタ	279

10.3	割り込みリクエストレジスタ	280
10.4	割り込みに関する注意事項	281
10.4.1	IME および IE のクリア	281
10.4.2	多重割り込み	281
10.4.3	DMA 動作中の割り込み遅延	281
10.4.4	ARM7 からの割り込み	281
10.4.5	DSP からの割り込み	281
10.4.6	カメラからの割り込み	281
11.	パワーマネジメント	282
11.1	スリープモード	282
11.2	各種電源制御	283
11.2.1	サウンド	283
11.2.2	LCD バックライト	283
11.2.3	LCD	283
11.2.4	マイク	283
11.2.5	システム	283
11.2.6	グラフィックス	284
11.3	パワーステータス	286
11.3.1	バッテリー低下状態とバッテリー残量	286
11.3.2	TWL 本体の開閉状態	286
12.	アクセラレータ	288
12.1	除算器（ディバイダ）	288
12.1.1	計算サイクル数	289
12.2	平方根演算器	290
12.2.1	計算サイクル数	290
13.	キー	291
13.1	キー入力	291
13.2	キー入力への割り込み処理	292
14.	サウンド	293
14.1	ハードウェアスペック	294
14.1.1	データ形式	294
14.1.1.1	8bitPCM のデータフォーマット	294
14.1.1.2	16bitPCM のデータフォーマット	294
14.1.1.3	IMA-ADPCM のデータフォーマット	295
14.1.1.4	PSG 矩形波	295
14.1.1.5	ノイズ	295
14.1.2	チャンネル	296
14.1.3	ミキサー	296
14.1.3.1	DSP サウンドとの合成	296
14.1.3.2	レジスタボリューム	296
14.1.3.3	CODEC	297
14.1.4	マスターボリューム	297
14.1.5	サウンドキャプチャ	297
14.1.6	注意事項	297
14.2	サウンドブロック図	298

14.2.1	サウンド全体	298
14.2.2	チャンネル 0～3、サウンドキャプチャ 0～1	299
14.2.3	チャンネル 4～7	301
14.2.4	チャンネル 8～15	302
14.2.5	サウンド使用例	303
14.2.5.1	通常使用例	303
14.2.5.2	リバーブ例	304
14.2.5.3	エフェクト例	305
14.3	NITRO-Composer	306
14.3.1	NITRO-Composer の再生形式	306
14.3.1.1	シーケンス再生	306
14.3.1.2	波形再生	306
14.3.1.3	ストリーム再生	306
14.3.1.4	CODEC 内部での遅延について	306
15.	ワイヤレス通信	307
15.1	ハードウェア仕様	307
15.2	ワイヤレスマネージャ	308
15.2.1	インフラストラクチャーモード	308
15.2.2	DS ワイヤレスプレイモード	308
15.2.3	DS ダウンロードプレイモード	308
16.	タッチパネル	309
16.1	タッチパネルの構造	310
17.	マイク	311
17.1	NITRO からの変更点	311
17.1.1	CODEC の起動モード	311
17.1.2	CODEC-TWL モードのマイク	312
17.1.3	CODEC-DS モードのマイク	312
17.1.4	マイクの入力値	313
18.	RTC (リアルタイムクロック)	316
19.	内蔵フラッシュメモリ	317
19.1	タッチパネルのキャリブレーションデータ	317
19.2	オーナー情報データ	317
19.3	TWL 初期設定データ	318
19.4	RTC 操作情報データ	318
20.	AES エンジン	319
20.1	CTR モード	320
20.2	CCM モード	320
20.3	注意事項	322
21.	カメラ	323
21.1	カメラの設定	323

21.2	トリミング	325
21.3	DMA 転送	325
21.4	スリープモードからの復帰	325
22.	DSP	326
22.1	CEVA-Teak コア	326
22.2	DSP メモリ	326
22.3	メインプロセッサとの通信	326
22.3.1	リセット	327
22.3.2	ステータスレジスタ	328
22.3.3	DSP メモリ転送	329
22.3.4	コマンド・リブライレジスタ	330
22.3.5	セマフォレジスタ	331
22.4	DSP の起動手順	332
23.	NAND フラッシュメモリ	333
24.	SD メモリカード	334
	付録	335
	付録 A. レジスター一覧表	336
A.1	0x04000000 番地以降	336
A.2	0x04001000 番地以降（2D グラフィックスエンジン B 関連）	358
A.3	0x04004000 番地以降（TWL 拡張関連）	360
	付録 B. VRAM データ容量一覧表	369
	付録 C. データフォーマット	370

1. システム

1.1 システム概要

TWL のシステム全体のブロック図を図 1-1 に示します。

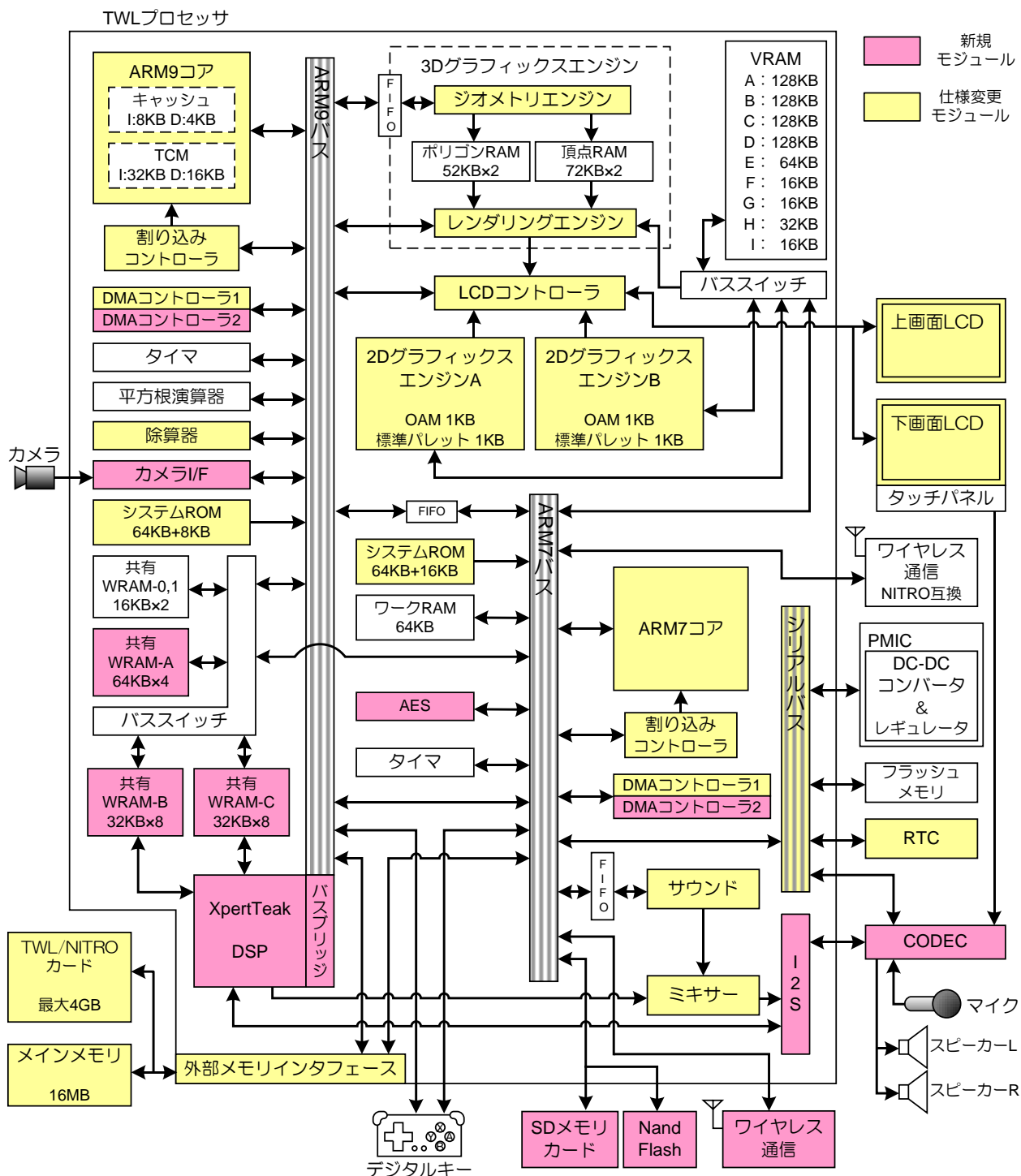


図 1-1 システムブロック全体図

1.1.1 TWL プロセッサ

TWL プロセッサは、ARM9 および ARM7CPU コア、DSP コア、2D および 3D グラフィックスエンジン等の TWL 機能、各メモリを混載した複合チップです。

TWL プロセッサの仕様を以下に示します。

● CMOS マルチ CPU

メインプロセッサコア	ARM946E-S (134.056MHz / 67.028MHz)
サブプロセッサコア	ARM7TDMI (33.514MHz)
DSP コア	TeakDSP (134.056MHz / 67.028MHz / 33.514MHz)
システム設定でメインプロセッサコアの動作周波数を切り換え可能。	

● 互換性

TWL モード、NITRO 互換モードの切り換え。

(NITRO でサポートしていました AGB 互換モードは削除されました)

● グラフィックスエンジン

2D グラフィックスエンジン A, B		33.514MHz
3D グラフィックスエンジン	ジオメトリエンジン	33.514MHz 最大 400 万頂点/秒 4×4 行列演算 6 面クリッピング ライティング (平行光源×4 個) 行列スタック テクスチャ座標変換 ボックススカラーテスト
	レンダリングエンジン	33.514MHz 最大 12 万ポリゴン/秒 最大 3000 万ピクセル/秒 三角形および四角形の描画 テクスチャフォーマット 4、16、256 色パレット形式 ビットマップ形式 4×4 テクセル圧縮形式 半透明 (A3I5、A5I3) 形式 テクスチャサイズ 8×8～1024×1024 アルファブレンド アルファテスト フォグ トゥーンシェーディング エッジマーキング アンチエイリアシング

システム設定で各エンジンの回路修正あり/なしを制御可能。

● LCD コントローラ (上画面、下画面の各 LCD 用に 2 画面分搭載)

表示サイズ	256×192×RGB ドット
表示色数	262,144 色 (R:G:B = 6:6:6)
ドットクロック	5.586MHz

● **メモリ**

システム ROM	ARM9：TWL 用 64K バイト (16K×32 ビット) 互換用 8K バイト (2K×32 ビット) ARM7：TWL 用 64K バイト (16K×32 ビット) 互換用 16K バイト (4K×32 ビット)
TWL プロセッサ 内部ワーク RAM	ARM9,ARM7 共用：32K バイト (8K×32 ビット) + 256K バイト (16K×32 ビット×4) ARM7 専用：64K バイト (16K×32 ビット) ARM9,ARM7,DSP 命令 RAM 共用：256K バイト (8K×32 ビット×8) ARM9,ARM7,DSP データ RAM 共用：256K バイト (8K×32 ビット×8)
VRAM	合計 656K バイト (128KB + 128KB + 128KB + 128KB + 64KB + 16KB + 16KB + 32KB + 16KB) (各 VRAM は 0 ウェイトアクセスやバイトライトに対応可能)
システムクロック	33.514MHz

● **サウンド**

ADPCM/PCM 16ch (PSG 音源に最大 6ch、ノイズに最大 2ch を割り当て可能)
サウンドキャプチャ (リバーブ等に利用) 機能付き
DSP サウンドとのミキシングが可能

● **タイマ**

ARM9 側：16bit タイマ×4
ARM7 側：16bit タイマ×4

● **DMA**

ARM9 側：4ch×2
ARM7 側：4ch×2 + サウンド DMA 機能
NITRO 互換 DMA コントローラおよびブロック転送、調停方式が選択可能な新 DMA コントローラ。
システム設定で NITRO 互換 DMA コントローラの回路修正あり／なしを制御可能。

● **アクセラレータ**

除算器 (システム設定で除算器の回路修正あり／なしを制御可能)
平方根演算器

● **外部メモリ I/F**

カード検出、ホットスワップ機能を追加した TWL / NITRO カード I/F を 1 スロット、SD カード I/F を 1 スロット搭載。
システム設定で TWL / NITRO カード I/F の回路修正あり／なしを制御可能。
(AGB 互換のためのカートリッジ I/F は削除されました)

● **NAND フラッシュメモリ**

256MB の NAND フラッシュメモリを搭載。

● **カメラ**

VGA サイズで秒間 20 フレーム、QVGA サイズで秒間 30 フレームの取り込みに対応
YUV→RGB 変換回路 (RGB555 形式)、トリミング機能付き

1.1.2 メインメモリ

メインメモリは 16M バイト（デバッグ用 TWL は 32M バイトに拡張されています）の大きさを持っており、単独チップとして TWL プロセッサに接続されています。

また、システム設定により NITRO 互換モード時には 4M バイトに制限されます。

TWL / NITRO カードバスは CPU のアドレス空間にマッピングされていないため、アプリケーションやデータをメインメモリに読み出してから実行する必要があります。

起動時には、システム ROM によって TWL / NITRO カードから ARM9 用のアプリケーションがメインメモリへ転送されます。

ARM7 用のアプリケーションは起動時に ARM7 専用ワーク RAM へ転送されます。

1.1.3 LCD

上画面 LCD と下画面 LCD の 2 画面を備えています。

両 LCD 画面の仕様一覧を表 1-1 に示します。

項目	内容
表示解像度	256×192 ドット（比率 4：3）
表示色数	262,144 色（RGB=6:6:6）
画面サイズ	3.25 インチ
タイプ	透過型
バックライト	ON/OFF を切替え可能 IPL の本体設定で 5 段階の輝度調整が可能

表 1-1 LCD 画面の仕様一覧

画面サイズと輝度調整の段階以外は NintendoDS Lite の基本仕様と同じですが、電源 OFF 時に画面を白表示または黒表示に自動制御することができます。

1.1.4 デジタルキー

デジタルキーとして START、SELECT、十字、A、B、X、Y、L、R を備えています。

1.1.5 タッチパネル

下画面 LCD は、全面がドット単位で座標を取得することができる抵抗膜方式のタッチパネルとなっています。

また、TWL 本体には標準でタッチペンが装備されています。

1.1.6 マイク

無指向性のコンデンサマイクを備え、「ニンテンドーDS シリーズ専用 イヤホンマイク」も使用することができます。マイクからの音声入力をサンプリング処理することができます。

1.1.7 RTC

時刻を保持し、計時を行います。

アラーム機能を持ち、指定時刻に TWL をスリープモードから復帰させることができます。

1.1.8 ワイヤレス通信

2.4GHz 帯でワイヤレス通信可能なユニットを二基搭載しています。

NITRO 互換のユニット（NITRO 無線）は従来通り、無線 LAN（IEEE802.11b/g）のアクセスポイントとの接続を可能にするインフラストラクチャーモード、最大 16 台通信可能な DS ワイヤレスプレイモード、親機から NITRO カードのないう子機にゲームをダウンロードする DS ワイヤレスダウンロードプレイモードが利用可能です。

新たに搭載されたユニット（TWL 無線）はインフラストラクチャーモード（IEEE802.11b/g）のみが利用可能ですが、NITRO 無線よりも高速・高セキュリティな接続が可能となっています。

1.1.9 TWL / NITRO カード

TWL / NITRO 専用のセキュリティ機能付きゲームカードです。

ROM 以外にバックアップデバイスを搭載することができます。

TWL プロセッサとは外部メモリーインタフェースで接続されています。

詳細については「ニンテンドーDS/TWL ゲームカードマニュアル」を参照してください。

1.1.10 カートリッジ

NITRO でサポートされていた AGB 互換モードはサポートされません。

なお、NITRO 互換モードでのカートリッジへのアクセスは、カートリッジが挿入されていない状態と同じになります。

1.1.11 SD メモリカード

SD カードスロットを 1 スロット搭載し、SD メモリカードおよび 2～32GB の SDHC メモリカードの読み書きをすることができます。

ただし、SPI モード、暗号認証（CPRM）、High-Speed モードには対応していません。

1.1.12 NAND フラッシュメモリ

256MB の NAND フラッシュメモリを搭載しています。

NAND アプリケーションの保存、システムアプリケーションを含む NAND アプリケーションのセーブデータの保存、写真データの保存に使用されます。

1.1.13 カメラ

VGA（640×480）サイズならば秒間 20 フレーム、QVGA（320×240）サイズならば秒間 30 フレームのムービーを取り込むことができます。

1.1.14 DSP(Digital Signal Processor)

画像や音声処理をアクセラレートします。

ワーク RAM を使用することで、DSP での処理結果を ARM9、ARM7 と共有することができます。また、サウンド回路の出力と DSP で生成されたサウンドをミキシングして出力することができます。

1.2 メモリマップ

製品版 TWL における ARM9 の全体メモリマップを図 1-2 に、ARM7 の全体メモリマップを図 1-3 に示します。メモリマップの ■ 色の部分は NITRO から仕様変更のあった部分を、■ 色の部分は TWL で新規に追加された部分を示しています。

ARM9 のメモリ空間に関しては、プロテクションユニットの設定によりアクセスの属性が決まります。「プロテクションユニット」を参照してください。

● イメージについて

CPU が出力するアドレスは、デコーダによってメモリのアドレスに変換されます。

デコーダは常にアドレスの全ビットをデコードしているわけではないため、メモリが何も実装されていないアドレスへアクセスを行うと、「アドレスが小さい方向へ最も近いメモリ」のアドレスに変換されます※。

この本来何も存在しないはずの領域（実体とは異なる領域）に現れるメモリの分身をイメージと呼びます。

CPU では異なるアドレスでも、メモリにとっては同じアドレスなので、イメージへのアクセスは実体へのアクセスと同じです。

また、イメージ領域が実体のメモリサイズに対して大きい場合は、実体サイズのイメージが繰り返し出現します。

例えば、表中で 2D グラフィックスエンジン A の BG-VRAM の実体は 0x06000000～0x0607FFFF の 512KB ですが、イメージ領域は 0x06080000～0x061FFFFFFF の 1536KB となっており、イメージが 3 個存在していることになります。

※ARM9,7 共用内部ワーク RAM については、実体を 0x03000000～0x03FFFFFFF の範囲である程度自由に配置することができます。

また、イメージが出現せず、不定データになる領域もあります（図 1-2、図 1-3 参照）。

注意事項

この不定データにはアクセスしないでください。

0xFFFF0000	システム ROM (64KB)
0x0E000000	不定データ
0x0D000000	メインメモリ (拡張時 16MB)
0x0C000000	メインメモリのイメージ (16MB)
0x0A010000	不定データ
0x0A000000	カートリッジ RAM (0~64KB) ※
0x08000000	カートリッジ ROM (32MB) ※
0x07000800	OAM のイメージ
0x07000400	2D グラフィックスエンジン B の OAM (1KB)
0x07000000	2D グラフィックスエンジン A の OAM (1KB)
0x068A4000	LCDC 用 VRAM のイメージ
0x06800000	LCDC 用 VRAM (656KB)
0x06620000	OBJ-VRAM のイメージ
0x06600000	2D グラフィックスエンジン B の OBJ-VRAM (max.128KB)
0x06440000	OBJ-VRAM のイメージ
0x06400000	2D グラフィックスエンジン A の OBJ-VRAM (max.256KB)
0x06220000	BG-VRAM のイメージ
0x06200000	2D グラフィックスエンジン B の BG-VRAM (max.128KB)
0x06080000	BG-VRAM のイメージ
0x06000000	2D グラフィックスエンジン A の BG-VRAM (max.512KB)
0x05000800	パレット RAM のイメージ
0x05000600	2D グラフィックスエンジン B の OBJ 用パレット RAM (512B)
0x05000400	2D グラフィックスエンジン B の BG 用パレット RAM (512B)
0x05000200	2D グラフィックスエンジン A の OBJ 用パレット RAM (512B)
0x05000000	2D グラフィックスエンジン A の BG 用パレット RAM (512B)
0x04000000	I/O レジスタ※
0x03800000	ARM9,7 共用内部ワーク RAM のイメージ
0x03738000	ARM9,7 共用内部ワーク RAM (max.800KB)
0x03000000	ARM9,7 共用内部ワーク RAM のイメージ
0x02FE0000	データ TCM (16KB) : 移動可※
0x02000000	メインメモリ (16MB)
0x01FF8000	命令 TCM (32KB)
0x01000000	命令 TCM のイメージ
0x00000000	不定データ

図 1-2 ARM9 全体メモリマップ

0x0E000000	不定データ
0x0D000000	メインメモリ (拡張時 16MB)
0x0C000000	メインメモリのイメージ (16MB)
0x0A010000	不定データ
0x0A000000	カートリッジ RAM (0~64KB) ※
0x08000000	カートリッジ ROM (32MB) ※
0x07000000	不定データ
0x06040000	内部拡張ワーク RAM のイメージ
0x06000000	内部拡張ワーク RAM (max.256KB)
0x04810000	不定データ
0x04808000	ワイヤレス通信ウェイトステート 1
0x04800000	ワイヤレス通信ウェイトステート 0
0x04000000	I/O レジスタ※
0x03810000	ARM7 専用内部ワーク RAM のイメージ
0x03800000	ARM7 専用内部ワーク RAM (64KB)
0x03738000	ARM7,9 共用内部ワーク RAM (max.800KB)
0x03000000	ARM7,9 共用内部ワーク RAM のイメージ
0x02000000	メインメモリ (16MB)
0x00010000	不定データ
0x00000000	システム ROM (64KB)

図 1-3 ARM7 全体メモリマップ

※ I/O レジスタは ARM9 と ARM7 で異なります。

※ データ TCM のアドレスマッピングは可変なので、上図のデータ TCM のアドレスは設定例となります。

※ カートリッジ領域はアクセス可能ですが、ライト動作は無視され、リードは固定値が返ります。

1.3 サブプロセッサへ接続されているデバイスへのアクセス

TWL では、NITRO と同様にサブプロセッサに接続されているデバイスに対しては、必ず API を使用してアクセスする必要があります。

また、API を使用することで、サブプロセッサの状態に留意することなく機器にアクセスすることができます。

サブプロセッサに接続されているデバイスとしては、タッチパネル、マイク、RTC、ワイヤレス通信、デジタルキーの一部、サウンド、内蔵フラッシュメモリ、SD メモリカード、NAND フラッシュメモリ、AES 暗号化・復号化回路があります。

● API とは？

API とは Application Program Interface の略で、アプリケーション開発時にプログラムの作成効率を向上させるための関数（群）のことをいいます。

一般的には、低レベルのシステムコールやハードウェア制御のために用意されています。

注意事項

ARM9 中のサブプロセッサ（ARM7）のインタフェースに関するレジスタへのアクセスは可能ですが、API を使用する場合はこれらのレジスタにアクセスしないでください。

1.4 起動モード

アプリケーション動作時のモードは下記のモードのいずれかを選択します。

ただし、このモードはアプリケーション作成時に決定され、それ以降は変更できません。

1.4.1 TWL モード

TWL の持つすべての機能を使用することができます。

1.4.2 NITRO 互換モード

NITRO との互換性を持つモードです。

注意事項

AGB カートリッジのスロットが廃止されたため、カートリッジへのアクセスはカートリッジが挿入されていない状態と同じになります。また、サウンド出力の方式が変更されています。

1.5 仕向地

表 1-2 の通り、仕向地によって 4 種類の TWL 本体があります。

仕向地	設定可能言語	使用可能バナーフォント
日本（JP）	日	ひらがな、カタカナ、アルファベット、符号付アルファベット、数字等
北米（US）	英、仏、西	同上
欧州（EU）	英、仏、西、独、伊	同上
豪州（AU）	英	同上

表 1-2 仕向け地による TWL 本体の相違点

注意事項

NITRO では、日本・北米・欧州・豪州の各仕向地で共通の本体が使用されており、使用する言語をユーザーが任意に選択可能でしたが、TWL では仕向地によって 4 種類の TWL 本体が存在し、選択可能な言語も表 1-2 のように仕向地ごとに異なります。

2. NITRO との違い

TWL は NITRO との互換性を有していますが、新たにモジュールが追加されたり、カートリッジスロットが廃止されたり、ハードウェアの不具合を修正した回路を使用することができたりと、NITRO とはいくつかの違いがあります。この章では、TWL と NITRO との違いをまとめ、TWL で使用するモジュールを管理するシステム設定について説明します。

2.1 新たに追加されたモジュール

カメラ、DSP、新規共有 WRAM、新 DMA、新無線 LAN（TWL 無線）、SD メモリカード、NAND フラッシュメモリ、AES 暗号化・復号化回路が新たに追加されたモジュールです。このうち、新無線 LAN（TWL 無線）、SD メモリカード、NAND フラッシュメモリ、AES 暗号化・復号化回路についてはサブプロセッサに接続されているため、API を介してのみ操作することができます。

2.2 カートリッジスロット

カートリッジスロットは廃止されますが、レジスタやマッピングされるアドレスに変更はありません。スリープ状態からの復帰要因としても使用することができます。ただし、マッピングされたアドレスに対する書き込みは無効となり、読み込まれるデータは必ず 0 となります。また、カートリッジ IREQ/DREQ 割り込み許可フラグが 1 に設定されているとスリープモードに移行することができません。

2.3 変更のあったモジュール

NITRO では PMIC で行われていたアナログ・デジタル変換が CODEC と呼ばれるモジュールで行われるようになり、ミキサーの出力（サウンド回路自体には変更はありません）とマイク入力の仕様が変更となりました。ミキサーの詳細については、「ミキサー」を参照してください。マイクの詳細については、「マイク」を参照してください。

2.4 システム設定

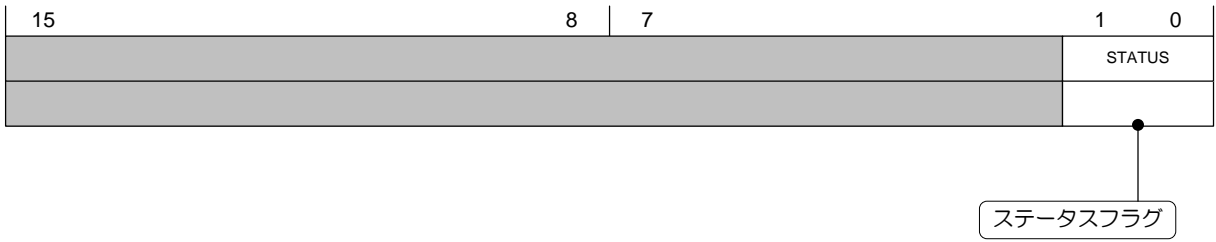
TWL モードではシステム設定レジスタ（SCFG_*）によって、新たに追加されたモジュールへのクロック供給やアクセスを制御することができます。

2.4.1 システム ROM

ARM9 が使用するシステム ROM のステータスを以下のレジスタで調べることができます。

システム設定 ROM ステータスレジスタ

名称 SCFG_A9ROM アドレス 0x04004000 属性 R 初期値 0x0000



● STATUS[d01~d00]：ステータスフラグ

00	NITRO 互換モード
01	TWL モード

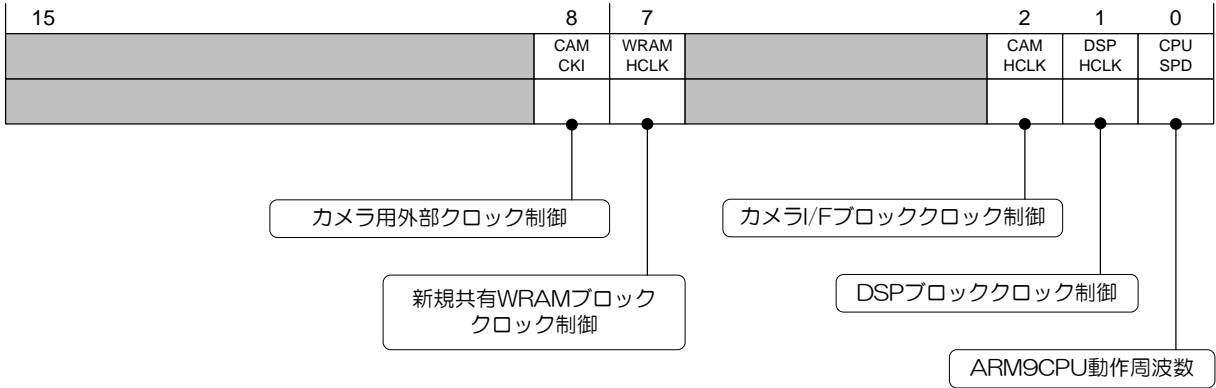
これらのビットはサブプロセッサで設定された値を読み込むためのもので、書き込みはできません。

2.4.2 新規ブロックのクロック制御

カメラ、新規共有 WRAM、DSP へのクロック供給や ARM9 の動作周波数を以下のレジスタで制御します。

システム設定 新規ブロッククロック制御レジスタ

名称 SCFG_CLK アドレス 0x04004004 属性 R/W 初期値 0x0084



● CAMCKI[d08] : カメラ用外部クロック制御

0	ディセーブル
1	イネーブル (16.76MHz を出力)

● CAMHCLK[d02] : カメラ I/F ブロッククロック制御

0	クロック停止
1	クロック供給

● WRAMHCLK[d07] : 新規共有 WRAM ブロッククロック制御

0	クロック停止
1	クロック供給

このビットはサブプロセッサでの設定を読み込むためのもので、書き込みはできません。

● DSPHCLK[d01] : DSP ブロッククロック制御

0	クロック停止
1	クロック供給

● CPUSPD[d00] : ARM9CPU 動作周波数

0	NITRO と同じ (67.03MHz)
1	倍速動作 (134.06MHz)

安全のため、クロックを停止させる場合は各ブロックが停止した状態で行ってください。

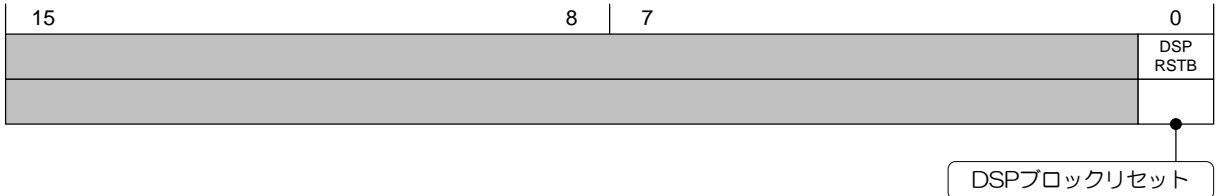
ARM9 の動作周波数を変更する場合は、ITCM 上のプログラムからビットを書き換え、その後最低 8 サイクルの間 ITCM 上でダミーループさせてください (ARM9 バスに対してアクセスを発生させないでください)。

2.4.3 新規ブロックのリセット

DSP ブロックにリセットをかける場合は、以下のレジスタを操作します。

システム設定 新規ブロックリセット制御レジスタ

名称 SCFG_RST アドレス 0x04004006 属性 R/W 初期値 0x0000



● DSPRSTB[d00] : DSP ブロックリセット

0	DSP ブロックにリセットをかける
1	DSP ブロックのリセットを解除する

● [d07～d00]：NITRO ブロック回路修正制御

修正される機能の詳細については、「修正される機能」を参照してください。

● MC[d07]：TWL / NITRO カード I/F 回路の修正

0	NITRO と同じ
1	修正された回路を使用

● REN[d02]：レンダラー回路の修正

0	NITRO と同じ
1	修正された回路を使用

● DIV[d04]：除算器回路の修正

0	NITRO と同じ
1	修正された回路を使用

● GEO[d01]：ジオメトリ回路の修正

0	NITRO と同じ
1	修正された回路を使用

● 2DE[d03]：2D エンジン回路の修正

0	NITRO と同じ
1	修正された回路を使用

● DMA[d00]：NITRO 互換 DMA 回路の修正

0	NITRO と同じ
1	修正された回路を使用

2.4.4.1 拡張される機能

ビットに対応するブロックが、以下のように機能が拡張されます。
安全のため、ビットを変更する場合は、各ブロックが動作していない状態で行ってください。

● メインメモリ

メインメモリの容量を 4MB（NITRO 互換）、16MB、32MB 境界に制限または拡張することができます。

0x0E000000			
0x0D000000	ALL 0	メインメモリのイメージ（16MB）	拡張されたメインメモリ（16MB）
0x0C000000	ALL 0	メインメモリのイメージ（16MB）	メインメモリのイメージ（16MB）
0x03000000			
0x02000000	メインメモリのイメージ（4MB）	メインメモリ（16MB）	メインメモリ（16MB）
	メインメモリのイメージ（4MB）		
	メインメモリのイメージ（4MB）		
	メインメモリ（4MB）		
4MB（NITRO 互換）		16MB	32MB

図 2-1 メインメモリの容量による違い

● VRAM アクセス

32 ビットバス幅に対応し、1 バイト書き込みが可能になります。

● LCDC 回路

表示ステータスレジスタ（DISPSTAT）に LCD 初期化信号が追加されます。

● 割り込み回路

新規ブロックに対応する割り込みフラグが追加されます。（新 DMA0～3、カメラ、DSP）
この拡張機能を OFF にした場合は、対応する拡張部分の割り込み許可フラグがハードウェアによってすべて 0 に初期化されます。
詳細については、「割り込み」を参照してください。

2.4.4.2 修正される機能

不具合を修正した回路を使用する場合は、該当するビットを 1 に設定して、使用する SDK を回路修正に対応したバージョンに変更する必要があります。また、SDK を変更することによってアプリケーションの細かい挙動が変わり、シビアなタイミングで組まれたルーチンの処理順序が変わってしまう恐れがありますのでご注意ください。
安全のため、ビットを変更する場合は、各ブロックが動作していない状態で行ってください。

- TWL / NITRO カード I/F
複数ページの書き込みおよび 32 ビット不定データの書き込みで発生していた不具合が修正されます。
- NITRO 互換 DMA
ARM9 システムバス上で DMA を複数チャネル並列に起動した際に優先順位の高い DMA が暴走する不具合と、DMA0 のソースアドレスに I/O レジスタまたはカートリッジバスが指定できない不具合が修正されます。
- ジオメトリ回路
 - ボックステスト
テストに使用するボックスの六面がすべてビューボリュームにかからない場合にビュー外と判定される不具合と、ポリゴンリダクションが発生した場合に間違った結果が返される不具合が修正されます。
 - TexImageParam コマンド
TexImageParam0, 1 で設定した属性が前のポリゴンに影響する不具合が修正されます。
 - コマンド FIFO
CPU によるジオメトリ FIFO への連続書き込みやコマンドバック時に、特定のデータ列によって意図しないジオメトリコマンドが挿入される不具合が修正されます。
 - 1 ドットポリゴン
連結ポリゴンで 1 ドットポリゴンが消去された場合に、強制的に頂点共有が解除される不具合が修正されます。
 - クリッピング
頂点がクリッピングされた場合に、青と緑の成分値が 0 になってしまう不具合（色化け）が修正されます。
- レンダラー回路
 - 4X4 圧縮テクスチャ
同じ色でも線形補間ありの方が若干暗く表示される不具合が修正されます。
 - シャドウポリゴン
マスク用と描画用のシャドウポリゴンの描画にずれが生じる不具合が修正されます。
- 2D グラフィックスエンジン
BG ウィンドウ機能で左上 Y 座標が 6 以下の場合にウィンドウが縦に伸びる不具合と、X 方向に表示全領域となるウィンドウを指定できない不具合が修正されます。
- 除算器の不具合を修正
ゼロ除算エラーが除算モードにかかわらず除数の 64 ビットすべてが 0 でなければ正しく機能しない不具合が修正されます。

2.4.5 新規共有 WRAM

ARM9 とサブプロセッサ、DSP 間で共有することのできるワーク RAM が追加されました。
詳細については、「ワーク RAM」を参照してください。

3. メモリ

TWL に搭載されているメモリの構成とメモリの仕様を表 3-1 に示します。

メモリ種別	バス幅	アクセス サイクル	DMA アクセス可能な ビット幅		メインプロセッサ アクセス可能な ビット幅		新 DMA アクセス可能な ビット幅	
			リード	ライト	リード	ライト	リード	ライト
OAM	32	1	16/32	16/32	8/16/32	16/32	32	32
VRAM	32※1/16	1	16/32	16/32	8/16/32	8※1/16/32	32	32
パレット RAM	16	1	16/32	16/32	8/16/32	16/32	32	32
I/O レジスタ	32	1	16/32	16/32	8/16/32	8/16/32	32	32
内部ワーク RAM	32	1	16/32	16/32	8/16/32	8/16/32	32	32
メインメモリ	16	1st R:5 / W:4 2nd 1	16/32	16/32	8/16/32	8/16/32	32	32
システム ROM	32	1	—	—	8/16/32	—	—	—
TCM/キャッシュ	32	$\frac{1}{2} / \frac{1}{4}$ ※2	—	—	8/16/32	8/16/32	—	—

表 3-1 メモリ構成とメモリ仕様

アクセスサイクル数はバス周波数 33.514MHz に対する値です。

また、この値はバス幅以下のビット幅でアクセスしたときの値になっています。

バス幅より大きなビット幅でアクセスすると、ビット幅をバス幅で割った数だけのアクセスサイクルがかかります。

※1 TWL モードでは、システム設定により VRAM のバス幅を 32 ビットに、1 バイト書き込みを可能にすることができます（SCFG_EXT レジスタで設定します）。

※2 TWL モードでは、システム設定によりメインプロセッサの動作周波数を NITRO の倍速（134.06 MHz）にすることができます（SCFG_CLK レジスタで設定します）。

● メモリ間の転送速度

表 3-1 において、バス幅とアクセスサイクルからメモリ間の DMA 転送速度を計算することができます。
内部ワーク RAM と VRAM 間の DMA 転送の例を表 3-2 と表 3-3 に示します。

転送メモリ	DMA の 転送ビット数	リードに かかるサイクル	ライトに かかるサイクル	合計サイクル	転送速度 (M バイト/sec)
内部ワーク RAM から VRAM	16	1	1	2	31.96
	32	1	2	3	42.62
VRAM から 内部ワーク RAM	16	1	1	2	31.96
	32	2	1	3	42.62

表 3-2 内部ワーク RAM と VRAM 間の DMA 転送速度 (VRAM のバス幅が 16 ビット時)

転送メモリ	DMA の 転送ビット数	リードに かかるサイクル	ライトに かかるサイクル	合計サイクル	転送速度 (M バイト/sec)
内部ワーク RAM から VRAM	16	1	1	2	31.96
	32	1	1	2	63.92
VRAM から 内部ワーク RAM	16	1	1	2	31.96
	32	1	1	2	63.92

表 3-3 内部ワーク RAM と VRAM 間の DMA 転送速度 (VRAM のバス幅が 32 ビット時)

● メインメモリの転送速度

メインメモリは、DMA を表 3-4 に示す設定で読み出す場合に先読みバッファが機能します。

設定項目	設定値
転送ビット数	32 ビット
ソースアドレス更新方法	インクリメント
デスティネーションアドレス	メインメモリ以外

表 3-4 先読みバッファが機能する DMA の設定

先読みバッファは 16 ビットの容量を持ち、デスティネーション側メモリへの書き込みと並列してメインメモリからの読み出しを行います。

表 3-5、表 3-6、表 3-7 にメインメモリと内部ワーク RAM 間およびメインメモリと VRAM 間の DMA 転送速度を示します。

また、表中の※印は先読みバッファによって合計サイクルが短縮されていることを示しています。

先読みバッファの詳細については「メインメモリ」を参照してください。

転送メモリ	DMA の 転送ビット数	リードに かかるサイクル	ライトに かかるサイクル	合計サイクル	転送速度 (M バイト/sec)
メインメモリから 内部ワーク RAM	16	1st 5 2nd~ 1	1	1st 6 2nd~ 2	2nd~ 31.96
	32	1st 6 2nd~ 2	1	1st 7 ※2nd~ 2	2nd~ 63.92
内部ワーク RAM から メインメモリ	16	1	1st 4 2nd~ 1	1st 5 2nd~ 2	2nd~ 31.96
	32	1	1st 5 2nd~ 2	1st 6 2nd~ 3	2nd~ 42.62

表 3-5 メインメモリと内部ワーク RAM 間の DMA 転送速度

転送メモリ	DMA の 転送ビット数	リードに かかるサイクル	ライトに かかるサイクル	合計サイクル	転送速度 (M バイト/sec)
メインメモリから VRAM	16	1st 5 2nd~ 1	1	1st 6 2nd~ 2	2nd~ 31.96
	32	1st 6 2nd~ 2	2	1st 8 ※2nd~ 3	2nd~ 42.62
VRAM から メインメモリ	16	1	1st 4 2nd~ 1	1st 5 2nd~ 2	2nd~ 31.96
	32	2	1st 5 2nd~ 2	1st 7 2nd~ 4	2nd~ 31.96

表 3-6 メインメモリと VRAM 間の DMA 転送速度 (VRAM のバス幅が 16 ビット時)

転送メモリ	DMA の 転送ビット数	リードに かかるサイクル	ライトに かかるサイクル	合計サイクル	転送速度 (M バイト/sec)
メインメモリから VRAM	16	1st 5 2nd~ 1	1	1st 6 2nd~ 2	2nd~ 31.96
	32	1st 6 2nd~ 2	1	1st 7 ※2nd~ 2	2nd~ 63.92
VRAM から メインメモリ	16	1	1st 4 2nd~ 1	1st 5 2nd~ 2	2nd~ 31.96
	32	1	1st 5 2nd~ 2	1st 6 2nd~ 3	2nd~ 42.62

表 3-7 メインメモリと VRAM 間の DMA 転送速度 (VRAM のバス幅が 32 ビット時)

CPU による転送速度は、命令実行時間やバスの取得にかかる時間などが関係するため計算よりも遅くなります。

3.1 外部メモリ

TWL / NITRO 専用メモリカードスロット用のハードウェアを「TWL / NITRO カード」と呼びます。

外部メモリコントロールレジスタ

名称 EXMEMCNT アドレス 0x04000204 属性 R/W 初期値 0x0000

15	14	11				8	7	0						
EP	IFM					MP								
メインメモリ				TWL / NITRO カード										

● [d15, d14]：メインメモリ：メインメモリに関する設定

- EP[d15]：優先権 CPU 選択
ARM9 と ARM7 が同時にメインメモリをアクセスした場合の優先順位です。

0	ARM9 優先
1	ARM7 優先

● IFM[d14]：インタフェースモード切り換えフラグ

0	Asynchronous モード（設定禁止）
1	Synchronous モード

必ず Synchronous モードで使用してください。

● [d11]：TWL / NITRO カードに関する設定

- MP[d11]：アクセス権 CPU 選択

0	ARM9
1	ARM7

3.1.1 メインメモリ

3.1.1.1 先読みバッファ

メインメモリとのインタフェースには先読みバッファが実装されており、ARM9 側の DMA と ARM7 側の DMA で共有されます。

「メインメモリをソースアドレスとし、メインメモリ以外をデスティネーションアドレスとした 32 ビット幅の DMA 転送」が行われると、デスティネーションメモリへ書き込んでいる間の期間を利用してメインメモリから次のデータを 16 ビット単位で先読みします。このため、32 ビット幅のリードを 1 サイクルで行うことが可能となります。その他のアクセス（CPU によるリードや 16 ビット幅の DMA 転送など）では先読みバッファは使用されません。

3.1.1.2 バーストモード

バーストモードではハーフワード（16 ビット幅）でシーケンシャルアクセスが 1 サイクルで行えます。ランダムアクセスとバーストモード時のシーケンシャルアクセスの基本アクセスサイクルを表 3-8 に示します。

	リード	ライト
ランダム	5 サイクル	4 サイクル
シーケンシャル	1 サイクル	1 サイクル

表 3-8 基本アクセスサイクル

3.1.1.2.1 バーストアクセスの条件

バーストモードでのシーケンシャルアクセスを「バーストアクセス」と呼びます。DMA 転送で、ソースアドレス/デスティネーションアドレスのいずれかがメインメモリに設定され、メインメモリ側のアドレス更新方法がインクリメントに設定されている場合、バーストアクセスとなります。メインメモリからメインメモリへの DMA 転送はすべて 1st アクセスになるため、一旦内部 RAM を介した方が速く転送できます。また、LDM や STM 命令による連続転送が実行されたときもバーストアクセスとなります。

3.1.1.2.2 ウェイトが追加される条件

236 ハーフワードをまたぐとターミネーションと呼ばれるウェイトが 3 サイクル入ります。また、16 ハーフワード単位でアドレスを 0 から定義すると、表 3-9 のように 13,14,15 のアドレスからアクセスを開始した場合にもウェイト（最大 3 ウェイト）が入ります。このため 4 ハーフワード境界 (=8 バイト境界) からデータを配置するようにすると効率が高くなります。

1 サイクル目	2	3	4	5	6	7	8	9	...
13	14	15	ウェイト	16	17	18	19	20	...
14	15	ウェイト	ウェイト	16	17	18	19	20	...
15	ウェイト	ウェイト	ウェイト	16	17	18	19	20	...

表 3-9 アクセス開始アドレスにより追加されるウェイト

3.1.1.2.3 メインメモリの DMA 転送サイクル

3.1.1.2.3.1 メインメモリからワーク RAM への 32 ビット DMA 転送サイクル

メインメモリは 16 ビット幅のバス、ワーク RAM は 32 ビット幅のバスのため、ハーフワード単位で読み込んだデータをワード単位で書き込むことになります。ワード単位で書き込んでいる最中、先読みバッファによって次のデータを読み込みます。

なお、ジオメトリコマンド FIFO への転送もワーク RAM への転送と同様になります。

●基本サイクル

基本サイクル時のメインメモリからワーク RAM への転送シーケンスを図 3-1 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
ワーク RAM							1W		2W		3W		4W
メインメモリ	wait	wait	wait	wait	1LR	1HR	2LR	2HR	3LR	3HR	4LR	4HR	5LR
...	237	238	239	240	241	242	243	244	245	246	247	248	249
...			117W					118W		119W		120W	
...	117LR	117HR	T	T	T	118LR	118HR	119LR	119HR	120LR	120HR	121LR	121HR

図 3-1 メインメモリからワーク RAM への転送シーケンス（基本サイクル）

●ワーストケース

アクセスを開始するメインメモリの番地によっては、ターミネーションと呼ばれるウェイトが入ります。

この場合、ターミネーション直後のアクセスは 1st アクセスとなります。

ワーストケース時のメインメモリからワーク RAM への転送シーケンスを図 3-2 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
ワーク RAM							1W						
メインメモリ	wait	wait	wait	wait	1LR	1HR	T	T	T	wait	wait	wait	wait
	14	15	16	17	18	19	20	21	22	23	24	25	26
			2W		3W		4W		5W		6W		7W
	2LR	2HR	3LR	3HR	4LR	4HR	5LR	5HR	6LR	6HR	7LR	7HR	8LR

wait	ウェイト
T	ターミネーション
nLR	n 番目の 32 ビットデータの低位 16 ビットをリード
nHR	n 番目の 32 ビットデータの上位 16 ビットをリード
nW	n 番目の 32 ビットデータをライト

図 3-2 メインメモリからワーク RAM への転送シーケンス（ワーストケース）

3.1.1.2.3.2 ワーク RAM からメインメモリへの 32 ビット DMA 転送サイクル

● 基本サイクル

基本サイクル時のワーク RAM からメインメモリへの転送シーケンスを図 3-3 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
ワーク RAM	1R						2R			3R			4R
メインメモリ		wait	wait	wait	1LW	1HW		2LW	2HW		3LW	3HW	

...	352	353	354	355	356	357	358	359	360	361			
...	117R									118R			
...	wait	T	T	T	wait	wait	wait	117LW	117HW				

図 3-3 ワーク RAM からメインメモリへの転送シーケンス（基本サイクル）

● ワーストケース

アクセスを開始するメインメモリの番地によっては、ターミネーションと呼ばれるウェイトが入ります。この場合、ターミネーション直後のアクセスは 1st アクセスとなります。

ワーストケース時のワーク RAM からメインメモリへの転送シーケンスを図 3-4 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
ワーク RAM	1R						2R						
メインメモリ		wait	wait	wait	1LW	1HW	wait	T	T	T	wait	wait	wait

	14	15	16	17	18	19	20	21					
			3R			4R							
	2LW	2HW		3LW	3HW		4LW	4HW					

wait	ウェイト
T	ターミネーション
nR	n 番目の 32 ビットデータをリード
nLW	n 番目の 32 ビットデータの下位 16 ビットをライト
nHW	n 番目の 32 ビットデータの上位 16 ビットをライト

図 3-4 ワーク RAM からメインメモリへの転送シーケンス（ワーストケース）

3.1.1.2.3.3 メインメモリから VRAM(16 ビット幅)への 32 ビット DMA 転送サイクル

システム設定で VRAM のバス幅を 16 ビットに設定している場合、メインメモリは 16 ビット幅のバス、VRAM は 16 ビット幅のバスのため、ハーフワード単位で読み込んだデータをハーフワード単位で書き込むことになります。ハーフワード単位で VRAM に書き込んでいる最中、先読みバッファによってメインメモリのデータを読み込みます。

●基本サイクル

基本サイクル時のメインメモリから VRAM への転送シーケンスを図 3-5 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
VRAM							1LW	1HW		2LW	2HW		3LW
メインメモリ	wait	wait	wait	wait	1LR	1HR	2LR		2HR	3LR		3HR	4LR
...	352	353	354	355	356	357	358	359	360	361	362		
...			117LW	117HW									
...	117LR	117HR	T	T	T	wait	wait	wait	wait	118LR	118HR		

図 3-5 メインメモリから VRAM(16 ビット幅)への転送シーケンス (基本サイクル)

●ワーストケース

アクセスを開始するメインメモリの番地によっては、ターミネーションと呼ばれるウェイトが入ります。この場合、ターミネーション直後のアクセスは 1st アクセスとなります。

ワーストケース時のメインメモリから VRAM への転送シーケンスを図 3-6 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
VRAM							1LW	1HW					
メインメモリ	wait	wait	wait	wait	1LR	1HR	T	T	T	wait	wait	wait	wait
	14	15	16	17	18	19	20	21	22	23	24	25	26
			2LW	2HW		3LW	3HW		4LW	4HW		5LW	5HW
	2LR	2HR	3LR		3HR	4LR		4HR	5LR		5HR	6LR	

wait	ウェイト
T	ターミネーション
nLR	n 番目の 32 ビットデータの下位 16 ビットをリード
nHR	n 番目の 32 ビットデータの上位 16 ビットをリード
nLW	n 番目の 32 ビットデータの下位 16 ビットをライト
nHW	n 番目の 32 ビットデータの上位 16 ビットをライト

図 3-6 メインメモリから VRAM(16 ビット幅)への転送シーケンス (ワーストケース)

3.1.1.2.3.4 VRAM(16 ビット幅)からメインメモリへの 32 ビット DMA 転送サイクル

● 基本サイクル

基本サイクル時の VRAM からメインメモリへの転送シーケンスを図 3-7 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
VRAM	1LR	1HR						2LR	2HR			3LR	3HR
メインメモリ			wait	wait	wait	1LW	1HW			2LW	2HW		

...	468	469	470	471	472	473	474	475	476	477	478	479	480
...	117LR	117HR			118LR	118HR							
...			117LW	117HW	wait	T	T	T	wait	wait	wait	118LW	118HW

図 3-7 VRAM(16 ビット幅)からメインメモリへの転送シーケンス（基本サイクル）

● ワーストケース

アクセスを開始するメインメモリの番地によっては、ターミネーションと呼ばれるウェイトが入ります。この場合、ターミネーション直後のアクセスは 1st アクセスとなります。

ワーストケース時の VRAM からメインメモリへの 32 ビット転送シーケンスを図 3-8 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	
VRAM	1LR	1HR						2LR	2HR			...
メインメモリ			wait	wait	wait	1LW	1HW	wait	T	T	T	...

	12	13	14	15	16	17	18	19	20
						3LR	3HR		
	wait	wait	wait	2LW	2HW			3LW	3HW

wait	ウェイト
T	ターミネーション
nLR	n 番目の 32 ビットデータの低位 16 ビットをリード
nHR	n 番目の 32 ビットデータの上位 16 ビットをリード
nLW	n 番目の 32 ビットデータの低位 16 ビットをライト
nHW	n 番目の 32 ビットデータの上位 16 ビットをライト

図 3-8 VRAM(16 ビット幅)からメインメモリへの 32 ビット転送シーケンス（ワーストケース）

3.1.1.2.3.5 メインメモリから VRAM(32 ビット幅)への 32 ビット DMA 転送サイクル

システム設定で VRAM のバス幅を 32 ビットに設定している場合、メインメモリは 16 ビット幅のバス、VRAM は 32 ビット幅のバスのため、ハーフワード単位で読み込んだデータをワード単位で書き込むことになります。ワード単位で書き込んでいる最中、先読みバッファによって次のデータを読み込みます。

なお、この転送サイクルはメインメモリからワーク RAM への転送と同様になります。

●基本サイクル

基本サイクル時のメインメモリから VRAM への転送シーケンスを図 3-9 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
VRAM							1W		2W		3W		4W
メインメモリ	wait	wait	wait	wait	1LR	1HR	2LR	2HR	3LR	3HR	4LR	4HR	5LR
...	237	238	239	240	241	242	243	244	245	246	247		
...			117W										
...	117LR	117HR	T	T	T	wait	wait	wait	wait	118LR	118HR		

図 3-9 メインメモリから VRAM(32 ビット幅)への転送シーケンス (基本サイクル)

●ワーストケース

アクセスを開始するメインメモリの番地によっては、ターミネーションと呼ばれるウェイトが入ります。

この場合、ターミネーション直後のアクセスは 1st アクセスとなります。

ワーストケース時のメインメモリから VRAM への転送シーケンスを図 3-10 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
VRAM							1W						
メインメモリ	wait	wait	wait	wait	1LR	1HR	T	T	T	wait	wait	wait	wait
	14	15	16	17	18	19	20	21	22	23	24	25	26
			2W		3W		4W		5W		6W		7W
	2LR	2HR	3LR	3HR	4LR	4HR	5LR	5HR	6LR	6HR	7LR	7HR	8LR

wait	ウェイト
T	ターミネーション
nLR	n 番目の 32 ビットデータの下位 16 ビットをリード
nHR	n 番目の 32 ビットデータの上位 16 ビットをリード
nW	n 番目の 32 ビットデータをライト

図 3-10 メインメモリから VRAM(32 ビット幅)への転送シーケンス (ワーストケース)

3.1.1.2.3.6 VRAM(32 ビット幅)からメインメモリへの 32 ビット DMA 転送サイクル

基本サイクル時の VRAM からメインメモリへの転送シーケンスを図 3-11 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	11	12	13
VRAM	1R						2R			3R			4R
メインメモリ		wait	wait	wait	1LW	1HW		2LW	2HW		3LW	3HW	
...	352	353	354	355	356	357	358	359	360	361	362	363	
...	117R			118R									
...		117LW	117HW	wait	T	T	T	wait	wait	wait	118LW	118HW	

図 3-11 VRAM(32 ビット幅)からメインメモリへの転送シーケンス (基本サイクル)

● ワーストケース

アクセスを開始するメインメモリの番地によっては、ターミネーションと呼ばれるウェイトが入ります。この場合、ターミネーション直後のアクセスは 1st アクセスとなります。

ワーストケース時の VRAM からメインメモリへの 32 ビット転送シーケンスを図 3-12 に示します。

サイクル	1	2	3	4	5	6	7	8	9	10	
VRAM	1R						2R				...
メインメモリ		wait	wait	wait	1LW	1HW	wait	T	T	T	...
	12	13	14	15	16	17	18	19			
						3R					
	wait	wait	wait	2LW	2HW		3LW	3HW			

wait	ウェイト
T	ターミネーション
nLR	n 番目の 32 ビットデータの低位 16 ビットをリード
nHR	n 番目の 32 ビットデータの上位 16 ビットをリード
nLW	n 番目の 32 ビットデータの低位 16 ビットをライト
nHW	n 番目の 32 ビットデータの上位 16 ビットをライト

図 3-12 VRAM(32 ビット幅)からメインメモリへの 32 ビット転送シーケンス (ワーストケース)

3.2 TWL プロセッサ内メモリ

3.2.1 VRAM

VRAM (A~I) の用途は固定されておらず、アプリケーションの用途に合わせてメモリを効率良く振り分けることができます。これを VRAM バンクコントロールと呼びます。

VRAM アクセス中はバンクを切り換えないようにしてください。

VRAM と用途の選択肢を表 3-10 に示します。

用途 \ VRAM		A	B	C	D	E	F	G	H	I
KB		128	128	128	128	64	16	16	32	16
LCDC		○	○	○	○	○	○	○	○	○
ARM7				○	○					
2D グラフィックス エンジン A	BG-VRAM	○	○	○	○	○	○	○		
	OBJ-VRAM	○	○			○	○	○		
	BG 拡張 パレットスロット					○	○	○		
	OBJ 拡張 パレットスロット						○	○		
2D グラフィックス エンジン B	BG-VRAM			○					○	○
	OBJ-VRAM				○					○
	BG 拡張 パレットスロット								○	
	OBJ 拡張 パレットスロット									○
3D グラフィックス (レンダリング エンジン)	テクスチャ イメージスロット	○	○	○	○					
	テクスチャ パレットスロット					○	○	○		

(BG はスクリーンデータおよびキャラクタデータ、OBJ はキャラクタデータ)

表 3-10 VRAM と用途の選択肢

LCDC, ARM7, BG-VRAM, OBJ-VRAM に割り当てた場合は ARM9 バスにもマッピングされるため、ARM9 で読み書きが可能になります。拡張パレットやテクスチャスロットに割り当てた場合は ARM9 バスにマッピングされません。

※1) LCDC

LCDC (LCD コントローラ) が扱える領域です。

VRAM A~D については、VRAM 表示モード時のビットマップデータ格納メモリとして使用することや、キャプチャ時のビットマップデータ書き込みメモリに指定することも可能になります。(詳しくは「VRAM 表示モード」や「表示キャプチャ」章を参照してください)

また、LCDC に割り当てた場合はそれぞれの VRAM が ARM9 バスに一意にマッピングされます。

(A~I すべての VRAM を LCDC に割り当てた場合、ARM9 バスの連続した領域にマッピングされます)

VRAM が拡張パレットスロットやテクスチャイメージ/パレットスロットに割り当てられている場合は CPU からアクセスできないため、データを書き換えるには一旦 LCDC に設定してください。

※2) BG-VRAM

BG のスクリーンデータ、キャラクタデータ、ビットマップデータを格納する領域です。

2D グラフィックスエンジン A には最大 512K バイト割り当てることができます。

2D グラフィックスエンジン B には最大 128K バイト割り当てることができます。

※3) OBJ-VRAM

OBJ のキャラクタデータ、ビットマップデータを格納する領域です。
2D グラフィックスエンジン A には最大 256K バイト割り当てることができます。
2D グラフィックスエンジン B には最大 128K バイト割り当てることができます。

※4) 拡張パレットスロット

2D グラフィックスエンジンが占有する空間です。
BG や OBJ を表示する際に、2D グラフィックスエンジンがカラーデータを参照します。
スロットは CPU のメモリ空間にはマッピングされません。

※5) テクスチャイメージスロット/テクスチャパレットスロット

3D グラフィックスエンジン内のレンダリングエンジンが占有する空間です。
テクスチャブレンディングの際に、レンダリングエンジンがテクセルカラーを参照します。
スロットは CPU のメモリ空間にはマッピングされません。

RAM バンクコントロールレジスタ 0

名称 VRAMCNT アドレス 0x04000240 属性 W 初期値 0x00000000

31	28	27	26	24	23	20	19	18	16	15	12	11	9	8	7	4	3	1	0
E			OFS	MST	E			OFS	MST	E			OFS	MST	E			OFS	MST
VRAM-D					VRAM-C					VRAM-B					VRAM-A				

● VRAM-D, VRAM-C

● E[d31][d23] : イネーブルフラグ

0	ディセーブル
1	イネーブル

● OFS[d28~d27][d20~d19] : 割り当てアドレス (MST : 「割り当て選択」によって下記アドレスに割り当てられます)

1) MST = 000 のとき

LCDC に割り当てられ、ARM9 メモリ空間にもマッピングされます。

VRAM-C	0x06840000~0x0685FFFF
VRAM-D	0x06860000~0x0687FFFF

2) MST = 001 のとき

2D グラフィックスエンジン A の BG に割り当てられ、ARM9 メモリ空間にもマッピングされます

00	0x06000000~0x0601FFFF
01	0x06020000~0x0603FFFF
10	0x06040000~0x0605FFFF
11	0x06060000~0x0607FFFF

3) MST = 010 のとき

ARM7 メモリ空間にマッピングされるため、ARM9 メモリ空間にはマッピングされません。

00	0x06000000~0x0601FFFF
01	0x06020000~0x0603FFFF
10	設定禁止
11	設定禁止

4) MST = 011 のとき

テクスチャイメージスロットに割り当てられ、ARM9 メモリ空間にはマッピングされません。

図 3-13 のテクスチャイメージスロットメモリマップを参照してください。

00	テクスチャイメージスロット 0
01	テクスチャイメージスロット 1
10	テクスチャイメージスロット 2 (クリアカラーイメージ)
11	テクスチャイメージスロット 3 (クリアデプスイメージ)

5) MST = 100 のとき

設定に関係なく、下記の ARM9 メモリ空間にマッピングされます。

VRAM-C	0x06200000~0x0621FFFF
VRAM-D	0x06600000~0x0661FFFF

● MST[d26~d24][d18~d16] : 割り当て選択

000	LCDC に割り当てる
001	2D グラフィックスエンジン A の BG に割り当てる
010	ARM7 に割り当てる
011	3D レンダリングエンジンのテクスチャイメージに割り当てる
100	VRAM-C の場合 : 2D グラフィックスエンジン B の BG に割り当てる VRAM-D の場合 : 2D グラフィックスエンジン B の OBJ に割り当てる
101~111	設定禁止

注意事項

VRAM-C、VRAM-D をサブプロセッサ (ARM7) に割り当てることはできますが、サブプロセッサの API を正常に動作させるためにはこのレジスタの設定を変更しないでください。

●VRAM-B, VRAM-A

●E[d15][d07] : イネーブルフラグ

0	ディセーブル
1	イネーブル

●OFS[d12~d11][d04~d03] : 割り当てアドレス（MST : 「割り当て選択」によって下記アドレスに割り当てられます）

1) MST = 00 のとき

LCDC に割り当てられ、ARM9 メモリ空間にもマッピングされます。

VRAM-A	0x06800000~0x0681FFFF
VRAM-B	0x06820000~0x0683FFFF

2) MST = 01 のとき

2D グラフィックスエンジン A の BG に割り当てられ、ARM9 メモリ空間にもマッピングされます

00	0x06000000~0x0601FFFF
01	0x06020000~0x0603FFFF
10	0x06040000~0x0605FFFF
11	0x06060000~0x0607FFFF

3) MST = 10 のとき

2D グラフィックスエンジン A の OBJ に割り当てられ、ARM9 メモリ空間にもマッピングされます

00	0x06400000~0x0641FFFF
01	0x06420000~0x0643FFFF
10	設定禁止
11	設定禁止

4) MST = 11 のとき

テクスチャイメージスロットに割り当てられ、ARM9 メモリ空間にはマッピングされません。

図 3-13 のテクスチャイメージスロットメモリマップを参照してください。

00	テクスチャイメージスロット 0
01	テクスチャイメージスロット 1
10	テクスチャイメージスロット 2（クリアカラーイメージ）
11	テクスチャイメージスロット 3（クリアデプスイメージ）

●MST[d09~d08][d01~d00] : 割り当て選択

00	LCDC に割り当てる
01	2D グラフィックスエンジン A の BG に割り当てる
10	2D グラフィックスエンジン A の OBJ に割り当てる
11	3D レンダリングエンジンのテクスチャイメージに割り当てる

複数の VRAM ブロックを同じ番地にマッピングしたり、同じスロットに割り当てた場合の動作は保証されません。
テクスチャイメージスロットは、レンダリングエンジン上において図 3-13 の通りです。

0x00080000	
0x00060000	スロット 3 (クリアデプスイメージ)
0x00040000	スロット 2 (クリアカラーイメージ)
0x00020000	スロット 1
0x00000000	スロット 0

図 3-13 テクスチャイメージスロット メモリマップ

テクスチャイメージスロット 2 および 3 は、レンダリングバッファを初期化するためのクリアイメージバッファとして使うこともできます（「レンダリングバッファ初期化」章のクリアイメージによる初期化参照）。

RAM バンクコントロールレジスタ 1

名称 WVRAMCNT アドレス 0x04000244 属性 W 初期値 0x00000000

名称: WVRAMONT サイクル: 0x04000244 属性: W 初期値: 0x00000000																							
31					24	23	20 19 18			16	15	12 11 10			8	7	2			0			
					BANK		E			OFS	MST		E			OFS	MST		E			MST	
WRAM						VRAM-G						VRAM-F						VRAM-E					

● VRAM-G, VRAM-F

● E[d23][d15] : イネーブルフラグ

0	ディセーブル
1	イネーブル

● OFS[d20~d19][d12~d11] : 割り当てアドレス (MST : 「割り当て選択」によって下記アドレスに割り当てられます)

1) MST = 000 のとき

LCDC に割り当てられ、ARM9 メモリ空間にもマッピングされます。

VRAM-F	0x06890000~0x06893FFF
VRAM-G	0x06894000~0x06897FFF

2) MST = 001 のとき

2D グラフィックスエンジン A の BG に割り当てられ、ARM9 メモリ空間にもマッピングされます

00	0x06000000~0x06003FFF
01	0x06004000~0x06007FFF
10	0x06010000~0x06013FFF
11	0x06014000~0x06017FFF

3) MST = 010 のとき

2D グラフィックスエンジン A の OBJ に割り当てられ、ARM9 メモリ空間にもマッピングされます

00	0x06400000~0x06403FFF
01	0x06404000~0x06407FFF
10	0x06410000~0x06413FFF
11	0x06414000~0x06417FFF

4) MST = 011 のとき

テクスチャパレットスロットに割り当てられ、ARM9 メモリ空間にはマッピングされません。

図 3-14 のテクスチャパレットスロットメモリマップを参照してください。

00	テクスチャパレットスロット 0
01	テクスチャパレットスロット 1
10	テクスチャパレットスロット 4
11	テクスチャパレットスロット 5

5) MST = 100 のとき

2D グラフィックスエンジン A の BG 拡張パレットスロットに割り当てられ、ARM9 メモリ空間にはマッピングされません。

図 3-15 の BG 拡張パレットスロットメモリマップを参照してください。

00	2D グラフィックスエンジン A の BG 拡張パレットスロット 0~1
01	2D グラフィックスエンジン A の BG 拡張パレットスロット 2~3
10	設定禁止
11	設定禁止

6) MST = 101 のとき

2D グラフィックスエンジン A の OBJ 拡張パレットスロットに割り当てられ、ARM9 メモリ空間にはマッピングされません。

下位 8K バイトをスロット 0 に割り当てますが、上位 8K バイトは無効になります。

図 3-16 の OBJ 拡張パレットスロットメモリマップを参照してください。

● MST[d18~d16][d10~d08] : 割り当て選択

000	LCDC に割り当てる
001	2D グラフィックスエンジン A の BG に割り当てる
010	2D グラフィックスエンジン A の OBJ に割り当てる
011	3D レンダリングエンジンのテクスチャパレットに割り当てる
100	2D グラフィックスエンジン A の BG 拡張パレットに割り当てる
101	2D グラフィックスエンジン A の OBJ 拡張パレットに割り当てる
110, 111	設定禁止

● VRAM-E

● E[d07] : イネーブルフラグ

0	ディセーブル
1	イネーブル

● MST[d02~d00] : 割り当て選択

000	LCDC に割り当てる
001	2D グラフィックスエンジン A の BG に割り当てる
010	2D グラフィックスエンジン A の OBJ に割り当てる
011	3D レンダリングエンジンのテクスチャパレットに割り当てる
100	2D グラフィックスエンジン A の BG 拡張パレットに割り当てる
101~111	設定禁止

注意：VRAM-E のマッピングは、MST の設定によって以下に固定となります（オフセット指定できません）。

000	ARM9 の 0x06880000~0x0688FFFF
001	ARM9 の 0x06000000~0x0600FFFF
010	ARM9 の 0x06400000~0x0640FFFF
011	テクスチャパレットスロット 0~3
100	BG 拡張パレットスロット 0~3（下位 32K バイトのみ）

VRAM E, F, G を割り当てる際のテクスチャパレットスロットは、レンダリングエンジン上において図 3-14 の通りです。
また、BG 拡張パレットは図 3-15 の通りです。

0x00018000	
0x00014000	スロット 5
0x00010000	スロット 4
0x0000C000	スロット 3
0x00008000	スロット 2
0x00004000	スロット 1
0x00000000	スロット 0

図 3-14 テクスチャパレットスロット メモリマップ

0x00008000	
0x00006000	スロット 3
0x00004000	スロット 2
0x00002000	スロット 1
0x00000000	スロット 0

図 3-15 BG 拡張パレットスロット メモリマップ

VRAM-F,G を OBJ 拡張パレットに割り当てた際のメモリマップは図 3-16 の通りです。

0x00004000	
0x00002000	スロット 1
0x00000000	スロット 0

図 3-16 OBJ 拡張パレットスロット メモリマップ

複数の VRAM ブロックを同じ番地にマッピングしたり、同じスロットに割り当てた場合の動作は保証されません。

RAM バンクコントロールレジスタ 2

名称 VRAM_HI_CNT アドレス 0x04000248 属性 W 初期値 0x0000

15					10	9	8	7					2	1	0
E							MST	E							MST
VRAM-I								VRAM-H							

● VRAM-I, VRAM-H

● E[d15][d07] : イネーブルフラグ

0	ディセーブル
1	イネーブル

● MST[d09~d08][d01~d00] : 割り当て選択

● VRAM-H の場合

00	LCDC に割り当てる メインプロセッサの 0x06898000~0x0689FFFF にマッピングされます
01	2D グラフィックスエンジン B の BG に割り当てる メインプロセッサの 0x06200000~0x06207FFF にマッピングされます
10	2D グラフィックスエンジン B の BG 拡張パレットに割り当てる スロット 0~3 にマッピングされます
11	設定禁止

● VRAM-I の場合

00	LCDC に割り当てる メインプロセッサの 0x068A0000~0x068A3FFFF にマッピングされます
01	2D グラフィックスエンジン B の BG に割り当てる メインプロセッサの 0x06208000~0x0620BFFF にマッピングされます
10	2D グラフィックスエンジン B の OBJ に割り当てる メインプロセッサの 0x06600000~0x06603FFF にマッピングされます
11	2D グラフィックスエンジン B の OBJ 拡張パレットに割り当てる スロット 0~1 にマッピングされます

MST ビット、OFS ビットによる VRAM の割り当てアドレスをまとめると、表 3-11 から表 3-16 の通りになります。

MST	OFS 割り当て	00	01	10	11
00	ARM9	VRAM-A : 0x06800000~0x0681FFFF VRAM-B : 0x06820000~0x0683FFFF			
01	2D グラフィックス エンジン A	BG-VRAM (0x06000000~ 0x0601FFFF)	BG-VRAM (0x06020000~ 0x0603FFFF)	BG-VRAM (0x06040000~ 0x0605FFFF)	BG-VRAM (0x06060000~ 0x0607FFFF)
10	2D グラフィックス エンジン A	OBJ-VRAM (0x06400000~ 0x0641FFFF)	OBJ-VRAM (0x06420000~ 0x0643FFFF)	設定禁止	
11	テクスチャイメージ	スロット 0	スロット 1	スロット 2	スロット 3
				(クリアイメージ)	

表 3-11 VRAM-A、VRAM-B の割り当て表

MST	OFS 割り当て	00	01	10	11
000	ARM9	VRAM-C : 0x06840000~0x0685FFFF VRAM-D : 0x06860000~0x0687FFFF			
001	2D グラフィックス エンジン A	BG-VRAM (0x06000000~ 0x0601FFFF)	BG-VRAM (0x06020000~ 0x0603FFFF)	BG-VRAM (0x06040000~ 0x0605FFFF)	BG-VRAM (0x06060000~ 0x0607FFFF)
010	ARM7	ARM7 の 0x06000000~ 0x0601FFFF	ARM7 の 0x06020000~ 0x0603FFFF	設定禁止	
011	テクスチャイメージ	スロット 0	スロット 1	スロット 2	スロット 3
				(クリアイメージ)	
100	2D グラフィックス エンジン B	VRAM-C : BG-VRAM (0x06200000~0x0621FFFF) VRAM-D : OBJ-VRAM (0x06600000~0x0661FFFF)			
101~ 111	設定禁止	設定禁止			

表 3-12 VRAM-C、VRAM-D の割り当て表

MST	割り当て	アドレス
000	ARM9	0x06880000~0x0688FFFF
001	2D グラフィックス エンジン A	BG-VRAM (0x06000000~0x0600FFFF)
010	2D グラフィックス エンジン A	OBJ-VRAM (0x06400000~0x0640FFFF)
011	テクスチャパレット	テクスチャパレットスロット 0~3
100	2D グラフィックス エンジン A	BG 拡張パレットスロット 0~3 (下位 32K バイトのみ有効)
101~ 111	設定禁止	設定禁止

表 3-13 VRAM-E の割り当て表

MST	OFS 割り当て	00	01	10	11
000	ARM9	VRAM-F : 0x06890000～0x06893FFF VRAM-G : 0x06894000～0x06897FFF			
001	2D グラフィックス エンジン A	BG-VRAM (0x06000000～ 0x06003FFF)	BG-VRAM (0x06004000～ 0x06007FFF)	BG-VRAM (0x06010000～ 0x06013FFF)	BG-VRAM (0x06014000～ 0x06017FFF)
010	2D グラフィックス エンジン A	OBJ-VRAM (0x06400000～ 0x06403FFF)	OBJ-VRAM (0x06404000～ 0x06407FFF)	OBJ-VRAM (0x06410000～ 0x06413FFF)	OBJ-VRAM (0x06414000～ 0x06417FFF)
011	テクスチャパレット	スロット 0	スロット 1	スロット 4	スロット 5
100	2D グラフィックス エンジン A	BG 拡張パレット スロット 0～1	BG 拡張パレット スロット 2～3	設定禁止	
101	2D グラフィックス エンジン A	OBJ 拡張パレットスロット 0（下位 8K バイトのみ有効）			
110	設定禁止	設定禁止			
111	設定禁止				

表 3-14 VRAM-F、VRAM-G の割り当て表

MST	割り当て	アドレス
00	ARM9	0x06898000~0x0689FFFF
01	2D グラフィックス エンジン B	BG-VRAM (0x06200000~0x06207FFF)
10	2D グラフィックス エンジン B	BG 拡張パレットスロット 0~3
11	設定禁止	設定禁止

表 3-15 VRAM-H の割り当て表

MST	割り当て	アドレス
00	ARM9	0x068A0000~0x068A3FFFF
01	2D グラフィックス エンジン B	BG-VRAM (0x06208000~0x0620BFFF)
10	2D グラフィックス エンジン B	OBJ-VRAM (0x06600000~0x06603FFF)
11	2D グラフィックス エンジン B	OBJ 拡張パレットスロット 0~1

表 3-16 VRAM-I の割り当て表

3.2.2 ワーク RAM

TWL には NITRO での共有 WRAM-0/1（32KB）に加えて、ARM9 と ARM7 で共有可能な WRAM-A0～A3（256KB）と ARM9、ARM7 と DSP で共有可能な WRAM-B0～B7（256KB）、WRAM-C0～C7（256KB）があります。WRAM-0/1 の設定については、NITRO と同様に RAM バンクコントロールレジスタ 1（WVRAMCNT）で設定します。WRAM-A,B,C の設定については、システム設定レジスタの WRAM バンクコントロールレジスタ 1～8（MBK1～MBK8）で設定します。また、有効領域が重なった場合は WRAM-A>WRAM-B>WRAM-C の優先順位でマッピングされ、無効領域には WRAM-0/1（ARM7 の 0x38000000 以降のアドレスでは ARM7 専用 WRAM）の実体およびイメージがマッピングされます。

3.2.2.1 WRAM-0/1

ワーク RAM の用途は固定されておらず、アプリケーションの用途に合わせてメモリを効率良く振り分けることができます。これをワーク RAM バンクコントロールと呼びます。ワーク RAM アクセス中はバンクを切り換えないようにしてください。

RAM バンクコントロールレジスタ 1

名称 WVRAMCNT アドレス 0x04000244 属性 W 初期値 0x00000000

3125242320191816151211108720																								
						BANK	E			OFS	MST	E			OFS	MST	E					MST		
WRAM							VRAM-G						VRAM-F						VRAM-E					

●WRAM

●BANK[d25～d24]：バンク指定

16K バイト×2 ブロックを ARM9 と ARM7 のどちらに割り当てるかを選択できます。

それぞれの値を設定した場合のメモリマップは図 3-17 を参照してください。

	ARM9	ARM7
00	32KB	なし
01	16KB (Block1)	16KB (Block0)
10	16KB (Block0)	16KB (Block1)
11	なし	32KB

注意事項

ワーク RAM の設定を変更することはできますが、サブプロセッサの API を正常に動作させるためにはこのレジスタの設定を変更しないでください。

ARM9 側メモリマップ

● 全域（32KB）割り当て

0x04000000	I/O・レジスタ
0x03800000	ARM9,7 共用内部ワーク RAM のイメージ
0x037F8000	ARM9,7 共用内部ワーク RAM (32KB)
0x03000000	ARM9,7 共用内部ワーク RAM のイメージ
0x02000000	メインメモリ

● ブロック 1（16KB）割り当て

0x04000000	I/O・レジスタ
0x03800000	ARM9,7 共用内部ワーク RAM (ブロック 1) のイメージ
0x037FC000	ARM9,7 共用内部ワーク RAM (ブロック 1 : 16KB)
0x03000000	ARM9,7 共用内部ワーク RAM (ブロック 1) のイメージ
0x02000000	メインメモリ

● ブロック 0（16KB）割り当て

0x04000000	I/O・レジスタ
0x03800000	ARM9,7 共用内部ワーク RAM (ブロック 0) のイメージ
0x037FC000	ARM9,7 共用内部ワーク RAM (ブロック 0 : 16KB)
0x03000000	ARM9,7 共用内部ワーク RAM (ブロック 0) のイメージ
0x02000000	メインメモリ

● 割り当てなし

0x04000000	I/O・レジスタ
0x03800000	不定データ
0x03000000	不定データ
0x02000000	メインメモリ

ARM7 側メモリマップ

● 割り当てなし

0x04000000	I/O・レジスタ
0x03810000	ARM7 専用内部ワーク RAM のイメージ
0x03800000	ARM7 専用内部ワーク RAM (64KB)
0x03000000	ARM7 専用内部ワーク RAM (32KB) のイメージ
0x02000000	メインメモリ

● ブロック 0（16KB）割り当て

0x04000000	I/O・レジスタ
0x03810000	ARM7 専用内部ワーク RAM のイメージ
0x03800000	ARM7 専用内部ワーク RAM (64KB)
0x037FC000	ARM9,7 共用内部ワーク RAM (ブロック 0 : 16KB)
0x03000000	ARM9,7 共用内部ワーク RAM (ブロック 0) のイメージ
0x02000000	メインメモリ

● ブロック 1（16KB）割り当て

0x04000000	I/O・レジスタ
0x03810000	ARM7 専用内部ワーク RAM のイメージ
0x03800000	ARM7 専用内部ワーク RAM (64KB)
0x037FC000	ARM9,7 共用内部ワーク RAM (ブロック 1 : 16KB)
0x03000000	ARM9,7 共用内部ワーク RAM (ブロック 1) のイメージ
0x02000000	メインメモリ

● 全域（32KB）割り当て

0x04000000	I/O・レジスタ
0x03810000	ARM7 専用内部ワーク RAM のイメージ
0x03800000	ARM7 専用内部ワーク RAM (64KB)
0x037F8000	ARM9,7 共用内部ワーク RAM (32KB)
0x03000000	ARM9,7 共用内部ワーク RAM のイメージ
0x02000000	メインメモリ

図 3-17 ARM9、ARM7 共用内部ワーク RAM 各設定時のメモリマップ

3.2.2.2 WRAM-A

WRAM-A は、サイズが 64KB の 4 つのメモリブロックを ARM9 または ARM7 の各スロット（オフセット）に割り当てることでメモリにマッピングされます。
どのメモリをどのスロットに割り当てるのかは、WRAM バンクコントロールレジスタ 1（MBK1）で設定します。

WRAM バンクコントロールレジスタ 1

名称 MBK1 アドレス 0x04004040 属性 R/W 初期値 0x00000000

31				27 26				24				23				19 18				16				15				11 10				8				7				3 2				0			
E					OF			M	E					OF			M	E					OF			M	E					OF			M												
WRAM-A3								WRAM-A2								WRAM-A1								WRAM-A0																							

● WRAM-A0～A3

● E[d31][d23][d15][d07]：イネーブル設定

WRAM-A の各メモリブロックの有効（1）／無効（0）を指定します。

● OF[d27～d26][d19～d18][d11～d10][d03～d02]：オフセット設定

WRAM-A の各メモリブロックの割り当て位置（オフセット 0～3）を指定します。

	設定内容
00	WRAM-Ax をオフセット 0 のメモリ領域（スロット 0）に割り当てる
01	WRAM-Ax をオフセット 1 のメモリ領域（スロット 1）に割り当てる
10	WRAM-Ax をオフセット 2 のメモリ領域（スロット 2）に割り当てる
11	WRAM-Ax をオフセット 3 のメモリ領域（スロット 3）に割り当てる

● M[d24][d16][d08][d00]：マスター設定

WRAM-A の各メモリブロックのマスターを指定します。

	設定内容
0	WRAM-Ax のマスターを ARM9 に設定する
1	WRAM-Ax のマスターを ARM7 に設定する

マスターとオフセットの組み合わせは各 WRAM-A メモリですべて異なるように設定してください。

例)

WRAM-A0	ARM9 のスロット 0（M=0, OF=00）
WRAM-A1	ARM9 のスロット 1（M=0, OF=01）
WRAM-A2	ARM7 のスロット 0（M=1, OF=00）
WRAM-A3	ARM7 のスロット 1（M=1, OF=01）

次に、ARM9 および ARM7 の WRAM コントロールレジスタ 6（MBK6）で領域イメージサイズと開始、終了アドレスを設定します。

WRAM バンクコントロールレジスタ 6

名称 MBK6 アドレス 0x04004054 属性 R/W 初期値 0x00000000

31	28	20	13	12	11	4	0
EADDR				IS		SADDR	
WRAM-A							

●WRAM-A

●EADDR[d28～d20]：エンドアドレス設定

WRAM-A 領域のエンドアドレスを指定します。アドレスの指定は 64KB きざみで行います。

エンドアドレス = 0x02FFFFFF + (EADDR << 16) (0x02FFFFFF～0x03FFFFFF)

●IS[d13～d12]：イメージサイズ設定

WRAM-A 領域のイメージサイズを指定します。

	設定内容
00	64KB 単位で WRAM-A スロット 0 のイメージを繰り返す
01	
10	128KB 単位で WRAM-A スロット 0～1 のイメージを繰り返す
11	256KB 単位で WRAM-A スロット 0～3 のイメージを繰り返す

●イメージサイズを 64KB 単位（IS=00,01）に設定

0x04000000

END	無効領域	0x03FF0000	スロット 0
START	有効領域	0x03030000	...
		0x03020000	スロット 0
		0x03010000	スロット 0
	無効領域	0x03000000	スロット 0

●イメージサイズを 128KB 単位（IS=10）に設定

0x04000000

END	無効領域	0x03FF0000	スロット 1
START	有効領域	0x03030000	...
		0x03020000	スロット 0
		0x03010000	スロット 1
	無効領域	0x03000000	スロット 0

●イメージサイズを 256KB 単位（IS=11）に設定

0x04000000

END	無効領域	0x03FF0000	スロット 3
START	有効領域	0x03030000	...
		0x03020000	スロット 2
		0x03010000	スロット 1
	無効領域	0x03000000	スロット 0

●SADDR[d11～d04]：スタートアドレス設定

WRAM-A 領域のスタートアドレスを指定します。アドレスの指定は 64KB きざみで行います。

スタートアドレス = 0x03000000 + (SADDR << 16) (0x03000000～0x03FF0000)

有効領域の重複は、WRAM-A>WRAM-B>WRAM-C の優先順位でマッピングされ、無効領域には他の共有 RAM がマッピングされていなければ WRAM-0/1 がマッピングされます。

また、メモリブロックが割り当てられていないスロットへのアクセスは、ライトが無効でリードは必ず 0 を返します。

例)

ARM9 のイメージサイズを 256KB 単位（IS=11）に設定し、各スロットに以下のようにメモリブロックを割り当てた場合は、ARM9 のスロット 1 の領域がライト無効・リード ALL0 となります。

WRAM-A0 ARM9 のスロット 0 (M=0, OF=00)

WRAM-A1 ARM7 のスロット 0 (M=1, OF=00)

WRAM-A2 ARM9 のスロット 2 (M=0, OF=10)

WRAM-A3 ARM9 のスロット 3 (M=0, OF=11)

3.2.2.3 WRAM-B

WRAM-B は、サイズが 32KB の 8 つのメモリブロックを ARM9、ARM7 または DSP のプログラムバスの各スロット（オフセット）に割り当てることでメモリにマッピングされます。

どのメモリブロックをどのスロットに割り当てるのかは、WRAM バンクコントロールレジスタ 2～3（MBK2, MBK3）で設定します。

WRAM バンクコントロールレジスタ 2

名称 MBK2 アドレス 0x04004044 属性 R/W 初期値 0x00000000

31	28	26	25	24	23	20	18	17	16	15	12	10	9	8	7	4	2	1	0
E			OF	M	E			OF	M	E			OF	M	E			OF	M
WRAM-B3					WRAM-B2					WRAM-B1					WRAM-B0				

● WRAM-B0～B3

● E[d31][d23][d15][d07]：イネーブル設定

WRAM-B の各メモリブロックの有効（1）／無効（0）を指定します。

● OF[d28～d26][d20～d18][d12～d10][d04～d02]：オフセット設定

WRAM-B の各メモリブロックの割り当て位置（オフセット 0～7）を指定します。

	設定内容
000	WRAM-Bx をオフセット 0 のメモリ領域（スロット 0）に割り当てる
001	WRAM-Bx をオフセット 1 のメモリ領域（スロット 1）に割り当てる
010	WRAM-Bx をオフセット 2 のメモリ領域（スロット 2）に割り当てる
011	WRAM-Bx をオフセット 3 のメモリ領域（スロット 3）に割り当てる
100	WRAM-Bx をオフセット 4 のメモリ領域（スロット 4）に割り当てる
101	WRAM-Bx をオフセット 5 のメモリ領域（スロット 5）に割り当てる
110	WRAM-Bx をオフセット 6 のメモリ領域（スロット 6）に割り当てる
111	WRAM-Bx をオフセット 7 のメモリ領域（スロット 7）に割り当てる

● M[d25～d24][d17～d16][d09～d08][d01～d00]：マスター設定

WRAM-B の各メモリブロックのマスターを指定します。

	設定内容
00	WRAM-Bx のマスターを ARM9 に設定する
01	WRAM-Bx のマスターを ARM7 に設定する
10	WRAM-Bx のマスターを DSP（プログラムバス）に設定する
11	

WRAM バンクコントロールレジスタ 3

名称 MBK3 アドレス 0x04004048 属性 R/W 初期値 0x00000000

31	28	26	25	24	23	20	18	17	16	15	12	10	9	8	7	4	2	1	0
E			OF	M	E			OF	M	E			OF	M	E			OF	M
WRAM-B7					WRAM-B6					WRAM-B5					WRAM-B4				

● WRAM-B4～B7

● E[d31][d23][d15][d07] : イネーブル設定

WRAM-B の各メモリブロックの有効 (1) / 無効 (0) を指定します。

● OF[d28～d26][d20～d18][d12～d10][d04～d02] : オフセット設定

WRAM-B の各メモリブロックの割り当て位置 (オフセット 0～7) を指定します。

	設定内容
000	WRAM-Bx をオフセット 0 のメモリ領域 (スロット 0) に割り当てる
001	WRAM-Bx をオフセット 1 のメモリ領域 (スロット 1) に割り当てる
010	WRAM-Bx をオフセット 2 のメモリ領域 (スロット 2) に割り当てる
011	WRAM-Bx をオフセット 3 のメモリ領域 (スロット 3) に割り当てる
100	WRAM-Bx をオフセット 4 のメモリ領域 (スロット 4) に割り当てる
101	WRAM-Bx をオフセット 5 のメモリ領域 (スロット 5) に割り当てる
110	WRAM-Bx をオフセット 6 のメモリ領域 (スロット 6) に割り当てる
111	WRAM-Bx をオフセット 7 のメモリ領域 (スロット 7) に割り当てる

● M[d25～d24][d17～d16][d09～d08][d01～d00] : マスター設定

WRAM-B の各メモリブロックのマスターを指定します。

	設定内容
00	WRAM-Bx のマスターを ARM9 に設定する
01	WRAM-Bx のマスターを ARM7 に設定する
10	WRAM-Bx のマスターを DSP (プログラムバス) に設定する
11	

マスターとオフセットの組み合わせは各 WRAM-B メモリですべて異なるように設定してください。

例)

WRAM-B0	ARM9 のスロット 0 (M=00, OF=000)
WRAM-B1	ARM9 のスロット 1 (M=00, OF=001)
WRAM-B2	ARM9 のスロット 2 (M=00, OF=010)
WRAM-B3	ARM9 のスロット 3 (M=00, OF=011)
WRAM-B4	ARM7 のスロット 0 (M=01, OF=000)
WRAM-B5	ARM7 のスロット 1 (M=01, OF=001)
WRAM-B6	DSP (プログラムバス) のスロット 0 (M=10, OF=000)
WRAM-B7	DSP (プログラムバス) のスロット 1 (M=10, OF=001)

WRAM バンクコントロールレジスタ 7

Diagram illustrating the structure of the 32-bit WRAM-B address:

- 31** to **19**: EADDR (19 bits)
- 18** to **16**: IS (3 bits)
- 15** to **3**: SADDR (11 bits)
- 2** to **0**: (3 bits, part of SADDR)

●EADDR[d28～d19]：エンドアドレス設定

●IS[d13～d12]：イメージサイズ設定

WRAM-B 領域のイメージサイズを指定します。

	設定内容
00	32KB 単位で WRAM-B スロット 0 のイメージを繰り返す
01	64KB 単位で WRAM-B スロット 0~1 のイメージを繰り返す
10	128KB 単位で WRAM-B スロット 0~3 のイメージを繰り返す
11	256KB 単位で WRAM-B スロット 0~7 のイメージを繰り返す

- イメージサイズを 64KB 単位 (IS=01) に設定
0x04000000

END	無効領域	0x03FF8000	スロット 0
		0x03FF0000	スロット 0
START	有効領域	0x03030000	...
		0x03028000	スロット 0
		0x03020000	スロット 0
		0x03018000	スロット 0
		0x03010000	スロット 0
	無効領域	0x03008000	スロット 0
		0x03000000	スロット 0

END	無効領域	0x03FF8000	スロット 1
		0x03FF0000	スロット 0
START	有効領域	0x03030000	...
		0x03028000	スロット 1
		0x03020000	スロット 0
		0x03018000	スロット 1
		0x03010000	スロット 0
	無効領域	0x03008000	スロット 1
		0x03000000	スロット 0

- イメージサイズを 256KB 単位 (IS=11) に設定
0x04000000

END	無効領域	0x03FF8000	スロット 3
		0x03FF0000	スロット 2
START	有効領域	0x03030000	…
		0x03028000	スロット 1
		0x03020000	スロット 0
		0x03018000	スロット 3
		0x03010000	スロット 2
		0x03008000	スロット 1
	無効領域	0x03000000	スロット 0

END	無効領域	0x03FF8000	スロット 7
		0x03FF0000	スロット 6
START	有効領域	0x03030000	...
		0x03028000	スロット 5
		0x03020000	スロット 4
		0x03018000	スロット 3
		0x03010000	スロット 2
		0x03008000	スロット 1
	無効領域	0x03000000	スロット 0

- SADDR[d11～d03]：スタートアドレス設定

WRAM-B 領域のスタートアドレスを指定します。アドレスの指定は 32KB きざみで行います。
 スタートアドレス = $0x03000000 + (SADDR \ll 15)$ (0x03000000~0x03FF8000)

また、メモリブロックが割り当てられていないスロットへのアクセスは、ライトが無効でリードは必ず 0 を返します。

3.2.2.4 WRAM-C

WRAM-C は、サイズが 32KB の 8 つのメモリブロックを ARM9、ARM7 または DSP のデータバスの各スロット（オフセット）に割り当てることでメモリにマッピングされます。

どのメモリをどのスロットに割り当てるのかは、WRAM バンクコントロールレジスタ 4～5（MBK4、MBK5）で設定します。

WRAM バンクコントロールレジスタ 4

名称 MBK4 アドレス 0x0400404C 属性 R/W 初期値 0x00000000

31	28	26	25	24	23	20	18	17	16	15	12	10	9	8	7	4	2	1	0
E			OF	M	E			OF	M	E			OF	M	E			OF	M
WRAM-C3					WRAM-C2					WRAM-C1					WRAM-C0				

● WRAM-C0～C3

● E[d31][d23][d15][d07]：イネーブル設定

WRAM-C の各メモリブロックの有効（1）／無効（0）を指定します。

● OF[d28～d26][d20～d18][d12～d10][d04～d02]：オフセット設定

WRAM-C の各メモリブロックの割り当て位置（オフセット 0～7）を指定します。

	設定内容
000	WRAM-Cx をオフセット 0 のメモリ領域（スロット 0）に割り当てる
001	WRAM-Cx をオフセット 1 のメモリ領域（スロット 1）に割り当てる
010	WRAM-Cx をオフセット 2 のメモリ領域（スロット 2）に割り当てる
011	WRAM-Cx をオフセット 3 のメモリ領域（スロット 3）に割り当てる
100	WRAM-Cx をオフセット 4 のメモリ領域（スロット 4）に割り当てる
101	WRAM-Cx をオフセット 5 のメモリ領域（スロット 5）に割り当てる
110	WRAM-Cx をオフセット 6 のメモリ領域（スロット 6）に割り当てる
111	WRAM-Cx をオフセット 7 のメモリ領域（スロット 7）に割り当てる

● M[d25～d24][d17～d16][d09～d08][d01～d00]：マスター設定

WRAM-C の各メモリブロックのマスターを指定します。

	設定内容
00	WRAM-Cx のマスターを ARM9 に設定する
01	WRAM-Cx のマスターを ARM7 に設定する
10	WRAM-Cx のマスターを DSP（データバス）に設定する
11	

WRAM バンクコントロールレジスタ 5

名称 MBK5 アドレス 0x04004050 属性 R/W 初期値 0x00000000

31	28	26	25	24	23	20	18	17	16	15	12	10	9	8	7	4	2	1	0
E			OF	M	E			OF	M	E			OF	M	E			OF	M
WRAM-C7					WRAM-C6					WRAM-C5					WRAM-C4				

● WRAM-C4～C7

● E[d31][d23][d15][d07] : イネーブル設定

WRAM-C の各メモリブロックの有効 (1) / 無効 (0) を指定します。

● OF[d28～d26][d20～d18][d12～d10][d04～d02] : オフセット設定

WRAM-C の各メモリブロックの割り当て位置 (オフセット 0～7) を指定します。

	設定内容
000	WRAM-Cx をオフセット 0 のメモリ領域 (スロット 0) に割り当てる
001	WRAM-Cx をオフセット 1 のメモリ領域 (スロット 1) に割り当てる
010	WRAM-Cx をオフセット 2 のメモリ領域 (スロット 2) に割り当てる
011	WRAM-Cx をオフセット 3 のメモリ領域 (スロット 3) に割り当てる
100	WRAM-Cx をオフセット 4 のメモリ領域 (スロット 4) に割り当てる
101	WRAM-Cx をオフセット 5 のメモリ領域 (スロット 5) に割り当てる
110	WRAM-Cx をオフセット 6 のメモリ領域 (スロット 6) に割り当てる
111	WRAM-Cx をオフセット 7 のメモリ領域 (スロット 7) に割り当てる

● M[d25～d24][d17～d16][d09～d08][d01～d00] : マスター設定

WRAM-C の各メモリブロックのマスターを指定します。

	設定内容
00	WRAM-Cx のマスターを ARM9 に設定する
01	WRAM-Cx のマスターを ARM7 に設定する
10	WRAM-Cx のマスターを DSP (データバス) に設定する
11	

マスターとオフセットの組み合わせは各 WRAM-C メモリですべて異なるように設定してください。

例)

WRAM-C0	ARM9 のスロット 0 (M=00, OF=000)
WRAM-C1	ARM9 のスロット 1 (M=00, OF=001)
WRAM-C2	ARM7 のスロット 0 (M=01, OF=000)
WRAM-C3	ARM7 のスロット 1 (M=01, OF=001)
WRAM-C4	DSP (データバス) のスロット 0 (M=10, OF=000)
WRAM-C5	DSP (データバス) のスロット 1 (M=10, OF=001)
WRAM-C6	DSP (データバス) のスロット 2 (M=10, OF=010)
WRAM-C7	DSP (データバス) のスロット 3 (M=10, OF=011)

WRAM バンクコントロールレジスタ 8

31	28	19	13	12	11	3	0
		EADDR			IS	SADDR	
WRAM-C							

●EADDR[d28～d19]：エンドアドレス設定

●IS[d13～d12]：イメージサイズ設定

WRAM-C 領域のイメージサイズを指定します。

	設定内容
00	32KB 単位で WRAM-C スロット 0 のイメージを繰り返す
01	64KB 単位で WRAM-C スロット 0~1 のイメージを繰り返す
10	128KB 単位で WRAM-C スロット 0~3 のイメージを繰り返す
11	256KB 単位で WRAM-C スロット 0~7 のイメージを繰り返す

0x04000000

END	無効領域	0x03FF8000	スロット 0
		0x03FF0000	スロット 0
START	有効領域	0x03030000	...
		0x03028000	スロット 0
		0x03020000	スロット 0
		0x03018000	スロット 0
		0x03010000	スロット 0
	無効領域	0x03008000	スロット 0
		0x03000000	スロット 0

0x04000000

END	無効領域	0x03FF8000	スロット 1
		0x03FF0000	スロット 0
START	有効領域	0x03030000	...
		0x03028000	スロット 1
		0x03020000	スロット 0
		0x03018000	スロット 1
		0x03010000	スロット 0
	無効領域	0x03008000	スロット 1
		0x03000000	スロット 0

0x04000000

END	無効領域	0x03FF8000	スロット 3
		0x03FF0000	スロット 2
START	有効領域	0x03030000	…
		0x03028000	スロット 1
		0x03020000	スロット 0
		0x03018000	スロット 3
		0x03010000	スロット 2
		0x03008000	スロット 1
	無効領域	0x03000000	スロット 0

0x04000000

END	無効領域	0x03FF8000	スロット 7
		0x03FF0000	スロット 6
START	有効領域	0x03030000	...
		0x03028000	スロット 5
		0x03020000	スロット 4
		0x03018000	スロット 3
		0x03010000	スロット 2
	無効領域	0x03008000	スロット 1
		0x03000000	スロット 0

WRAM-C 領域のスタートアドレスを指定します。アドレスの指定は 32KB きざみで行います。

スタートアドレス = 0x03000000 + (SADDR << 15) (0x03000000~0x03FF8000)

また、メモリブロックが割り当てられていないスロットへのアクセスは、ライトが無効でリードは必ず 0 を返します。

3.2.3 TWL-SDK におけるワーク RAM の設定

TWL-SDK では、メモリインタフェースライブラリの WRAM マネージャが WRAM-A/B/C に対する各プロセッサの割り当て要求を調停しています。

WRAM-A (64KB×4) はすべて ARM7 が使用するため割り当てを変更することができませんが、WRAM-B (32KB×8) と WRAM-C (32KB×8) に関しては、ARM9、ARM7、DSP 間で柔軟に割り当てを変更することができます。WRAM マネージャについての詳細は、TWL-SDK 関数リファレンスマニュアルのメモリインタフェース概要「ワーク RAM」を参照してください。

3.2.4 I/O レジスタ

各 I/O レジスタ個別のマッピングに関しては付録を参照してください。

- 未定義レジスタへのアクセスについて
I/O レジスタ領域の中で未定義アドレスへアクセスした場合、表 3-17 の通りの挙動となります。

アクセス先	ライト	リード
未定義レジスタ	無効	ALL ゼロ

表 3-17 未定義レジスタへのアクセス結果

3.3 カードブート時のメモリマップ

カードブート時のメモリマップを図 3-18 に示します。

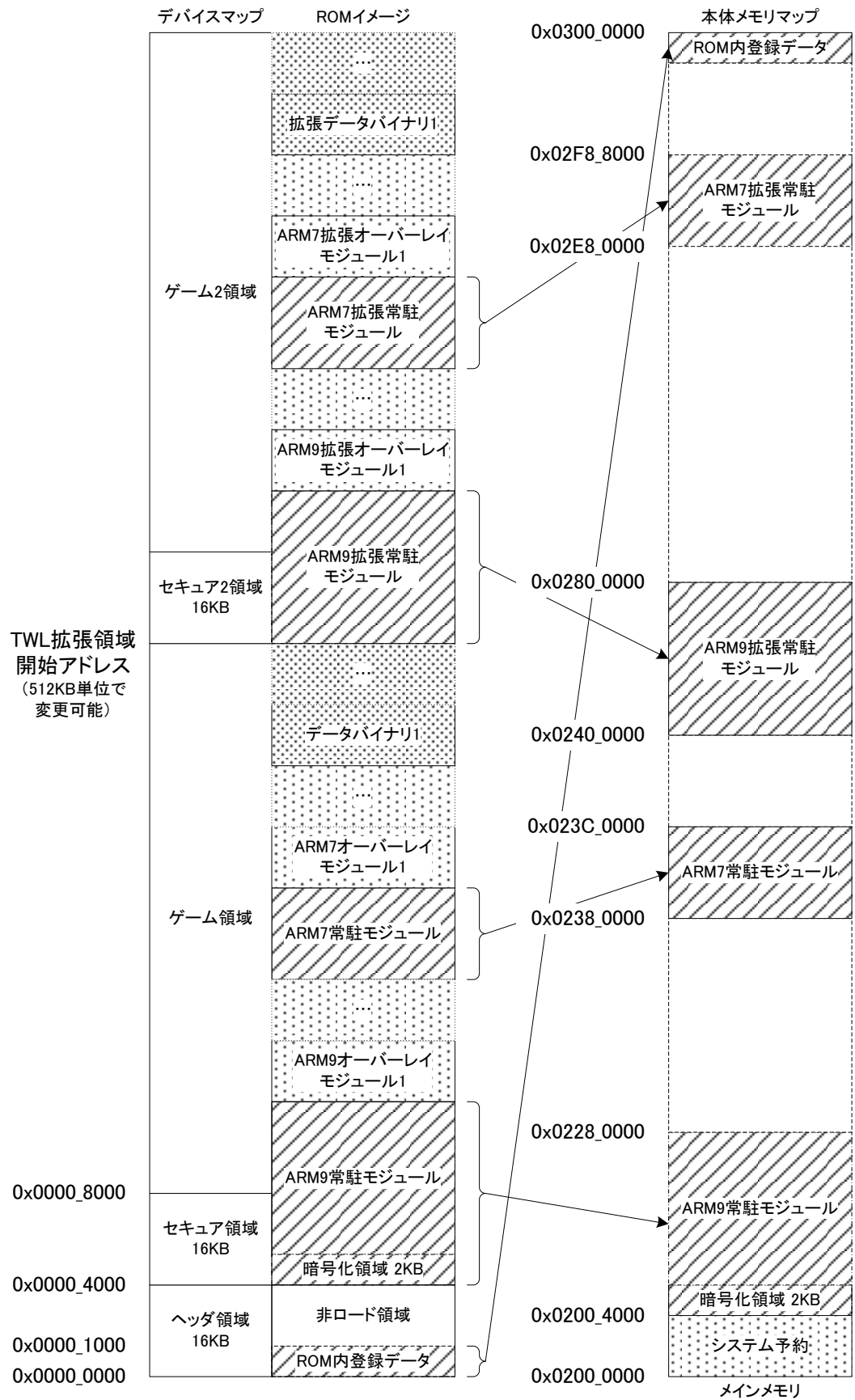


図 3-18 カードブート時のメモリマップ

各領域の詳細は以下の通りです。名称に「拡張」が付いている領域とセキュア2領域、ゲーム2領域は、TWLで拡張された領域です。

NITRO 互換モードでは、メインメモリは先頭から 4MB のみが有効となります。また、先頭のシステム予約領域は存在せず、ARM9 常駐モジュールをメインメモリの先頭から配置することができます。

- 常駐モジュール、拡張常駐モジュール
アプリケーション起動時にロードされるモジュールです。
- オーバーレイモジュール、拡張オーバーレイモジュール
アプリケーション自身がロードするリンクモジュールです。
- データバイナリ、拡張データバイナリ
アプリケーション自身がロードするデータです。
リンクされていないため、FAT を参照してロードします。
- ヘッド領域
「ROM 内登録データ」が格納され、「ROM 内登録データ」はブート時にメインメモリのシステム領域へロードされます。
この領域は、アプリケーション起動後に再ロードすることはできません。
イニシャルコード決定後にアプリケーション毎の ROM ヘッドテンプレートと呼ばれるバイナリファイルが提供されます。
- セキュア領域
「ARM9 常駐モジュール」の先頭部分が格納され、ブート時に「ROM 内登録データ」で指定されたメインメモリ上のアドレスにロードされます。
セキュリティ上、この領域はメインメモリの先頭 64KB 以内へ配置されなければなりません。
先頭の 2KB は暗号化領域となり、ロード時に復号されます。
具体的には、ARM9 常駐モジュール生成時に暗号化されたシステムコールライブラリを先頭へリンクするようにします。
アプリケーション起動後には再ロードできなくなるため、基本的にセキュア領域には .text と .rodata セクションのみを配置しなければ、ソフトリセットが実現できなくなることに注意してください。
- セキュア2領域
「ARM9 拡張常駐モジュール」の先頭部分が格納され、ブート時に「ROM 内登録データ」で指定されたメインメモリ上のアドレスにロードされます。
セキュア領域同様、アプリケーション起動後には再ロードできなくなります。
- ゲーム領域
「ARM9 常駐モジュール」のセキュア領域の後続部分と「ARM7 常駐モジュール」が、ブート時に「ROM 内登録データ」で指定されたアドレスにロードされます。
ブート可能なサイズについて、ARM9 側はメインメモリの先頭から 2.5MB（システム予約領域のために 16KB 差し引かれることに注意してください）を超えないようにしてください。また、ARM7 側についてはメインメモリのアドレス 0x02380000 から 256KB を超えないようにしてください。
この領域は、常に読み込み可能な領域となります。「オーバーレイモジュール」や「データバイナリ」は必要に応じてアプリケーションでロードしてください。
- ゲーム2領域
「ARM9 拡張常駐モジュール」のセキュア2領域の後続部分と「ARM7 拡張常駐モジュール」が、ブート時に「ROM 内登録データ」で指定されたアドレスにロードされます。
ブート可能なサイズについては、ARM9 側はメインメモリのアドレス 0x02400000 から 4MB を超えないようにしてください。また、ARM7 側については、アドレス 0x02E80000 から 1.03MB を超えないようにしてください。
この領域は、常に読み込み可能な領域となります。「拡張オーバーレイモジュール」や「拡張データバイナリ」は必要に応じてアプリケーションでロードしてください。

※「ROM 内登録データ」については「ニンテンドーDS/TWL ゲームカードマニュアル」を参照してください。

4. メインプロセッサコア (ARM946E-S)

メインプロセッサコアのブロック図を図 4-1 に示します。

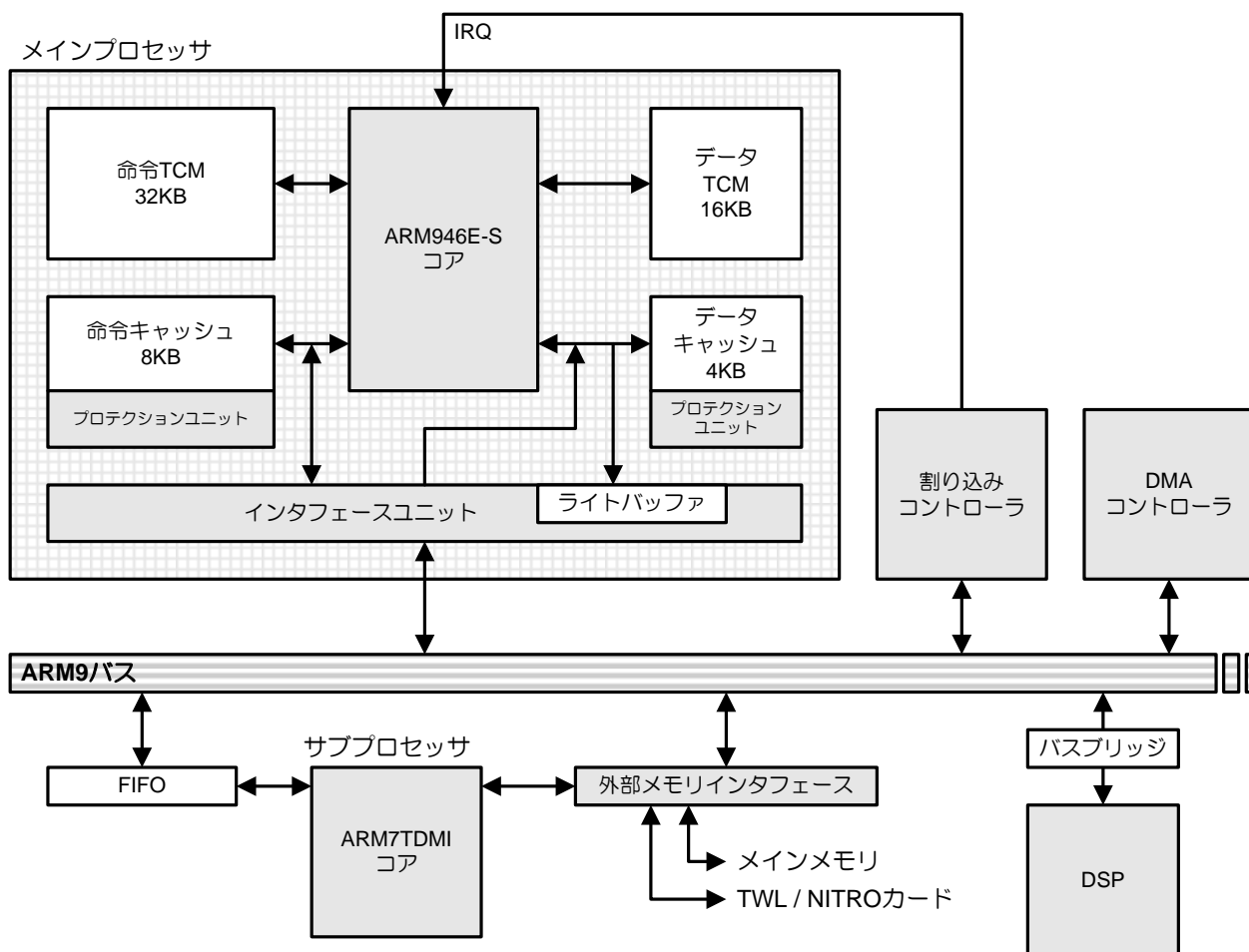


図 4-1 メインプロセッサコアのブロック図

4.1 プロテクションユニット

プロテクションユニットは、リード／ライト属性やキャッシュおよびライトバッファ使用の有無をプロテクションリージョン毎に設定することによってメモリを保護します。

プロテクションリージョンは、バックグラウンド（CPU からアクセスできる全アドレス空間）の 4G バイトから最大 8 つの領域を任意に設定することができ、リージョン番号の大きいプロテクションリージョンの設定ほど優先順位が高くなります。（プロテクションリージョンの設定が重複する領域はプロテクションリージョン番号の大きい方の設定が優先されます）

キャッシュやライトバッファの有効／無効や、リード／ライトの可否は、プロテクションリージョン毎に個別の設定を行うことができます。

● デバッグバージョンとリリースバージョン

TWL には、製品版とは別にメインメモリが 32M バイトに増設されたデバッグ用 TWL があります。

増設された 16M バイトの一部はデバッグ領域として使用されますが、それ以外の領域はアプリケーションの開発に使用することができます。デバッグ用と製品版の TWL ではメモリ構成が異なるため、プロテクションユニットの設定もそれぞれ異なったものになります。

4.2 TCM (Tightly-coupled Memory)

TCM は、直接 ARM9 コアと接続された高速なメモリです。独立したワーク RAM として使用することができます。ARM9 バスを経由しないため、DMA 等が ARM9 バスを使用している間においても TCM を使用して ARM9 が処理を行うことができます。

このため、使用頻度の高いプログラムや DMA 中にアクセスしたいデータなどを置くことによってパフォーマンスが向上します。

TCM は、命令用とデータ用の 2 種類があります。

なお、TCM は DMA によるアクセスができません。

4.2.1 命令 TCM

32K バイトの高速なメモリです。

ARM9 の 0x01FF8000 番地からマッピングされます。

高速で動かすプログラムや、動作クロック数を一定にさせたいプログラムを置くのに適しています。

例) 割り込み分岐ルーチン、グラフィックスライブラリなど

また、命令フェッチ時に ARM9 バスを経由しないため、DMA が ARM9 バスを使用している間に実行したいプログラムを置く用途としても効果的です。

例) ディスプレイリスト生成ルーチン、計算ルーチンなど

命令 TCM にはデータを配置することもできますが、命令フェッチとデータアクセスが衝突するとストールが発生します。

4.2.2 データ TCM

16K バイトの高速なメモリです。

メモリマップ上にデータ TCM を配置するアドレスは任意に設定することができます。

高速で読み書きするデータを置くのに適しています。

例) スタック、アクセス頻度の高いテーブルなど

また、アクセス時に ARM9 バスを経由しないため、ディスプレイリストをメインメモリからジオメトリエンジンへ DMA 転送している最中においても次のディスプレイリストを生成する等の用途に使用することができます。

ただし、TCM からの転送は ARM9 コアで行わなければなりません。

また、命令は配置できません。つまり命令フェッチによってデータアクセスがストールすることはありません。

4.3 キャッシュメモリ

ARM9 がメモリを参照すると、付近の 32 バイト（参照されたアドレスの上位 27 ビットが一致する範囲のデータ）をひとまとまりとしてキャッシュへ取り込み、次にその範囲が参照されたときにはキャッシュメモリの速度で高速にアクセスされます。

キャッシュにヒットした場合は、キャッシュから高速にデータが読み込まれます。

キャッシュにヒットしなかった場合は、メモリの内容をキャッシュライン単位で読み込む（ラインフェッチする）ことになります。

この際、置換アルゴリズムに従ってラインの内容を置き換えます。

なお、プロテクションユニットを有効にしなければキャッシュを使用することはできません。

キャッシュの仕様一覧を表 4-1 に示します。

容量	命令キャッシュ：8K バイト データキャッシュ：4K バイト
構成	4way セット アソシアティブ方式
キャッシュライン	8 ワード（32 バイト）
ライトミス時の動作	リードアロケート方式 ※1
置換アルゴリズム	ラウンドロビン※2 または 擬似ランダム※3 を選択可能
バス・スヌープ機能 ※4	なし
その他の機能 （命令キャッシュ）	ロックダウン 命令プリフェッチ
その他の機能 （データキャッシュ）	ロックダウン ライトスルーモード／ライトバックモード

表 4-1 キャッシュ仕様一覧

（※1）リードアロケート方式

ライトミス時にはメモリ（ライトバッファが有効であればライトバッファ）のみに対してライトを行います。キャッシュへのロードは行いません。

（参考：リードアロケート方式では、キャッシュにロードした後、ライトヒットとして扱います）

（※2）ラウンドロビン（推奨）

キャッシュラインを順番に置換する方式です。

最悪時のパフォーマンスが安定します。

（※3）擬似ランダム

キャッシュラインをランダムで置換する方式です。

ピーク時のパフォーマンスは向上しますが、最悪時のパフォーマンスが低下します。

（※4）バス・スヌープ機能

キャッシュに格納しているメモリ領域への書き込みを他のプロセッサや DMA が行ったかどうかを、バスを監視することによって検知する機能です。

ARM946E-S にはこの機能がないため、キャッシュとメモリの不整合が生じないように注意する必要があります（「コヒーレンスの確保」を参照してください）。

4.3.1 命令キャッシュ

8K バイトの高速なメモリです。命令コード専用です。

キャッシュにヒットしている間は ARM9 バスを経由しないため、ARM9 以外のバスマスタ（DMA やサブプロセッサ）が ARM9 バスを占有している間でもキャッシュ内のプログラムを動かすことができます。

4.3.1.1 ヒット／ミス判定

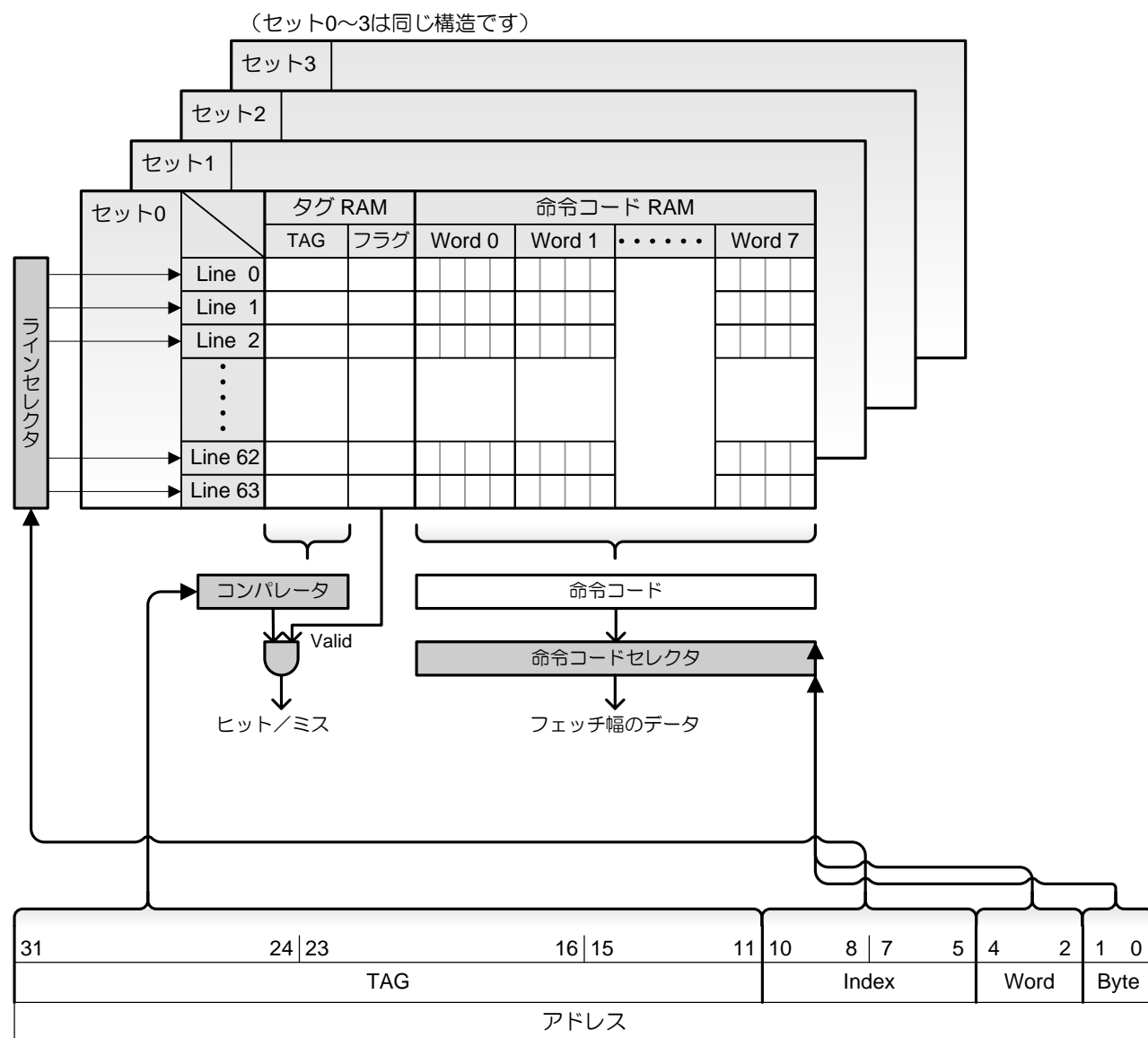
ARM9 がメモリから命令コードを取り出そうとすると、命令キャッシュのコントローラはメモリアドレスから Index ビットと TAG ビットを抜き出し、「Index 番目のキャッシュラインにおけるタグ RAM の内容」と「メモリアドレスの TAG ビット」を比較します。ARM946E-S は 4 セット構成のキャッシュであるため、4 つのラインに対して比較します。

もし 4 つのうちどれかが一致した場合、フラグの Valid ビットも有効であれば、そのライン内に目的の命令コードが存在している（ヒット）と判定されます。そうでなければミスと判定されます。

ヒットしている場合は、アドレスの Word および Byte からデータ RAM 内にある目的の命令コードが特定され、高速にアクセスすることができます。

ミスの場合は、ARM9 バスを通してメモリからキャッシュラインをフェッチします。

命令キャッシュの構造と動作を図 4-2 に示します。



ARM9が命令をフェッチするアドレス

図 4-2 命令キャッシュの構造と動作

命令キャッシュのタグ RAM について

- TAG

キャッシュラインに入っているデータのメモリアドレスが上位 21 ビット分格納されます。

- フラグ

キャッシュラインの有効／無効を示す「Valid ビット」から成ります。

4.3.2 データキャッシュ

4K バイトの高速なメモリです。データ専用です。

キャッシュにヒットしている間は ARM9 バスを経由しないため、ARM9 以外のバスマスタ（DMA やサブプロセッサ）が ARM9 バスを占有している間でもキャッシュ内のデータにアクセスすることができます。

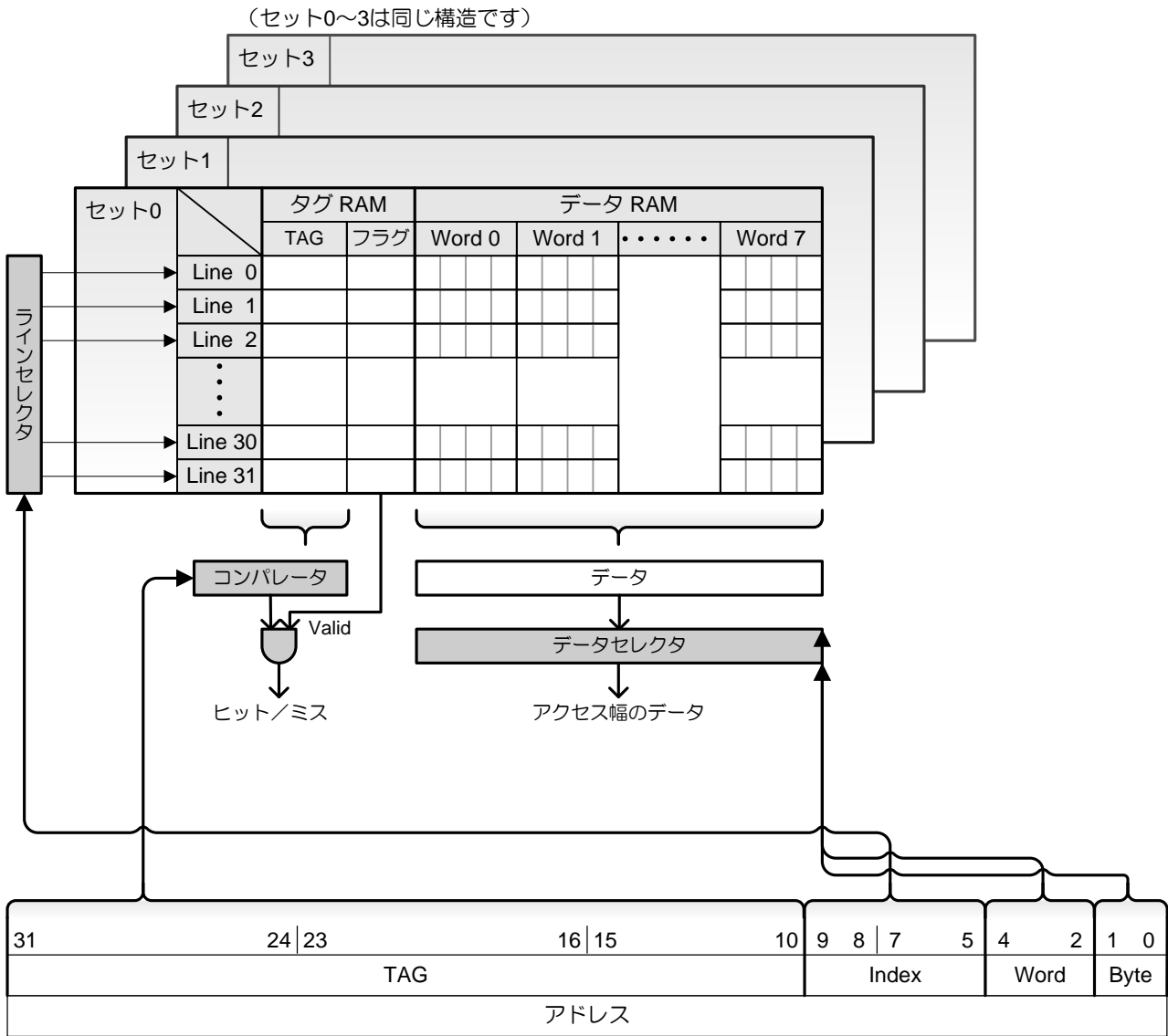
4.3.2.1 ヒット／ミス判定

ARM9 がメモリへデータの読み書きを行おうとすると、データキャッシュのコントローラはメモリアドレスから Index ビットと TAG ビットを抜き出し、「Index 番目のキャッシュラインにおけるタグ RAM の内容」と「メモリアドレスの TAG ビット」を比較します。ARM946E-S は 4 セット構成のキャッシュであるため、4 つのラインに対して比較します。もし 4 つのうちどれかが一致した場合、フラグの Valid ビットも有効であれば、そのライン内に目的のデータが存在している（ヒット）と判定されます。そうでなければミスと判定されます。

ヒットしている場合は、アドレスの Word および Byte からデータ RAM 内にある目的のデータが特定され、高速にアクセスすることができます。

ミスの場合は、ARM9 バスを通してメモリからアクセスすることになります。

データキャッシュの構造と動作を図 4-3 に示します。



ARM9がデータをリード/ライトするアドレス
(命令キャッシュとの相違点は、Indexビットが5ビットになっている点です)

図 4-3 データキャッシュの構造と動作

データキャッシュのタグ RAM について

- TAG
キャッシュラインに入っているデータのメモリアドレスが上位 22 ビット分格納されます。
- フラグ
キャッシュラインの有効/無効を示す「Valid ビット」、キャッシュとメモリの内容が一致/不一致を示す「ライン前半の Dirty ビット」、「ライン後半の Dirty ビット」から成ります。
キャッシュがメモリの内容と一致している状態を「クリーン」と呼び、一致していない状態を「ダーティ」と呼びます。
このうち、Dirty ビットについてはライトバックモード時においてのみ使用されます。
ライトバックモードについては、「ライトバッファ」章を参照してください。
Dirty ビットはラインの内容をメモリに書き戻す際に参照され、Dirty である場合は実際に書き戻されます。
Clean であった場合は書き戻されません（書き戻す必要がありません）。

4.3.3 キャッシュに対するオペレーション

キャッシュの操作には表 4-2 のように、全体を操作するもの、セットを指定するもの、ラインを指定して行うもの、メモリアドレスを指定することでキャッシュラインを特定して行うものがあります。

キャッシュ操作	命令キャッシュ	データキャッシュ
全体に対する直接操作	<ul style="list-style-type: none"> ・イネーブル／ディセーブル ・無効化 	<ul style="list-style-type: none"> ・イネーブル／ディセーブル ・無効化
セットに対する直接操作	<ul style="list-style-type: none"> ・ロックダウン 	<ul style="list-style-type: none"> ・ロックダウン
ラインに対する直接操作	—	<ul style="list-style-type: none"> ・クリーン ・クリーン&無効化
メモリアドレスに対応したキャッシュラインに対する操作	<ul style="list-style-type: none"> ・プリフェッチ ・無効化 	<ul style="list-style-type: none"> ・クリーン ・無効化 ・クリーン&無効化

表 4-2 キャッシュ操作一覧

「クリーン」、「無効化」、「クリーン&無効化」の操作はキャッシュ使用上の状態変化を起こすことがあります。キャッシュの状態と遷移については「コヒーレンシの確保」を参照してください。

各オペレーションについて

● クリーン (Clean)

キャッシュにおけるダーティなデータをメモリへ書き戻します。
 キャッシュ内のデータはそのまま残ります。
 ただしライトバッファが有効になっている場合は、実際にメモリへ書き戻されるまで遅延が発生します。

● 無効化 (Invalidate)

キャッシュ内のデータが無効となります。
 次にこのメモリ領域をリードすると、リードミスとなりメモリからキャッシュにラインフェッチされます。
 ラインフェッチされたキャッシュ内のデータは有効となります。

● クリーン&無効化 (Clean and Invalidate)

キャッシュにおけるダーティなデータをメモリへ書き戻すとともに無効化します。
 このため、次にこのメモリ領域にアクセスすると、メモリからキャッシュにラインフェッチされます。

注意事項

ライトバッファ（詳細は「ライトバッファ」章を参照してください）がフルの状態でクリーン&無効化命令を実行すると、既にクリーンなキャッシュラインは無効化されません。

● 命令プリフェッチ (prefetch)

プログラムがまだフェッチしていない命令コードを、命令キャッシュへ先読みさせることができます。
 コプロセッサ命令を用います。

● ロックダウン (Lockdown)

キャッシュの 1 セットをロックダウンすると、そのセットにおける全ラインがラインフェッチによって置換されなくなり、ひとかたまりのワーク RAM として使用できるようになります。
 しかし、その分キャッシュとして使用される領域が減ることになり、ミス率が上昇してしまいます。

● イネーブル／ディセーブル

キャッシュを使用する場合には、キャッシュをイネーブルにするとともにプロテクションユニットもイネーブルにする必要があります。

命令キャッシュおよびデータキャッシュは、プロテクションリージョン毎にイネーブル／ディゼーブルを行うことができます。
プロテクションリージョンとその設定については「プロテクションユニット」を参照してください。

注意事項

データプリロード（preload）はプログラムがまだアクセスしていないデータをデータキャッシュへ先読みさせる機能であり通常 PLD 命令を用いることで機能しますが、ARM946E-S では PLD 命令を認識しても何も動作しない仕様となっています。ARM946E-S ではデータプリロードが機能しませんので注意してください。

4.3.4 キャッシュの最適化

キャッシュは、プログラムの多くが持つメモリ参照の局所性を利用してメモリアクセスを高速化するものです。

- 時間的局所性：あるデータが参照されると、まもなく再びそのデータが参照される可能性が高いという性質
- 空間的局所性：あるデータが参照されると、その付近のデータも参照される可能性が高いという性質

参照の局所性が高いプログラムほどキャッシュのヒット率が高くなり、平均アクセス速度が速くなります。

プログラムの組み方によって、ある程度意識的に参照の局所性を高めることができます。

例えば、大きい 2 次元以上の配列を扱うループを作る場合、アドレスが連続して扱われるように組むことによって空間的局所性が高まります。

例)

ループ例 A	ループ例 B
<pre>u32 RESULT, TEST[0x100][0x100]; for(j=0; j<0x100; j++) { for(i=0; i<0x100; i++) { RESULT += TEST[i][j]; } }</pre>	<pre>u32 RESULT, TEST[0x100][0x100]; for(i=0; i<0x100; i++) { for(j=0; j<0x100; j++) { RESULT += TEST[i][j]; } }</pre>

ループ例 A の場合、内側のループで配列 TEST が 0x100 アドレス毎に参照されるため、初回のループではキャッシュにヒットしません。このため初回のループで次々とラインフェッチされますが、データキャッシュの容量は 32 ライン×4 セットの合計 128 ライン（0x80 ライン）であるため、ループ分収まり切らず 2 回目以降のループにおいても半分しかヒットしないことになります。

これに対してループ例 B では、内側のループで配列 TEST がアドレス順に参照されるため、最大限ヒットさせることができます。

4.4 ライトバッファ

ライトバッファは 32 バイト×16 段で、アドレスとデータ混在の FIFO メモリです。
アドレス／データ識別フラグでエントリタイプを決定し、アドレスエントリに対してはデータサイズが付加されます。
データをライトする際、メモリではなく高速なライトバッファに書き込むことで、ライト時に ARM9 がストールせずに済みます。ただしライトバッファがフルの場合に書き込みが行われると ARM9 はストールします。

プロテクションユニットが有効の場合、各プロテクションリージョンにおけるデータキャッシュとライトバッファの設定によって、データライト時のアクセスモードを表 4-3 のように選択することができます。
プロテクションリージョンについては「プロテクションユニット」を参照してください。

データキャッシュの設定	ライトバッファの設定	アクセスモード
ディセーブル	ディセーブル	NCNB モード ※2 (データキャッシュもライトバッファも無効)
ディセーブル	イネーブル	NCB モード (データキャッシュ無効、ライトバッファ有効)
イネーブル	ディセーブル ※1	ライトスルーモード
イネーブル	イネーブル	ライトバックモード

※1 ライトスルーモードの場合はライトバッファをディセーブル設定にしますが、ライトバッファは使用されます。

※2 NCNB モードの場合はライト時ライトバッファの内容が書き出され、リード時ライトバッファの書き出しより先にアクセスされます。

表 4-3 データライト時のアクセスモード

- ライトバックモード（推奨）
データ書き込み時にヒットすれば、ライトバッファには書き込まずキャッシュのみに書き込みます。
このためキャッシュの内容は必ずしもメモリの内容と一致しませんが、ライトは高速です。
CPU で書き換えたデータをメモリに反映させるには、クリーン操作を行う必要があります。
また、キャッシュが一杯の状態ではリードミスが発生すると、置換アルゴリズムによって追い出すキャッシュラインがダーティであった場合にライトバッファへ書き戻されます。
- ライトスルーモード
データ書き込み時にヒットすれば、キャッシュに書き込むと同時にライトバッファにも書き込みます。
このため ARM9 の書き込みによってキャッシュラインがダーティになることはありませんが、ライトは低速です。
また、キャッシュが一杯の状態ではリードミスした場合は、置換アルゴリズムによって単にキャッシュラインが上書きされます。

ライトバックモード、ライトスルーモードのいずれの場合も、ライトミス時にはライトバッファのみに書き出されます。
また、ラインフェッチされる場合はデータの整合性を保つため前もってライトバッファの内容が吐き出されます。

注意事項

ライトスルーモードを使用する際は、ライトするメモリのアクセス幅（ライト時）に注意してください。
(例：VRAM に対して 8 ビット幅のアクセスはできません)
メモリ毎のアクセス幅については「メモリ」章を参照してください。
ライトバックモードおよびライトスルーモードにおけるキャッシュの状態遷移や制御については、「コヒーレンシの確保」を参照してください。

4.4.1 ライトバッファに関するオペレーション

- ライトバッファエンプティ待ち
ライトバッファの内容がすべてメモリに書き込まれるまで ARM9 をストールさせることが可能です。
CPU で書き換えた内容を確実にメモリへ反映するにはこの操作を行ってください。
なお、I/O レジスタなどのキャッシュ不可／バッファ不可領域への書き込み時は、ライトバッファが空になるまで CPU は停止しますのでライトバッファエンプティ待ちは必要ありません。
この操作の必要性については「コヒーレンシの確保」を参照してください。

4.5 コヒーレンシの確保

キャッシュを使用する場合は、メモリとキャッシュの内容に不整合が生じないように注意する必要があります。キャッシュの状態は、キャッシュライン毎にタグ RAM 内のフラグで管理されます。フラグは、Valid ビット（1 ビット）と Dirty ビット（2 ビット）から構成され、Dirty ビットの 2 ビットはそれぞれラインの前半と後半の状態を表しています。データキャッシュのライトバックモード時はこれら 3 ビットが使用されますが、ライトスルーモード時や命令キャッシュについては Valid ビットのみ使用されます。

4.5.1 ライトバックモード時

フラグの状態によって表 4-4 のようにライトバックモード時のキャッシュラインの状態を定義します。

状態名	フラグの状態		説明
	Valid	Dirty	
Dirty	1	1	キャッシュラインは有効でメモリの内容と異なります
Clean	1	0	キャッシュラインは有効でメモリの内容と同じです
Invalid	0	*	キャッシュラインは無効です

表 4-4 キャッシュラインの各状態（ライトバックモード）

キャッシュコントローラによる自動管理

図 4-4 のように、ARM9 からのアクセスによるリードミス/ヒット、ライトミス/ヒット、および置換アルゴリズムによる状態遷移は自動的に行われます。

また、置換アルゴリズムによって Valid かたダーティなラインが置換される場合は、置換される前にメモリ（ライトバッファが有効の場合はライトバッファ）に書き戻されます。実際には、ライン単位ではなくラインの前半と後半で処理が行われるため、書き戻されるデータ量は 0, 16, 32 バイトのいずれかになります。

ユーザーが行わなければならない管理

ARM946E-S にはバス・スヌープ機能がないため、キャッシュされているメモリに ARM9 以外のバスマスタ（サブプロセッサや DMA 等）によるアクセスが行われる場合、そのキャッシュラインを手動で操作する必要があります。ARM9 以外のバスマスタによるメモリへのライトが行われた場合は該当するキャッシュラインを無効化してください。また、ARM9 以外のバスマスタによってメモリがリードされる場合は前もってキャッシュラインをクリーンにし、ライトバッファのエンプティ待ちを行ってください。

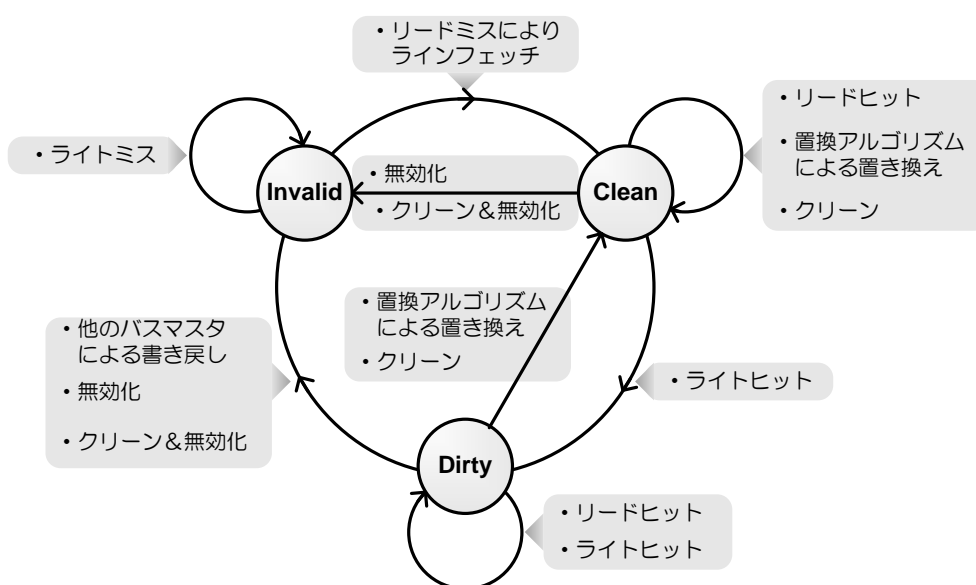


図 4-4 キャッシュライン状態遷移図（ライトバックモード）

具体例

- オーバーレイ等でメインメモリ上のプログラムを書き換えた場合
該当領域の命令キャッシュを無効化（Invalidate）してください。

4.5.2 ライトスルーモード時

フラグの状態によって表 4-5 のようにライトスルーモード時のキャッシュラインの状態を定義します。

状態名	フラグの状態		説明
	Valid	Dirty	
Clean	1	*	キャッシュラインは有効でメモリの内容と同じです
Invalid	0	*	キャッシュラインは無効です

表 4-5 キャッシュラインの各状態（ライトスルーモード）

キャッシュコントローラによる自動管理

図 4-5 のように、ARM9 からのアクセスによるリードミス/ヒット、ライトミス/ヒット、および置換アルゴリズムによる状態遷移は自動的に行われます。

ユーザーが行わなければならない管理

ARM946E-S にはスヌープ機能がないため、キャッシュされているメモリに ARM9 以外のバスマスタ（サブプロセッサや DMA 等）によるアクセスが行われる場合、そのキャッシュラインを手動で操作する必要があります。
ARM9 以外のバスマスタによるメモリへのライトが行われた場合は該当するキャッシュラインを無効化してください。
また、ARM9 以外のバスマスタによってメモリがリードされる場合は、前もってライトバッファのエンプティ待ちを行ってください。

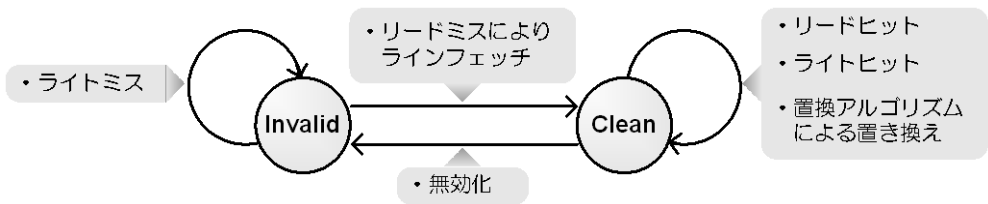


図 4-5 キャッシュライン状態遷移図（ライトスルーモード）

具体例

- オーバーレイ等でメインメモリ上のプログラムを書き換えた場合
該当領域の命令キャッシュを無効化（Invalidate）してください。

5. 表示

※この章の内容は表示ステータスに INI ビットが追加されたこと以外に、NITRO からの変更はありません。

5.1 表示システム

表示システムのブロック図を図 5-1 に示します。

ブロック図における各セレクトは、表 5-1 のレジスタの選択フラグによってコントロールできます。

セレクト名	レジスタ名	フラグ名
SEL DISP	DISPCNT	表示モード選択
SEL BG0	同上	BG0 の 2D/3D 表示選択
SEL DISP VRAM	同上	表示 VRAM 選択
SEL A	DISPCAPCNT	キャプチャ元 A 選択
SEL B	同上	キャプチャ元 B 選択
SEL CAP	同上	キャプチャモード選択
SEL CAP VRAM	同上	キャプチャデータ書き込み先 VRAM 選択
SEL LCD	POWCNT	LCD 出力先切り換え

表 5-1 セレクトとレジスタの選択フラグ対応表

SEL DISP によってグラフィックス表示/VRAM 表示/メインメモリ表示のいずれかが選択された後の画像出力を「画像出力 A」とします。

一方、2D グラフィックスエンジン B の画像出力を「画像出力 B」とします。

画像出力 A と B はそれぞれマスター輝度アップ/ダウン A と B を通り、LCD に出力する「表示出力 A」および「表示出力 B」となります。

最終的に LCD へ出力する際には、これらの表示出力を重ね合わせることはできません。

「表示出力 A を上画面 LCD へ出力し、表示出力 B を下画面 LCD へ出力する」もしくは、「表示出力 A を下画面 LCD へ出力し、表示出力 B を上画面 LCD へ出力する」のいずれかを選択することとなります。

LCD を 1 つしか必要としないゲームにおいては、いずれかの LCD 表示をディセーブルにすることができます。

詳細は「パワーマネジメント」を参照してください。

画像出力 A では、グラフィックス表示で 2D グラフィックスと 3D グラフィックスをブレンディングして表示することができます。

3D 画像生成のハードウェア・ブロックについては、「3D グラフィックス」を参照してください。

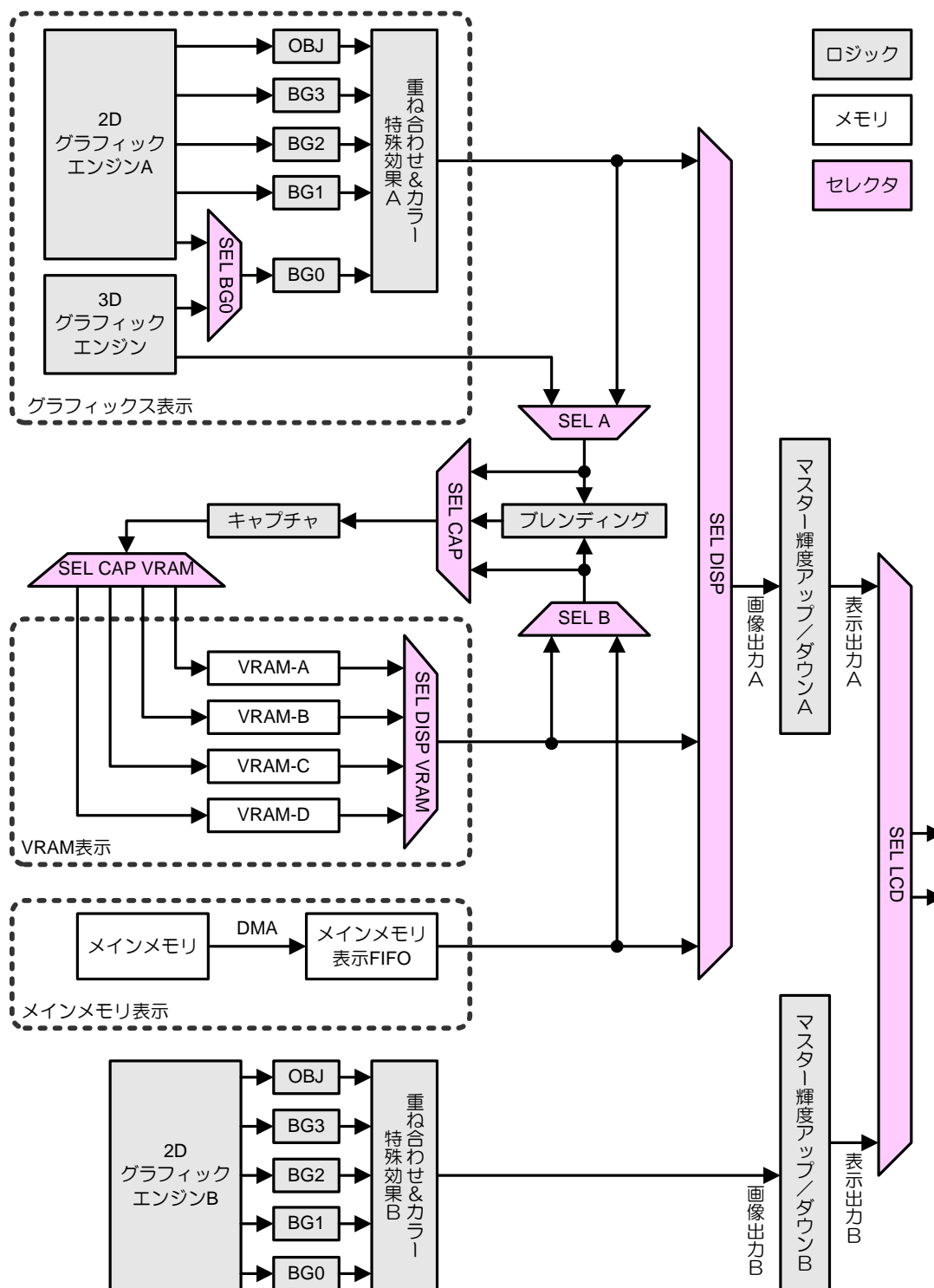


図 5-1 表示システムブロック図

5.2 LCD

TWL に搭載されている 2 つの LCD コントローラの仕様を以下に示します。

5.2.1 LCD コントローラ仕様

LCD コントローラの LCD クロック仕様を表 5-2 に、LCD 走査タイミングを図 5-2 に、LCD 走査タイミングの仕様を表 5-3 にそれぞれ示します。

LCD クロック	周波数（時間）
画像処理クロック	33.513982Mhz（ 29.838293ns）
ドットクロック	5.585664Mhz（179.029757ns）※

※ドットクロックは画像処理クロックの 6 分周となっています。

表 5-2 LCD クロックの仕様

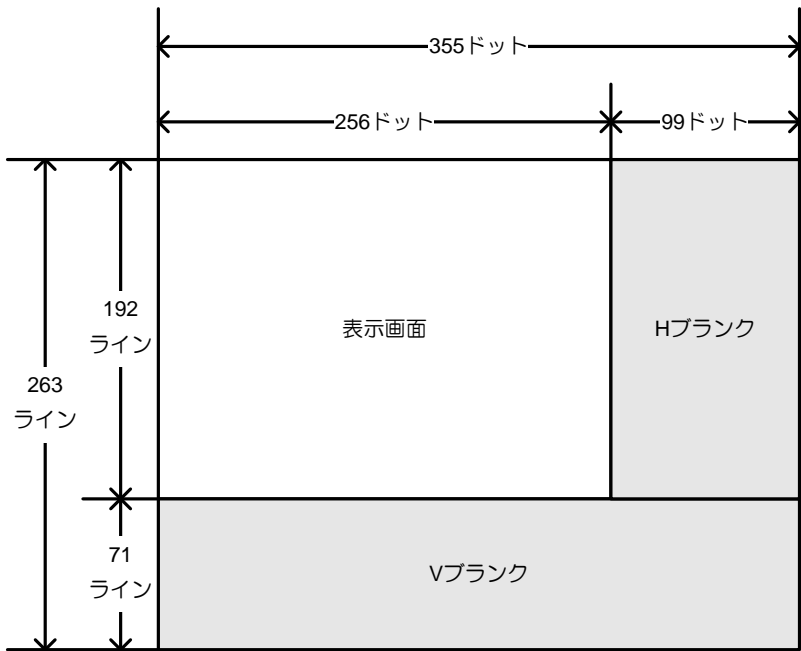


図 5-2 LCD 走査タイミング

項目		スペック	期間	参考：AGB 値
表示画面 サイズ	水平ドット数	256 ドット	45.8316us	240 ドット（57.221us）
	垂直ライン数	192 ライン	12.2027ms	160 ライン（11.749ms）
総ドット数	水平ドット数	355 ドット	63.5556us	308 ドット（73.433us）
	垂直ライン数	263 ライン	16.7151ms	228 ライン（16.743ms）
ブランキング	H ブランクドット数	99 ドット	17.7239us	68 ドット（16.212us）
	V ブランクライン数	71 ライン	4.5124ms	68 ライン（4.994ms）
走査周期	H 周期	15.7343KHz	63.5556us	13.618KHz（73.443us）
	V 周期	59.8261Hz	16.7151ms	59.727Hz（16.743ms）

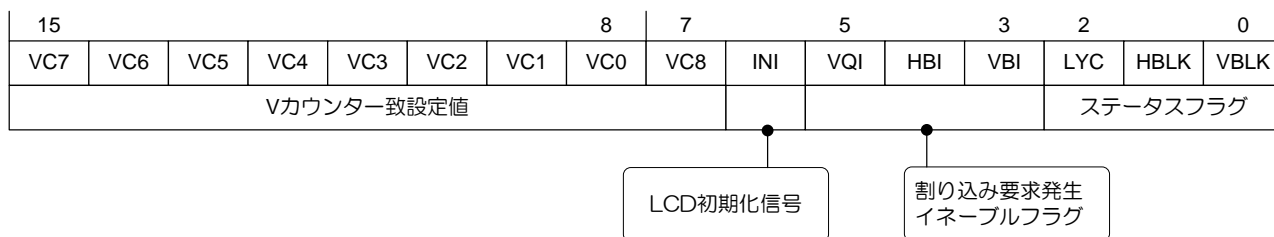
表 5-3 LCD 走査タイミング仕様

3D レンダリングエンジンの V ブランク期間は、191～213 の 23 ラインです（「レンダリングエンジン」を参照してください）。

5.3 表示ステータス

表示ステータスレジスタ

名称 DISPSTAT アドレス 0x04000004 属性 R/W 初期値 0x0000



● [d15～d07]：V カウンタ一致設定値

VC8 が d07 に配置される（AGB との互換性のため）ことに注意してください。

0～262 の値が設定可能です。263 以上の値を設定した場合の動作は保証されません。

● INI[d06]：LCD 初期化信号

0	LCD 初期化状態
1	LCD 表示状態

このビットへの書き込みは無効となります。
システム設定の拡張機能が OFF に設定されている場合は 0 固定となります。

● [d05～d03]：割り込み要求発生イネーブルフラグ

● VQI[d05]：V カウンタ一致割り込み要求発生イネーブルフラグ

0	ディセーブル
1	イネーブル

● HBI[d04]：H ブランク割り込み要求発生イネーブルフラグ

0	ディセーブル
1	イネーブル

イネーブルに設定し、割り込みイネーブルレジスタ（IE）で H ブランク割り込みを許可した場合、表示期間中だけでなく、V ブランク期間も含めた LCD の垂直 263 ライン（0～262 ライン）すべてに H ブランク割り込み要求をかけることができます。

● VBI[d03]：V ブランク割り込み要求発生イネーブルフラグ

0	ディセーブル
1	イネーブル

● [d02～d00]：ステータスフラグ

● LYC[d02]：V カウンタ一致検出フラグ

0	一致期間中でない
1	一致期間中

● HBLK[d01]：H ブランク検出フラグ

0	H ブランク期間中でない
1	H ブランク期間中

● VBLK[d00]：V ブランク検出フラグ

0	V ブランク期間中でない
1	V ブランク期間中

注意事項

V ブランク検出フラグは 192 ラインに入った瞬間に 1 となり、262 ラインに入った瞬間に 0 となります。これは、OBJ 描画回路による OBJ-VRAM および OAM へのアクセスが実際の表示より 1 ライン手前である 262 ラインから開始されるためです。なお、OBJ-VRAM および OAM へのアクセスが終了するタイミングは H ブランク期間に OBJ 処理を行うかどうかによって異なります。一方、BG-VRAM や、BG パレット RAM、OBJ パレット RAM については、0 ラインからアクセスが開始され、191 ラインまでで終了します。これらをまとめると、表 5-4 のようになります。

グラフィックスエンジンがアクセスするメモリ			V カウンタ値
OBJ 描画回路による	OBJ-VRAM OAM	へのアクセス期間	
		H ブランク期間に OBJ 処理を行う	0～191, 262
		H ブランク期間に OBJ 処理を行わない	0～190, 262
描画回路による	BG-VRAM BG パレット RAM OBJ パレット RAM	へのアクセス期間	0～191
（参考） V ブランク検出フラグが 1 となる期間			192～261

表 5-4 グラフィックスエンジンがメモリへアクセスしている期間

V カウンタレジスタ

名称 VCOUNT アドレス 0x04000006 属性 R/W 初期値 0x0000

15							8	7								0	
							V8	V7	V6	V5	V4	V3	V2	V1	V0		
							V カウンタ値										

●[d08～d00] : V カウンタ値

DISPSTAT レジスタの V カウンタ一致設定値とはビット配列が異なり、通常のビット配置です。

1) 値を読み出す場合

LCD の全 263 ラインのうち、現在どのラインを表示中であるかを読み出すことができます。

読み出す値は 0～262 となります。

0～191 であれば描画中、192～262 であれば V ブランク期間ということになります。

LCD の表示タイミングについては、「LCD」を参照してください。

2) 値を書き込む場合

書き込んだ値は、ハードウェアの V カウンタ更新時に反映されます。

このレジスタを使うことで、複数台の TWL 間で通信を行う際に V カウント値を調整して、すべての TWL の V 周期を同期させることが可能となります。

現在の V カウント値が 202～212 であることを確認した上で 202～212 の範囲の値を書き込んでください。

この範囲外の値を書き込んだ場合の 3D エンジンの動作は保証できません。

注意事項

表示回路の VRAM へのアクセスと CPU からの VRAM へのアクセスが衝突した場合は、CPU からのアクセスが待たされます。

LCD コントローラのドットクロックが画像処理クロックおよびシステムクロックの 6 分周となっていますので、LCD コントローラが VRAM にアクセスするタイミングは 6 サイクルに 1 回となります。

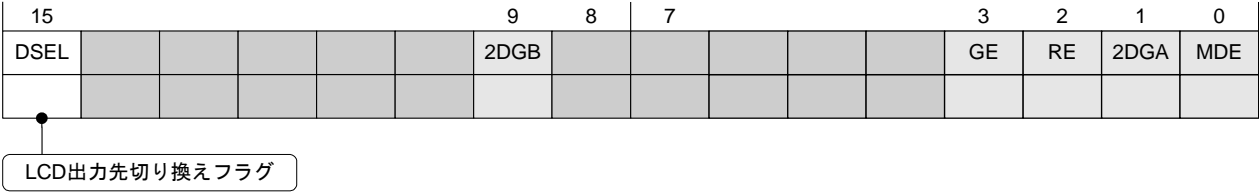
このタイミングで、CPU から同時にアクセスした場合は CPU のアクセスが 1 サイクル待たされることになります。

5.4 表示制御

5.4.1 上画面 LCD／下画面 LCD 出力先切り換え

パワーコントロールレジスタ

名称 POWCNT アドレス 0x04000304 属性 R/W 初期値 0x0000



●DSEL[d15]：LCD 出力先切り換えフラグ

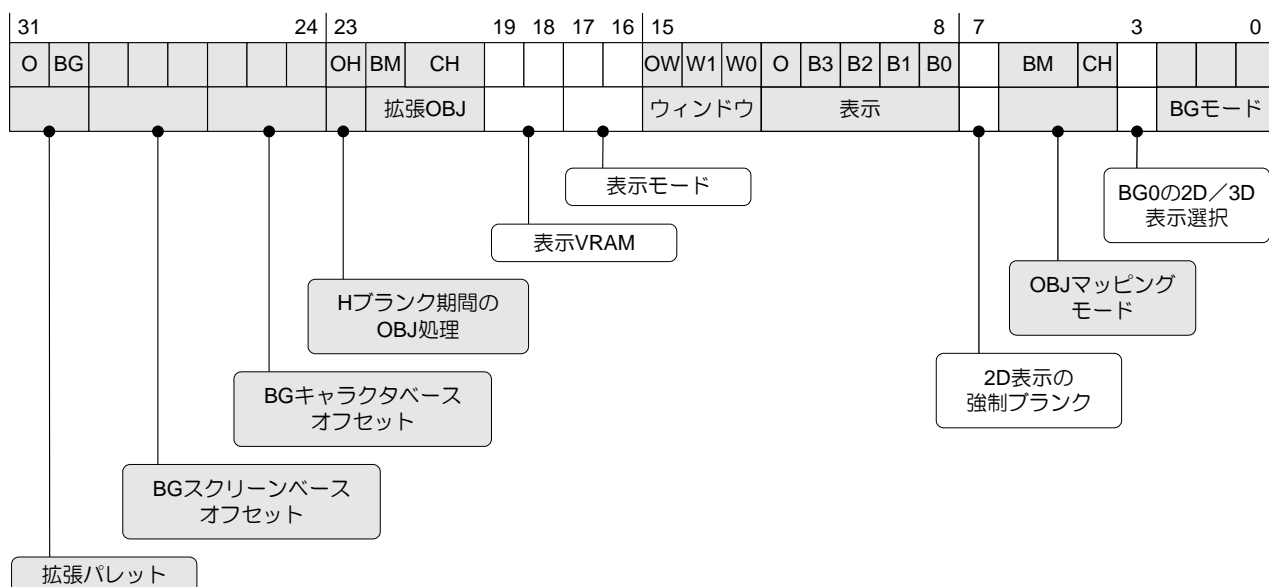
0	表示出力 A を下画面 LCD に出力し、表示出力 B を上画面 LCD に出力する
1	表示出力 A を上画面 LCD に出力し、表示出力 B を下画面 LCD に出力する

LCD 出力先の切り換えは、上記レジスタを設定することにより遅延なく行われます。

5.4.2 2D グラフィックスエンジン A の表示制御

表示コントロールレジスタ

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



●[d19～d18]：表示 VRAM

VRAM 表示モード（d17～d16 の「表示モード」参照）時に表示する VRAM を選択します。

00	VRAM-A
01	VRAM-B
10	VRAM-C
11	VRAM-D

●[d17～d16]：表示モード

表示 OFF にすると、2D/3D グラフィックス、VRAM 表示、メインメモリ表示のすべてが選択されず白色表示となります。

グラフィックス表示モードでは、2D および 3D グラフィックスを表示します。

VRAM 表示モードでは、VRAM 上に格納されたビットマップデータを表示します。

メインメモリ表示モードでは、メインメモリに格納されたビットマップデータを表示します（DMA の設定が必要）。

詳細については、それぞれの章を参照してください。

0	表示 OFF
1	グラフィックス表示
2	VRAM 表示
3	メインメモリ表示

●[d07]：2D 表示の強制ブランク

CPU によって強制的に 2D 表示を停止します。

2D 表示が止まるため、BG0 を利用している 3D グラフィックスも表示されなくなります。

強制ブランク中は 2D グラフィックス回路が VRAM へアクセスせず、LCD 画面は白を表示した状態になります。

ただし強制ブランク中においても、内部 HV 同期カウンタは動作しています。

内部 HV 同期カウンタが表示期間中に強制ブランクを変更すると、ON→OFF にした場合は設定直後に、OFF→ON にした場合は 3 ライン後の先頭から ON/OFF が切り換わります。

●[d03] : BG0 の 2D/3D 表示選択

BG のうち一面 (BG0) を 2D グラフィックスに使用するか、3D グラフィックスに使用するかを設定します。

3D グラフィックスを選択した場合、BG0 に対する 2D グラフィックスの機能が限定され、カラー特殊効果の仕様も変わります。

「レンダリング後の 3D 面に適用できる 2D グラフィックス機能」を参照してください。

0	2D グラフィックスを表示する
1	3D グラフィックスを表示する

●その他のビット

DISPCNT レジスタのうち、ここで説明していないビットについては下記を参照してください。

2D グラフィックス機能の表示制御に関連するビットについては、「2D グラフィックス」章を参照してください。

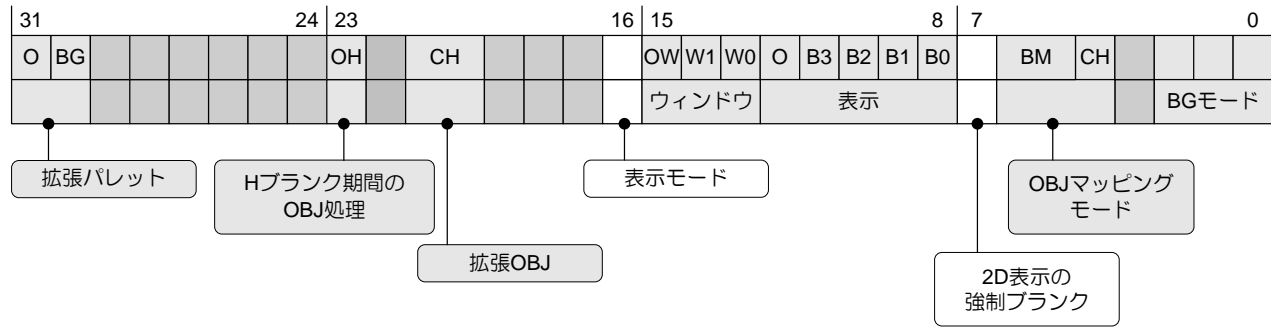
BG に関連するビットについては、「BG」章を参照してください。

OBJ に関連するビットについては、「OBJ」章を参照してください。

5.4.3 2D グラフィックスエンジン B の表示制御

表示コントロールレジスタ 1

名称 DB_DISPCNT アドレス 0x04001000 属性 R/W 初期値 0x00000000



●[d16] : 表示モード

0	表示 OFF
1	表示 ON

●[d07] : 2D 表示の強制ブランク

CPU によって強制的に 2D グラフィックス回路を停止します。
強制ブランク中は 2D グラフィックス回路が VRAM へアクセスせず、LCD 画面は白を表示した状態になります。
ただし強制ブランク中においても、内部 HV 同期カウンタは動作しています。
内部 HV 同期カウンタが表示期間中に強制ブランクを変更すると、ON→OFF にした場合は設定直後に、OFF→ON にした場合は 3 ライン後の先頭から ON/OFF が切り換わります。

5.4.4 表示モード

表示出力 A（2D グラフィックスエンジン A）側では表 5-5 のように、グラフィックス回路で生成した画像を表示するモードのほかに、VRAM やメインメモリ上のビットマップデータを表示するモードがあります。
表示出力 B（2D グラフィックスエンジン B）側では表 5-6 のように、モードの選択がグラフィックス表示の ON/OFF のみとなります。

表示モード 番号	表示モード	表示サイズ	フレーム レート	機能			
				3D 表示	キャラ クタ BG 表示	ビット マップ BG 表示	OBJ 表示
0	表示 OFF	—	—	—	—	—	—
1	グラフィックス表示	256×192	60 fps	○	○	○	○
2	VRAM 表示	256×192	60 fps	×	×	○	×
3	メインメモリ表示	256×192	60 fps	×	×	○	×

表 5-5 表示モード一覧（2D グラフィックスエンジン A）

表示モード 番号	表示モード	表示サイズ	フレーム レート	機能		
				キャラ クタ BG 表示	ビット マップ BG 表示	OBJ 表示
0	表示 OFF	—	—	—	—	—
1	グラフィックス表示	256×192	60 fps	○	○	○

表 5-6 表示モード一覧（2D グラフィックスエンジン B）

図 5-3 は、図 5-1 の「表示システムブロック図」を表示出力 A 側の表示モードに絞って簡略化したものです。

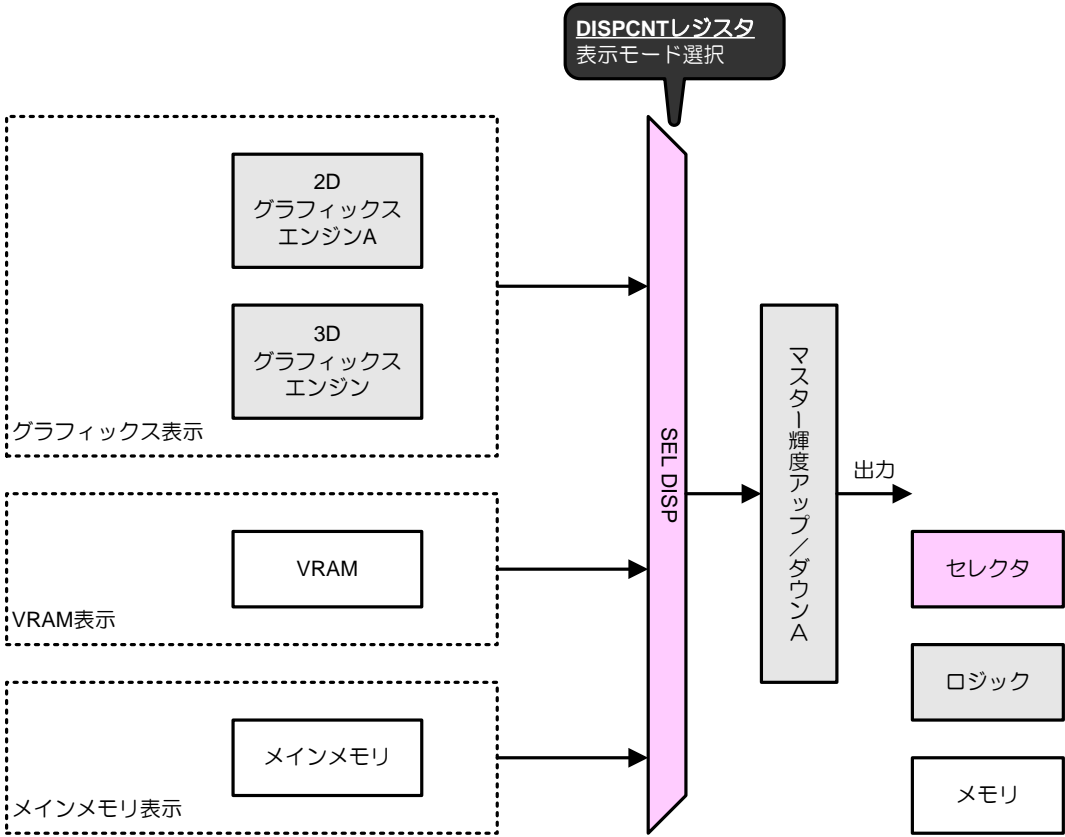


図 5-3 表示モードの選択（表示出力 A 側のみ）

5.4.4.1 グラフィックス表示モード

2D および 3D のグラフィックス機能によって生成した画像を表示します。

表示システム全体については、「表示システム」のブロック図（図 5-1）を参照してください。

グラフィックスの各機能については、「2D グラフィックス」および「3D グラフィックス」を参照してください。

●グラフィックス表示モード例 1

図 5-4 の例では、3D レンダリング結果を 2D 面と重ね合わせて表示しています。

3D の表示は BG0 面として扱われますが、BG0 の 2D グラフィックス機能は限定され、カラー特殊効果の仕様も変わります（「レンダリング後の 3D 面に適用できる 2D グラフィックス機能」を参照してください）。

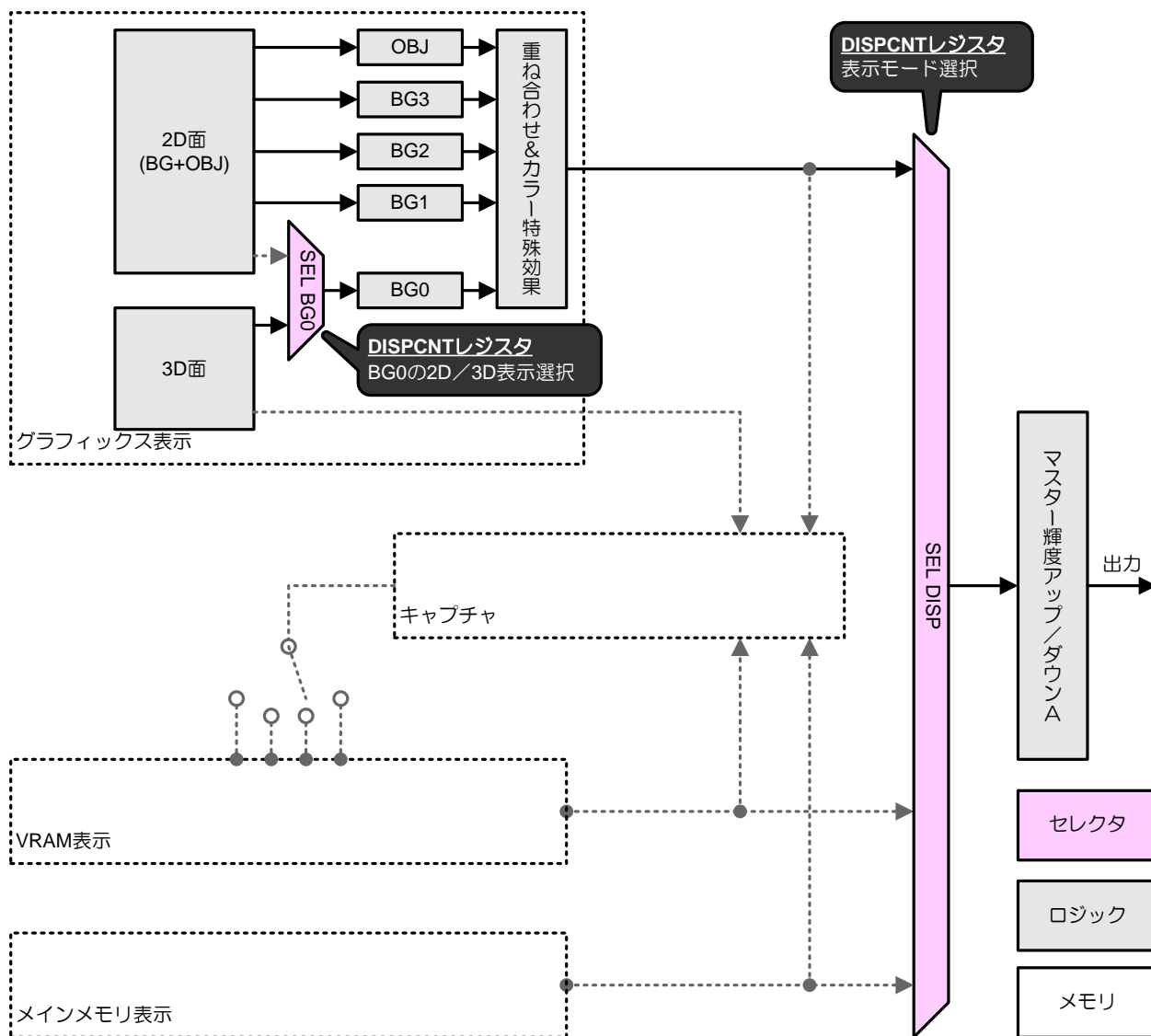


図 5-4 2D と 3D の重ね合わせ表示例

●グラフィックス表示モード例 2

図 5-5 の例では、3D レンダリング結果をビットマップ OBJ に貼り付けて表示しています。

レンダリングエンジンのクリア α 値を0にして3Dレンダリング結果をキャプチャし、次のフレームでそのVRAMをRAMバンクコントロールレジスタによってビットマップOBJに割り当てれば、3Dレンダリング結果をOBJとして表示させることができます。

このとき、3D の α プレンディング処理において α 値が 0 のままに保たれていた部分が「抜き」になります（キャプチャ機能およびレンダリングエンジンの α プレンディングの仕様を参照してください）。

なお、この例では VRAM-A と VRAM-B を交互に LCDC と OBJ-VRAM に割り当て、ダブルバッファリングを行います。

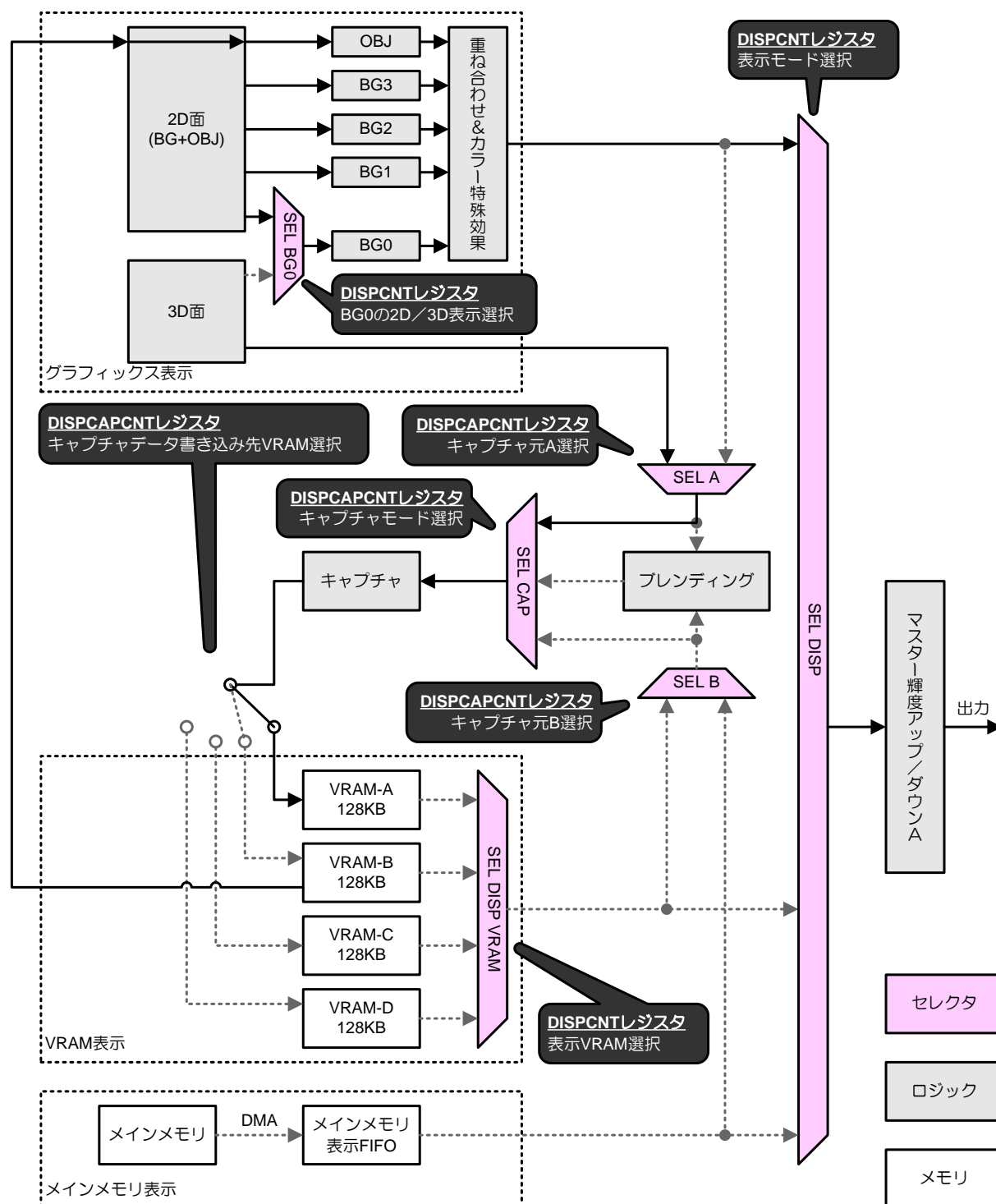


図 5-5 3D レンダリング結果のビットマップ OBJ 表示の例

5.4.4.2 VRAM 表示モード

DISPCNT レジスタが VRAM 表示モードに設定されると、次の表示開始から VRAM 上にある 1 フレーム分のビットマップデータを表示します。

どの VRAM を使用するかは DISPCNT レジスタで設定することができます。

2D および 3D グラフィックス回路とは別系統で VRAM を表示します。

このため VRAM 表示モードで画像を表示しながら同時にグラフィックス回路により画像を生成し、その画像を VRAM へ取り込む（キャプチャ）ことも可能です。

「表示システム」のブロック図（図 5-1）を参照してください。

表示する VRAM と画像を取り込む VRAM は同じ VRAM に設定しても構いません。

画像の取り込みに関する詳細は、「表示キャプチャ」を参照してください。

VRAM 表示モードでのピクセルデータフォーマットは以下の通りです。

VRAM 表示モードデータフォーマット

15	14	10	9	8	7	5	4	0
	BLUE			GREEN			RED	
	ピクセル色データ							

図 5-6 に LCD ピクセルの VRAM アドレスマップを示します。

	ドット 0	1	2	3		253	254	255
ライン 0	0h	2h	4h	6h		1FAh	1FCh	1FEh
1	200h	202h	204h	206h		3FAh	3FCh	3FEh
2	400h	402h					5FCh	5FEh
3	600h	602h					7FCh	7FEh
4	800h							9FEh
187	17600h							177FEh
188	17800h	17802h					179FCh	179FEh
189	17A00h	17A02h					17BFCh	17BFEh
190	17C00h	17C02h	17C04h	17C06h		17DFAh	17DFCh	17DFEh
191	17E00h	17E02h	17E04h	17E06h		17FFAh	17FFCh	17FFEh

図 5-6 LCD ピクセルの VRAM アドレスマップ

●VRAM 表示モード例

図 5-7 では、グラフィックス回路により生成した画像をキャプチャ機能によって VRAM に取り込み、その画像を VRAM 表示モードで表示しています。

キャプチャする際に表示用 VRAM とブレンディングすることによって、モーションブラー効果が得られます。

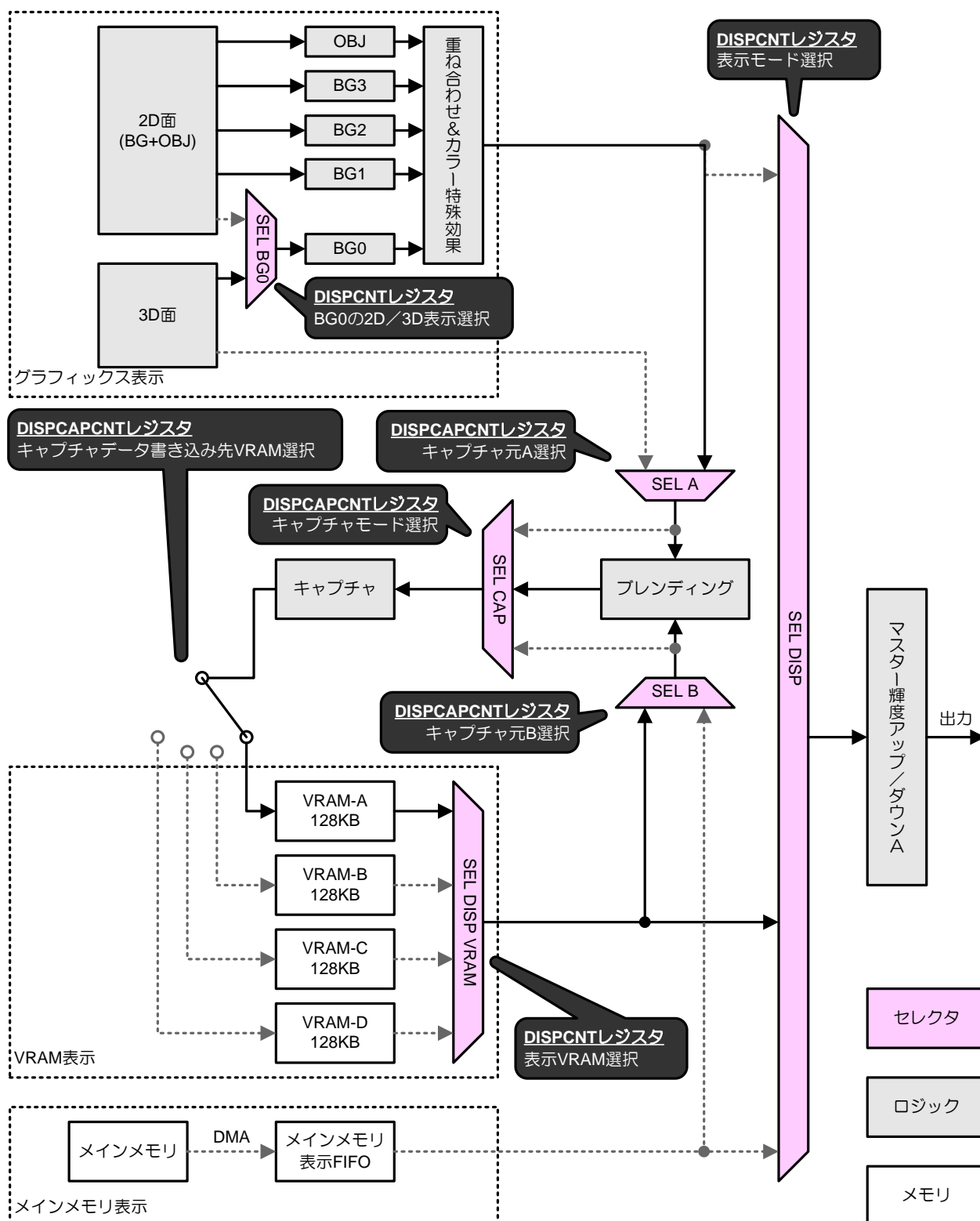


図 5-7 表示キャプチャを利用したモーションブラー効果の例

5.4.4.3 メインメモリ表示モード

メインメモリ上のビットマップデータを表示する機能です。

DISPCNT レジスタがメインメモリ表示モードに設定されると、次の表示開始からメインメモリ表示 FIFO レジスタのデータを LCD モジュールへ転送します。

転送毎に DMA ヘデータリクエストを行います。

メインメモリ表示 FIFO レジスタと LCD モジュール間は 4 段の FIFO になっており、LCD モジュールは一度に 4 ワード取り込みます。このためメインメモリ表示 FIFO レジスタへの書き込みは一度に 4 段分行うようにしてください。

具体的には、DMA の転送ビット幅を 32 ビットに設定しワードカウントを 4 に設定した上で、DMA 起動モードをメインメモリ表示モードに設定してください。

メインメモリ表示モードでは、DMA のソースアドレスは必ずメインメモリの領域に設定してください。

表 5-7 にメインメモリ表示モードを使用する場合の DMA 設定を示します。

設定項目	設定値
ソースアドレス	メインメモリ
転送ビット幅	32 ビット
ワードカウント	4

表 5-7 メインメモリ表示モードを使用する場合の DMA 設定

DMA 設定の詳細については、「DMA」を参照してください。

2D および 3D グラフィックス回路とは別系統でメインメモリを表示します。このためメインメモリ表示モードで画像を表示しながら同時にグラフィックス回路により画像を生成し、その画像を VRAM へ取り込む（キャプチャ）ことも可能です。「表示システム」のブロック図（図 5-1）を参照してください。

メインメモリ表示 FIFO レジスタ

名称 DISP_MMEN_FIFO アドレス 0x04000068 属性 R/W 初期値 0x00000000

31	30	26	25	24	23	21	20	16	15	14	10	9	8	7	5	4	0
BLUE		GREEN		RED				BLUE		GREEN		RED					
ODD										EVEN							

図 5-8 に LCD ピクセルのメインメモリ表示 FIFO レジスタの EVEN/ODD マップを示します。

	ドット 0	1	2	3		253	254	255
ライン 0	EVEN	ODD	EVEN	ODD		ODD	EVEN	ODD
1	EVEN	ODD	EVEN	ODD		ODD	EVEN	ODD
2	EVEN	ODD					EVEN	ODD
3	EVEN	ODD					EVEN	ODD
4	EVEN							ODD
187	EVEN							ODD
188	EVEN	ODD					EVEN	ODD
189	EVEN	ODD					EVEN	ODD
190	EVEN	ODD	EVEN	ODD		ODD	EVEN	ODD
191	EVEN	ODD	EVEN	ODD		ODD	EVEN	ODD

図 5-8 LCD ピクセルのメインメモリ表示 FIFO レジスタ EVEN/ODD マップ

5.5 表示キャプチャ

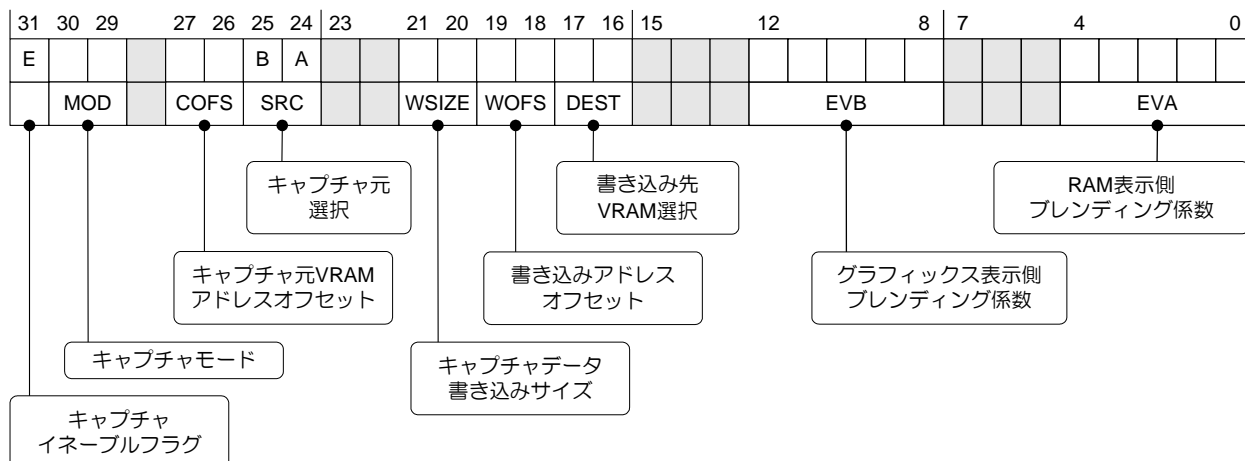
2D および 3D グラフィックスや、VRAM、メインメモリの画像出力を VRAM へ取り込む機能です。

表示出力 A（2D グラフィックスエンジン A）側の画像出力のみキャプチャ可能です。

2 つのソース画像をブレンディングしてキャプチャすることも可能です。

表示キャプチャコントロールレジスタ

名称 DISPCAPCNT アドレス 0x04000064 属性 R/W 初期値 0x00000000



- E[d31]：表示キャプチャイネーブルフラグ
1 をセットすると、次の 0 ライン目から 1 画面分のデータをキャプチャした後、0 がセットされます。

0	ディセーブル
1	イネーブル

- MOD[d30~d29]：キャプチャモード

00	キャプチャ元 A のデータをキャプチャ
01	キャプチャ元 B のデータをキャプチャ
10	キャプチャ元 A と B のデータをブレンディングしてキャプチャ
11	

- COFS[d27~d26]：キャプチャデータ元 VRAM の読み込みアドレスオフセット
VRAM 表示モード時は無効となります。
読み込み中にオフセットが 0x20000 を超えた場合は 0x00000 番地に回り込んで読み込みます。

00	0x00000
01	0x08000
10	0x10000
11	0x18000

- SRC[d25~d24]：キャプチャデータ元選択

B	
0	VRAM
1	メインメモリ

A	
0	グラフィックス表示画面（3D／2D ブレンディング後）
1	3D 面

- WSIZE[d21~d20]：キャプチャサイズ
キャプチャデータを書き込む際のサイズ指定です。ただし、RAM 側のキャプチャ時は常に 1 ラインが 256 ドットのイメージとして読み込まれるため、128×128 ドットの設定ではブレンディングしてキャプチャ（キャプチャモード参照）することはできません。

00	128×128 ドット（0x08000 バイト）
01	256× 64 ドット（0x08000 バイト）
10	256×128 ドット（0x10000 バイト）
11	256×192 ドット（0x18000 バイト）

- WOFS[d19~d18]：キャプチャデータ書き込みアドレスオフセット

指定された VRAM に書き込むアドレスのオフセット値を指定できます。書き込み中にオフセットが 0x20000 を超えた場合は 0x00000 番地に回り込んで書き込みます。

00	0x00000
01	0x08000
10	0x10000
11	0x18000

- DEST[d17~d16]：キャプチャデータ書き込み先 VRAM 選択

書き込み先の VRAM は LCD C に割り当てておく必要があります。

00	VRAM-A
01	VRAM-B
10	VRAM-C
11	VRAM-D

- EVB[d12~d08], EVA[d04~d00]：ブレンディング係数
キャプチャ元 A, B のブレンディング係数を設定します。後述の計算式を参照してください。

VRAM 表示モード時には、表示する VRAM とキャプチャ画像を書き込む VRAM を同じ VRAM に設定しても構いません。

キャプチャデータ・フォーマット

15	14	10	9	8	7	5	4	0
A	BLUE			GREEN			RED	
α	ピクセル色データ							

3D グラフィックスは (R:G:B = 6:6:6) カラーで出力されますが、キャプチャは (R:G:B = 5:5:5) カラー（上位 5 ビットの採用）で行われるため、画像の階調が若干粗くなります。

図 5-9 にキャプチャサイズ 256×192 ドット時のキャプチャデータの LCD ピクセルマップを示します。

	ドット 0	1	2	3		253	254	255
ライン 0	0h	2h	4h	6h		1FAh	1FCh	1FEh
1	200h	202h	204h	206h		3FAh	3FCh	3FEh
2	400h	402h					5FCh	5FEh
3	600h	602h					7FCh	7FEh
4	800h							9FEh
187	17600h							177FEh
188	17800h	17802h					179FCh	179FEh
189	17A00h	17A02h					17BFCh	17BFEh
190	17C00h	17C02h	17C04h	17C06h		17DFAh	17DFCh	17DFEh
191	17E00h	17E02h	17E04h	17E06h		17FFAh	17FFCh	17FFEh

図 5-9 キャプチャデータの LCD ピクセルマップ（キャプチャサイズ 256×192 ドット時）

●書き込まれるデータの計算式

1) 「キャプチャ元 A のデータをキャプチャ」する場合

$$CAP = Ca$$

α 値はキャプチャ元 A の α 値が使われます。

2) 「キャプチャ元 B のデータをキャプチャ」する場合

$$CAP = Cb$$

α 値はキャプチャ元 B の α 値が使われます。

3) 「キャプチャ元 A と B のデータをブレンディングしてキャプチャ」する場合

$$CAP = \frac{(Ca \times Aa \times EVA) + (Cb \times Ab \times EVB)}{16}$$

α 値については、「EVA が 0 以外かつ A のキャプチャ元データの α 値が 1 の場合」もしくは、「EVB が 0 以外かつ B のキャプチャ元データの α 値が 1 の場合」に 1 が書き込まれます。それ以外の場合は 0 が書き込まれます。

CAP：書き込みカラー（計算結果の小数点以下は四捨五入されます）

Ca：A のキャプチャ元データカラー、EVA：A のブレンディング係数

Cb：B のキャプチャ元データカラー、EVB：B のブレンディング係数

Aa：A の α 値：A のキャプチャ元 α 値。下記の通りに決定されます。

キャプチャ元 A の選択	3D 面の α 値	Aa
0	—	1
1	0	0
	1～31	1

Ab：B の α 値：B のキャプチャ元 α 値

注意事項

表示回路の VRAM へのアクセスと CPU からの VRAM へのアクセスが衝突した場合は、CPU からのアクセスが待たされます。

LCD コントローラのドットクロックが画像処理クロックおよびシステムクロックの 6 分周となっていますので、LCD コントローラが VRAM にアクセスするタイミングは 6 サイクルに 1 回となります。

表示キャプチャでキャプチャをしながらその VRAM を表示させている場合は、表示回路がその VRAM にアクセスする頻度が 2 倍になり、3 サイクルに 1 回のタイミングで VRAM にアクセスすることになります。

このタイミングで、CPU から同時にアクセスした場合は CPU のアクセスが 1 サイクル待たされることになります。

5.6 マスター輝度

画像出力 A および画像出力 B の輝度アップ/ダウン処理を、画像出力 A は MASTER_BRIGHT レジスタ、画像出力 B は DB_MASTER_BRIGHT レジスタの設定で処理することができます。

マスター輝度アップ/ダウンレジスタ

名称 MASTER_BRIGHT アドレス 0x0400006C 属性 R/W 初期値 0x0000

15	14									4	0
E_MOD										E_VALUE	
モード										係数	

マスター輝度アップ/ダウン B レジスタ

名称 DB_MASTER_BRIGHT アドレス 0x0400106C 属性 R/W 初期値 0x0000

15	14									4	0
E_MOD										E_VALUE	
モード										係数	

レジスタの設定内容は MASTER_BRIGHT レジスタ、DB_MASTER_BRIGHT レジスタともに共通です。

- E_MOD [d15～d14]：モード
輝度アップ/ダウン処理のモードを設定します。

設定	処理
00	輝度変更なし
01	輝度アップ
10	輝度ダウン
11	設定禁止

- E_VALUE [d04～d00]：係数
下記計算式の係数を設定します。

1) 輝度アップの計算式

$$\begin{aligned} \text{Rout} &= \text{Rin} + (63 - \text{Rin}) \times (\text{E_VALUE} / 16) \\ \text{Gout} &= \text{Gin} + (63 - \text{Gin}) \times (\text{E_VALUE} / 16) \\ \text{Bout} &= \text{Bin} + (63 - \text{Bin}) \times (\text{E_VALUE} / 16) \end{aligned}$$

2) 輝度ダウンの計算式

$$\begin{aligned} \text{Rout} &= \text{Rin} - \text{Rin} \times (\text{E_VALUE} / 16) \\ \text{Gout} &= \text{Gin} - \text{Gin} \times (\text{E_VALUE} / 16) \\ \text{Bout} &= \text{Bin} - \text{Bin} \times (\text{E_VALUE} / 16) \end{aligned}$$

なお、輝度アップおよび輝度ダウンにおける計算結果（Rout, Gout, Bout）の小数点以下は四捨五入されます。

6. 2D グラフィックス

TWL の 2D グラフィックスエンジンは、BG ウィンドウ機能の回路修正（バグフィックス）がなされた以外には NITRO から変更はありません。変更の詳細については「ウィンドウ」を参照してください。

TWL の 2D グラフィックスエンジンは A と B の二つが用意されています。

2D グラフィックスエンジン A では、BG タイプに 3D グラフィックス BG と大画面 256 色ビットマップ BG を使用することができますが、2D グラフィックスエンジン B では使用することができません。

以降、**2D グラフィックスエンジン A** は「**2D_A**」と、**2D グラフィックスエンジン B** は「**2D_B**」と略すことがあります。**2D グラフィックスエンジン A** と **2D グラフィックスエンジン B** において、レジスタ名称が異なる場合は、**[]** 内に **2D グラフィックスエンジン B** でのレジスタ名称を記載しています。

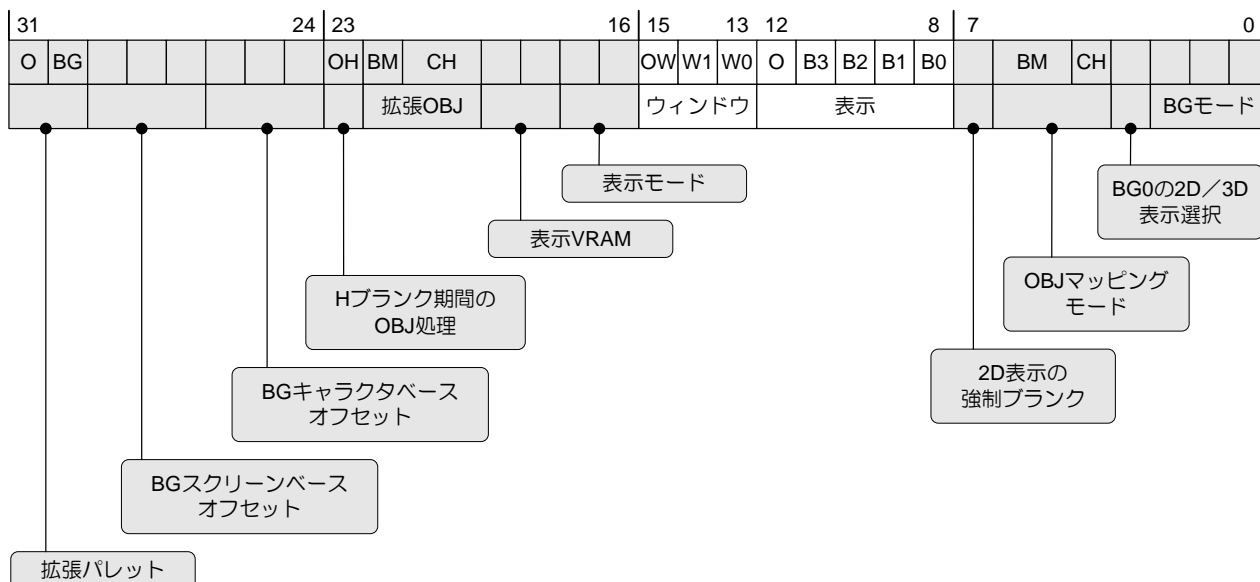
6.1 2D 表示制御

2D グラフィックスの各機能について、表示の ON/OFF を個別に制御することができます。

2D グラフィックスエンジン A と 2D グラフィックスエンジン B では制御に使用するレジスタの設定内容が異なります。

表示コントロールレジスタ (2D グラフィックスエンジン A)

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



- [d15~d13]: ウィンドウ表示イネーブルフラグ
ウィンドウ機能については、「ウィンドウ」章を参照してください。

- OW[d15]: OBJ ウィンドウ表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

OBJ ウィンドウを表示するには、OBJ ウィンドウ表示イネーブルフラグだけでなく OBJ 表示イネーブルフラグもイネーブルにする必要があります。

- W1[d14]: ウィンドウ 1 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- W0[d13]: ウィンドウ 0 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- [d12~d08]: 表示選択フラグ

- O[d12]: OBJ 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- B3[d11]: BG3 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- B2[d10]: BG2 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- B1[d09]: BG1 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- B0[d08]: BG0 表示イネーブルフラグ

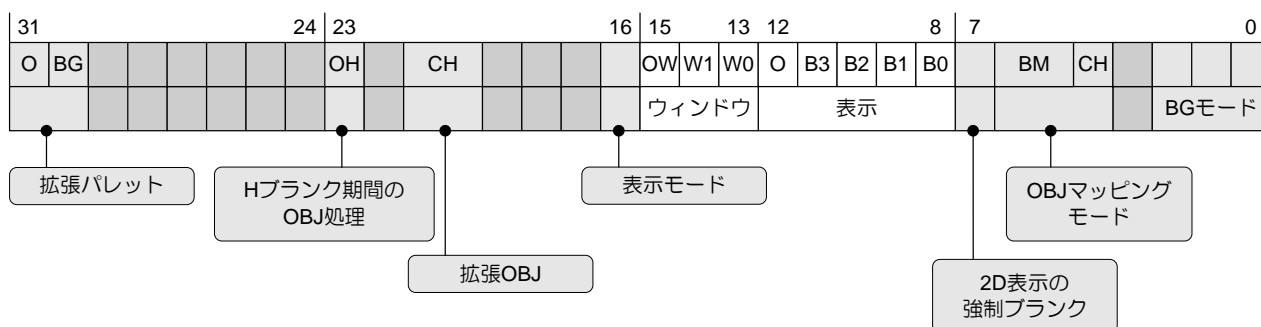
0	表示ディセーブル
1	表示イネーブル

注意事項

[d12~d08]の表示選択フラグは、1→0にした場合は設定直後に、0→1にした場合は3ライン後の先頭から表示イネーブル/ディセーブルが切り換わります。

表示コントロールレジスタ1 (2D グラフィックスエンジン B)

名称	DB_DISPCNT	アドレス	0x04001000	属性	R/W	初期値	0x00000000
----	------------	------	------------	----	-----	-----	------------



- [d15～d13]: ウィンドウ表示イネーブルフラグ
ウィンドウ機能については、「ウィンドウ」章を参照してください。

- OW[d15] : OBJ ウィンドウ表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

OBJ ウィンドウを表示するには、OBJ ウィンドウ表示イネーブルフラグだけでなく OBJ 表示イネーブルフラグもイネーブルにする必要があります。

- W1[d14]：ウィンドウ 1 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- W0[d13] : ウィンドウ 0 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- [d12～d08]：表示選択フラグ^a

- O[d12]:OBJ 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- B3[d11] : BG3 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- B2[d10] : BG2 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- B1[d09] : BG1 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

- B0[d08] : BG0 表示イネーブルフラグ

0	表示ディセーブル
1	表示イネーブル

注意事項

[d12～d08]の表示選択フラグは、1→0にした場合は設定直後に、0→1にした場合は3ライン後の先頭から表示イネーブ
ル／ディセーブルが切り換わります。

6.2 BG

6.2.1 BG モード

2D グラフィックスエンジン A と 2D グラフィックスエンジン B では設定可能な BG モードが異なります。

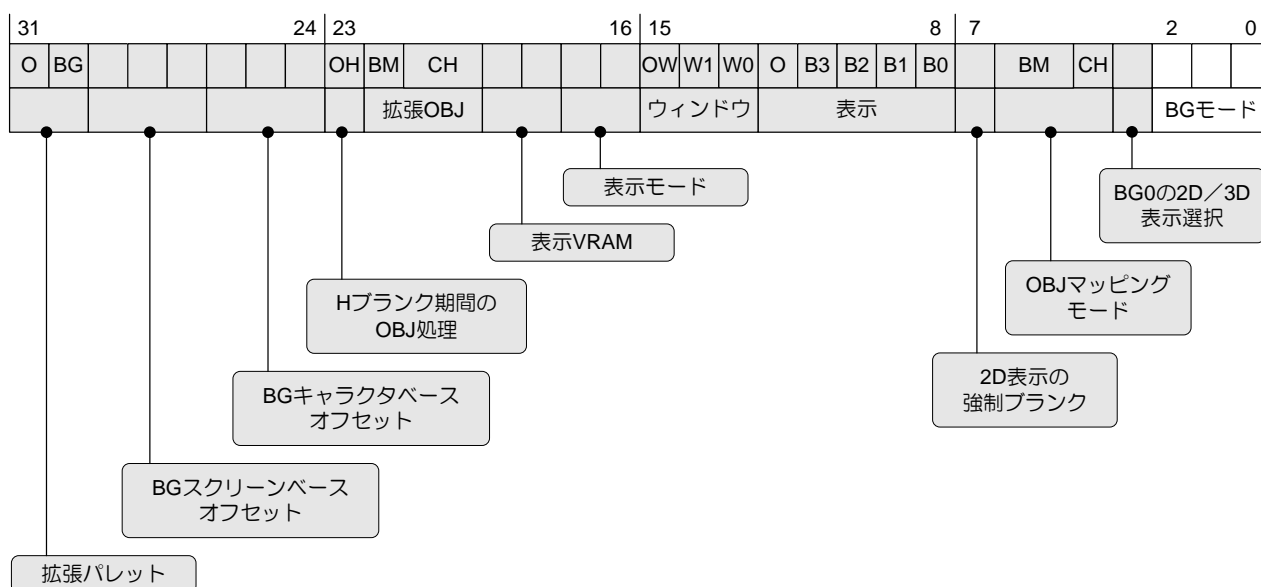
6.2.1.1 2D グラフィックスエンジン A

2D グラフィックスエンジン A では、BG0 の表示を 2D または 3D から選択することができます。

また、BG2 の BG タイプに大画面 256 色ビットマップ BG を選択することができます。

表示コントロールレジスタ (2D グラフィックスエンジン A)

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



●[d02~d00] : BG モード

BG モード番号を設定します。

BG モードにより、使用できる BG のタイプを選択することができます。

2D グラフィックスエンジン A の BG モード一覧については表 6-1 を参照してください。

BG0 については、DISPCNT レジスタによって BG タイプをテキスト BG または 3DBG から選択することができます。3DBG の表示については、「3D グラフィックス」の章を参照してください。

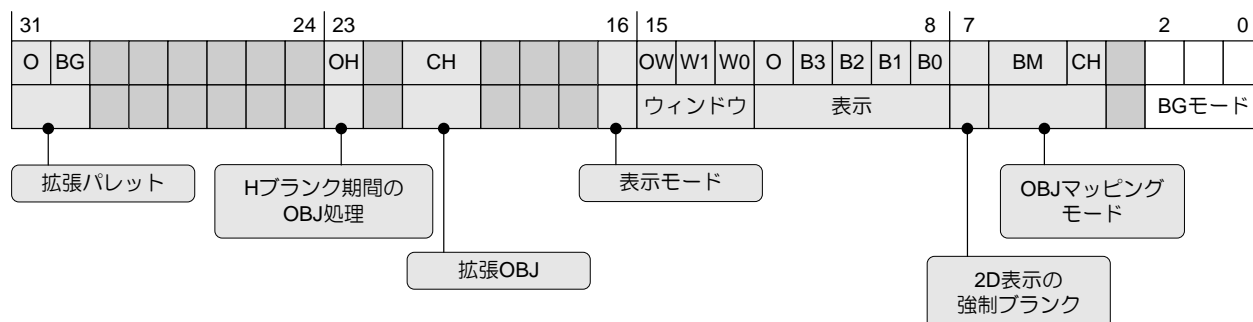
BG モード 番号	BG0	BG1	BG2	BG3
0	テキスト BG/3DBG	テキスト BG	テキスト BG	テキスト BG
1	テキスト BG/3DBG	テキスト BG	テキスト BG	アフィン BG
2	テキスト BG/3DBG	テキスト BG	アフィン BG	アフィン BG
3	テキスト BG/3DBG	テキスト BG	テキスト BG	アフィン拡張 BG
4	テキスト BG/3DBG	テキスト BG	アフィン BG	アフィン拡張 BG
5	テキスト BG/3DBG	テキスト BG	アフィン拡張 BG	アフィン拡張 BG
6	3DBG	—	大画面 256 色 ビットマップ BG	—
7	設定禁止			

表 6-1 BG モード一覧 (2D グラフィックスエンジン A)

6.2.1.2 2D グラフィックスエンジン B

表示コントロールレジスタ 1 (2D グラフィックスエンジン B)

名称 DB_DISPCNT アドレス 0x04001000 属性 R/W 初期値 0x00000000



●[d02～d00] : BG モード

BG モード番号を設定します。

BG モードにより、使用できる BG のタイプを選択できます。

2D グラフィックスエンジン B の BG モード一覧については表 6-2 を参照してください。

注意事項

2D グラフィックスエンジン A とは異なり、BG2 の BG タイプに大画面 256 色ビットマップ BG は設定することができません。

また、BG0 の BG タイプに 3DBG 表示を設定することができません。

BG モード番号	BG0	BG1	BG2	BG3
0	テキスト BG	テキスト BG	テキスト BG	テキスト BG
1	テキスト BG	テキスト BG	テキスト BG	アフィン BG
2	テキスト BG	テキスト BG	アフィン BG	アフィン BG
3	テキスト BG	テキスト BG	テキスト BG	アフィン拡張 BG
4	テキスト BG	テキスト BG	アフィン BG	アフィン拡張 BG
5	テキスト BG	テキスト BG	アフィン拡張 BG	アフィン拡張 BG
6	設定禁止			
7	設定禁止			

表 6-2 BG モード一覧 (2D グラフィックスエンジン B)

6.2.1.3 BG タイプ別の基本的な特徴

BG タイプ別の特徴は表 6-3 の通りです。

BG タイプ	特徴
3DBG	3D グラフィックスエンジンによって生成された画像を表示できます。 他の BG 面とは、 α ブレンディングや優先順位に応じた表示等が可能です。 2D グラフィックスエンジン B では使用することができません。
テキスト BG	キャラクタ方式の BG です。 唯一 16 色で定義されたキャラクタを扱うことができ VRAM の消費を抑えることができますが、回転拡大縮小は行えません。
アフィン BG	回転拡大縮小を行うことができるキャラクタ方式の BG です。 キャラクタ単位の処理 (HV フリップ) を行うことはできません。
アフィン拡張 BG	下記の 3 種類の中から選択できます。 <ul style="list-style-type: none">• 256 色×16 パレットを使用できるキャラクタ BG• 256 色ビットマップ BG• 直接カラーを指定できるダイレクトカラービットマップ BG
大画面 256 色 ビットマップ BG	大画面のビットマップ BG です。 1 面で BG-VRAM の最大容量 (512K バイト) を使い切るため、他の BG と併用することはできませんが、3D 面との併用は可能です。 2D グラフィックスエンジン B では使用することができません。

表 6-3 BG タイプ別の基本的な特徴

6.2.1.4 BG タイプ別仕様一覧

BG タイプ別の仕様は表 6-4 の通りです。

カテゴリ	キャラクタ BG				ビットマップ BG		
BG タイプ 仕様	3DBG	テキスト BG	アフィン BG	アフィン拡張 BG			大画面 256 色ビット マップ BG
				256 色×16 パレット	256 色	ダイレクト カラー	
スクリーン サイズ	256×192	256×256 512×256 256×512 512×512	128×128 256×256 512×512 1,024×1,024	128×128 256×256 512×512 1,024×1,024	128×128 256×256 512×256 512×512	128×128 256×256 512×256 512×512	512×1,024 1,024×512
指定可能 キャラクタ数	—	1,024	256	1,024	—	—	—
色数/パレット	262,144	16/16 256/1 256/16	256/1	256/16	256/1	32,768	256/1
回転拡大縮小			○	○	○	○	○
HV フリップ		○		○			
H スクロール	○	○	○	○	○	○	○
V スクロール		○	○	○	○	○	○
モザイク		○	○	○	○	○	○
フェード イン/アウト	○	○	○	○	○	○	○
αブレンディング	○	○	○	○	○	○	○
優先順位	○	○	○	○	○	○	○

表 6-4 BG タイプ別の仕様一覧

注意事項

2D グラフィックスエンジン B では BG タイプに「3DBG」と「大画面 256 色ビットマップ BG」は設定できません。

2D グラフィックスエンジン B への BG-VRAM 割り当ての制限のため、以下の設定が使用できません。

- ・「256 色ビットマップ」 サイズ 512×512
- ・「ダイレクトカラービットマップ」 サイズ 512×256、512×512

6.2.2 BG コントロール

BG コントロールレジスタは BG の面数に対応して 4 つ存在します。

2D グラフィックスエンジン A では BG0CNT, BG1CNT, BG2CNT, BG3CNT の各レジスタで制御します。

2D グラフィックスエンジン B では DB_BG0CNT, DB_BG1CNT, DB_BG2CNT, DB_BG3CNT の各レジスタで制御します。

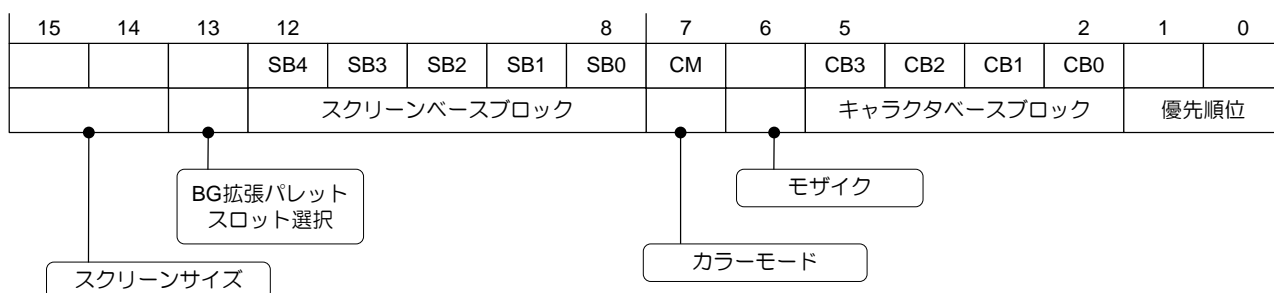
注意事項

2D グラフィックスエンジン A と 2D グラフィックスエンジン B では、レジスタの名称と BG スクリーンデータベースアドレス値の算出方法、BG キャラクタデータベースアドレス値の算出方法が異なります。

BGx (x=0, 1) コントロールレジスタ

(2D_A) 名称 BGxCNT (x=0, 1) アドレス 0x04000008, 0x0400000A 属性 R/W 初期値 0x0000

(2D_B) 名称 DB_BGxCNT (x=0, 1) アドレス 0x04001008, 0x0400100A 属性 R/W 初期値 0x0000



●[d15～d14]：スクリーンサイズ

スクリーンサイズ 設定値	テキスト BG	
	スクリーンサイズ	スクリーンデータサイズ
00	256×256	2K バイト
01	512×256	4K バイト
10	256×512	4K バイト
11	512×512	8K バイト

●[d13]：BG 拡張パレットスロット選択

BG 拡張パレットがイネーブルのときに、使用する拡張パレットのスロット No.を指定します。

BG0 と BG1 で設定が異なります。

拡張パレットのイネーブル／ディセーブルは DISPCNT [DB_DISPCNT] レジスタで設定できます。

パレットスロットのメモリマッピングについては、「VRAM」を参照してください。

1) BG0CNT [DB_BG0CNT]

0	スロット 0
1	スロット 2

2) BG1CNT [DB_BG1CNT]

0	スロット 1
1	スロット 3

●SB4～SB0 [d12～d08]：スクリーンベースブロック

スクリーンデータが格納されている VRAM 上の先頭ブロックを指定します。（2K バイト単位）

実際にスクリーンデータを参照する際の先頭アドレスは以下の算出値となります。

2D グラフィックスエンジン A

DISPCNT レジスタのスクリーンベースオフセット値を 64K バイト単位でオフセットした値に、スクリーンベースブロックを 2K バイト単位でオフセットした値を加算した値が先頭アドレスとなります。

$$(\text{スクリーンベースオフセット} \times 0x10000) + (\text{スクリーンベースブロック} \times 0x800)$$

2D グラフィックスエンジン B

スクリーンベースブロックを 2K バイト単位でオフセットした値が先頭アドレスとなります。

$$(\text{スクリーンベースブロック} \times 0x800)$$

●CM[d07]：カラーモード

スクリーンデータによって参照される BG キャラクタデータを、16 色か 256 色どちらのフォーマットで参照するかを指定します。

0	16 色モード
1	256 色モード

●[d06]：モザイク

BG に対するモザイク処理の ON / OFF を制御します。

モザイクのサイズは MOSAIC [DB_MOSAIC] レジスタで設定してください。

●CB3～CB0 [d05～d02]：キャラクタベースブロック

キャラクタデータの格納されている VRAM 上の先頭ブロックを指定します（16K バイト単位）。

実際にキャラクタデータを参照する際の先頭アドレスは以下の算出値となります。

2D グラフィックスエンジン A

DISPCNT レジスタのキャラクタベースオフセット値を 64K バイト単位でオフセットした値に、キャラクタデータベースブロック値を 16K バイト単位でオフセットした値を加算した値が先頭アドレスとなります。

$$(\text{キャラクタベースオフセット} \times 0x10000) + (\text{キャラクタベースブロック} \times 0x4000)$$

2D グラフィックスエンジン B

キャラクタベースブロックを 16K バイト単位でオフセットした値が先頭アドレスとなります。

$$(\text{キャラクタベースブロック} \times 0x4000)$$

●[d01～d00]：優先順位

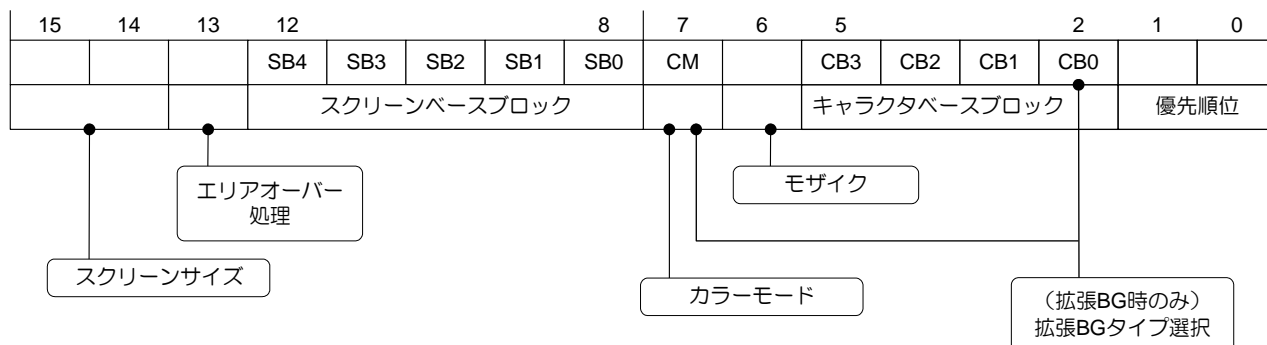
デフォルト（優先順位指定値が同じ値）の場合は、優先順が BG0>BG1>BG2>BG3 となっていますが、これを任意に組み換えることができます。0（最優先）～3 の指定となります。

BG の優先順位を変更したときは、カラー特殊効果の対象ピクセル指定に注意してください。

BGx (x=2, 3) コントロールレジスタ

(2D_A) 名称 BGxCNT (x=2, 3) アドレス 0x0400000C, 0x0400000E 属性 R/W 初期値 0x0000

(2D_B) 名称 DB_BGxCNT (x=2, 3) アドレス 0x0400100C, 0x0400100E 属性 R/W 初期値 0x0000



カラーモード、モザイク、キャラクタベースブロック、優先順位のビットの詳細は前述の BGx (x=0, 1) コントロールレジスタと同様です。

拡張パレットがイネーブルのとき、BG2 は拡張パレットスロット 2 を使用し、BG3 は拡張パレットスロット 3 を使用します。BG2 および 3 において、使用する拡張パレット No.を変更することはできません。

拡張パレットのイネーブル/ディセーブルは DISPCNT [DB_DISPCNT] レジスタで設定できます。

●[d15～d14]：スクリーンサイズ

BG タイプによって設定可能なサイズが異なり、表 6-5 および表 6-6 の通りになります。

注意事項

2D グラフィックスエンジン A と 2D グラフィックスエンジン B では設定可能な BG タイプとスクリーンサイズの組み合わせが異なります。

2D グラフィックスエンジン B は BG タイプに大画面 256 色ビットマップ BG を設定できません。

2D グラフィックスエンジン B へ割り当てられる BG-VRAM は最大 128K バイトのため、128K バイトを超えるサイズは設定禁止となります。

スクリーン サイズ 設定値	テキスト BG	アフィン BG	アフィン拡張 BG			大画面 256 色 ビットマップ BG
			256 色×16 パレット キャラクタ BG	256 色 ビットマップ BG	ダイレクトカラー ビットマップ BG	
00	256×256 (2K バイト)	128×128 (256 バイト)	128×128 (512 バイト)	128×128 <16K バイト>	128×128 <32K バイト>	512×1,024 <512K バイト>
01	512×256 (4K バイト)	256×256 (1K バイト)	256×256 (2K バイト)	256×256 <64K バイト>	256×256 <128K バイト>	1,024×512 <512K バイト>
10	256×512 (4K バイト)	512×512 (4K バイト)	512×512 (8K バイト)	512×256 <128K バイト>	512×256 <256K バイト>	—
11	512×512 (8K バイト)	1,024×1,024 (16K バイト)	1,024×1,024 (32K バイト)	512×512 <256K バイト>	512×512 <512K バイト>	—

表 6-5 スクリーンサイズ (2D グラフィックスエンジン A)

スクリーン サイズ 設定値	テキスト BG	アフィン BG	アフィン拡張 BG		
			256 色×16 パレット キャラクタ BG	256 色 ビットマップ BG	ダイレクトカラー ビットマップ BG
00	256×256 (2K バイト)	128×128 (256 バイト)	128×128 (512 バイト)	128×128 <16K バイト>	128×128 <32K バイト>
01	512×256 (4K バイト)	256×256 (1K バイト)	256×256 (2K バイト)	256×256 <64K バイト>	256×256 <128K バイト>
10	256×512 (4K バイト)	512×512 (4K バイト)	512×512 (8K バイト)	512×256 <128K バイト>	設定禁止
11	512×512 (8K バイト)	1,024×1,024 (16K バイト)	1,024×1,024 (32K バイト)	設定禁止	設定禁止

表 6-6 スクリーンサイズ (2D グラフィックスエンジン B)

() はスクリーンデータサイズ
<> はビットマップデータサイズ

●[d13]：エリアオーバー処理

アフィン変換によって、表示画面がスクリーンを越えた場合、その領域を透明とするか、回り込んで表示するかを選択することができます。

0	透明表示
1	回り込んで表示

エリアオーバー処理の違いを図 6-1 に示します。

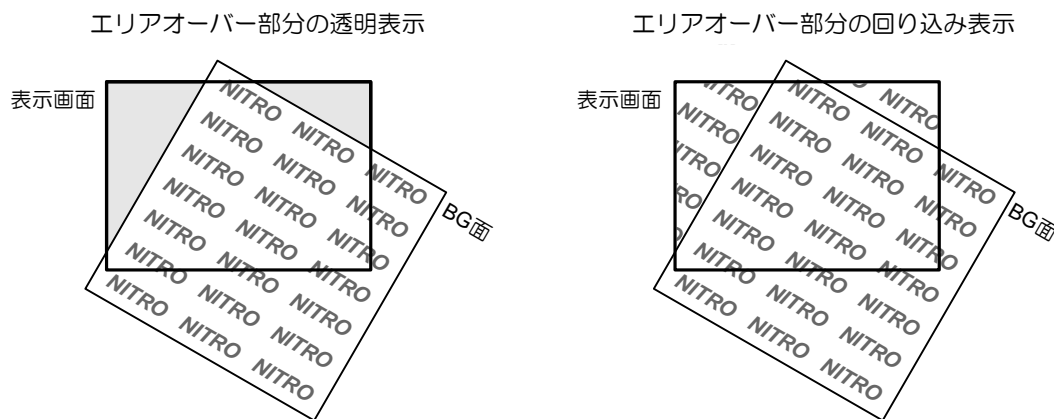


図 6-1 エリアオーバー処理の違い

●SB4～SB0 [d12～d08]：スクリーンベースブロック

BG モードによってスクリーンベースブロックのベースアドレス値の算出方法が異なります。

1) キャラクタ BG のとき

スクリーンデータの格納されている VRAM 上の先頭ブロックを指定します。（2K バイト単位）
実際にスクリーンデータを参照する際の実アドレスは以下の算出値となります。

2D グラフィックスエンジン A

DISPCNT レジスタのスクリーンベースオフセット値を 64K バイト単位でオフセットした値に、スクリーンベースブロックを 2K バイト単位でオフセットした値を加算した値が先頭アドレスとなります。

$$(\text{スクリーンベースオフセット} \times 0x10000) + (\text{スクリーンベースブロック} \times 0x800)$$

2D グラフィックスエンジン B

スクリーンベースブロックを 2K バイト単位でオフセットした値が先頭アドレスとなります。

$$(\text{スクリーンベースブロック} \times 0x800)$$

2) 256 色ビットマップ BG およびダイレクトカラービットマップ BG のとき

ビットマップデータの格納されている BG-VRAM 上のオフセットアドレスを指定します（16K バイト単位）。
DISPCNT レジスタのスクリーンベースオフセット値とは無関係になりますので、2D グラフィックスエンジン A、2D グラフィックスエンジン B とともに BG ビットマップデータの先頭アドレスは同じ算出方法となります。

2D グラフィックスエンジン A、2D グラフィックスエンジン B

$$(\text{スクリーンベースブロック} \times 0x4000)$$

3) 大画面 256 色ビットマップ BG のとき

2D グラフィックスエンジン A ではスクリーンベースブロックの値は無効となります。

2D グラフィックスエンジン B では大画面 256 色ビットマップ BG に設定することができません。

- [d07, d02] : アフィン拡張 BG タイプ選択 (アフィン拡張 BG 時のみ)

CM	CB0	アフィン拡張 BG タイプ
0	※	256 色×16 パレットキャラクタ BG
1	0	256 色ビットマップ BG
1	1	ダイレクトカラービットマップ BG

※CM=0 の場合は一意に 256 色×16 パレットキャラクタ BG となり、CB3~CB0 は通常通りキャラクタベースブロックとして扱われます。

6.2.2.1 スクリーンサイズと表示画面

6.2.2.1.1 テキスト BG

テキスト BG 画面でのスクリーンサイズは図 6-2 の通りです。

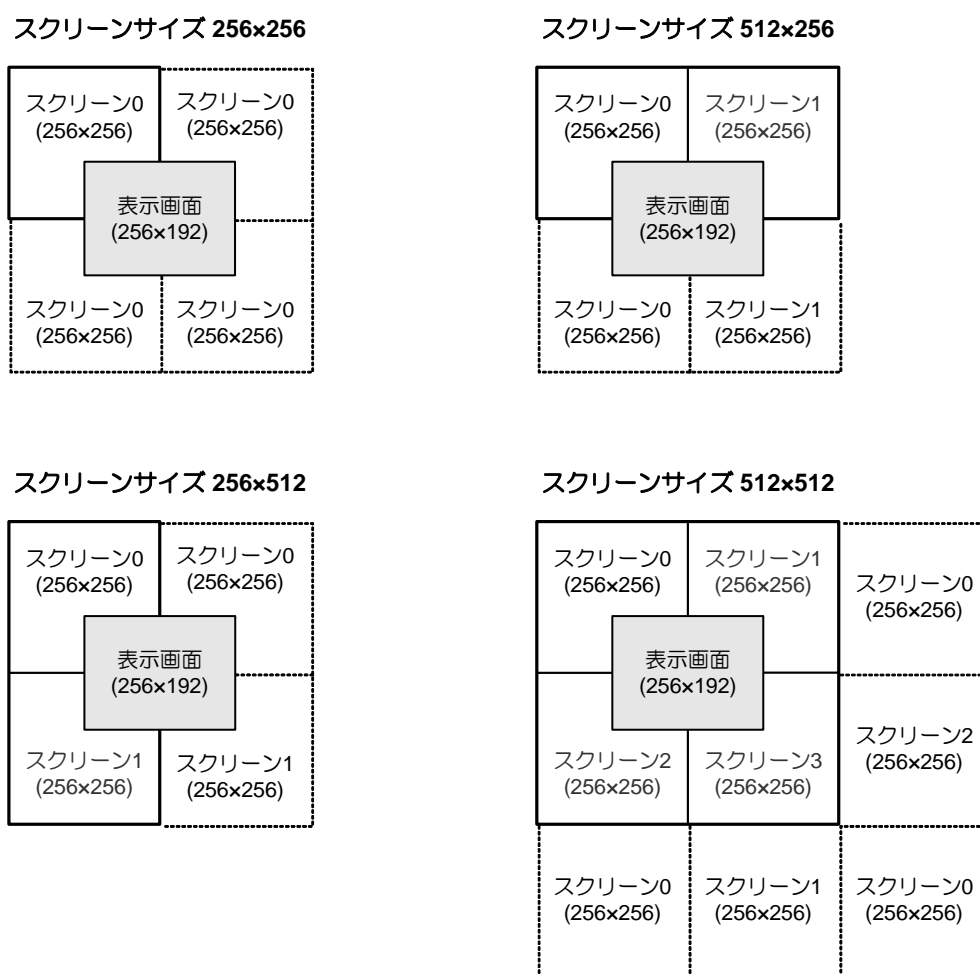


図 6-2 テキスト BG 画面のスクリーンサイズ概念図

6.2.2.1.2 アフィン BG

アフィン BG 画面でのスクリーンサイズは図 6-3 の通りです。

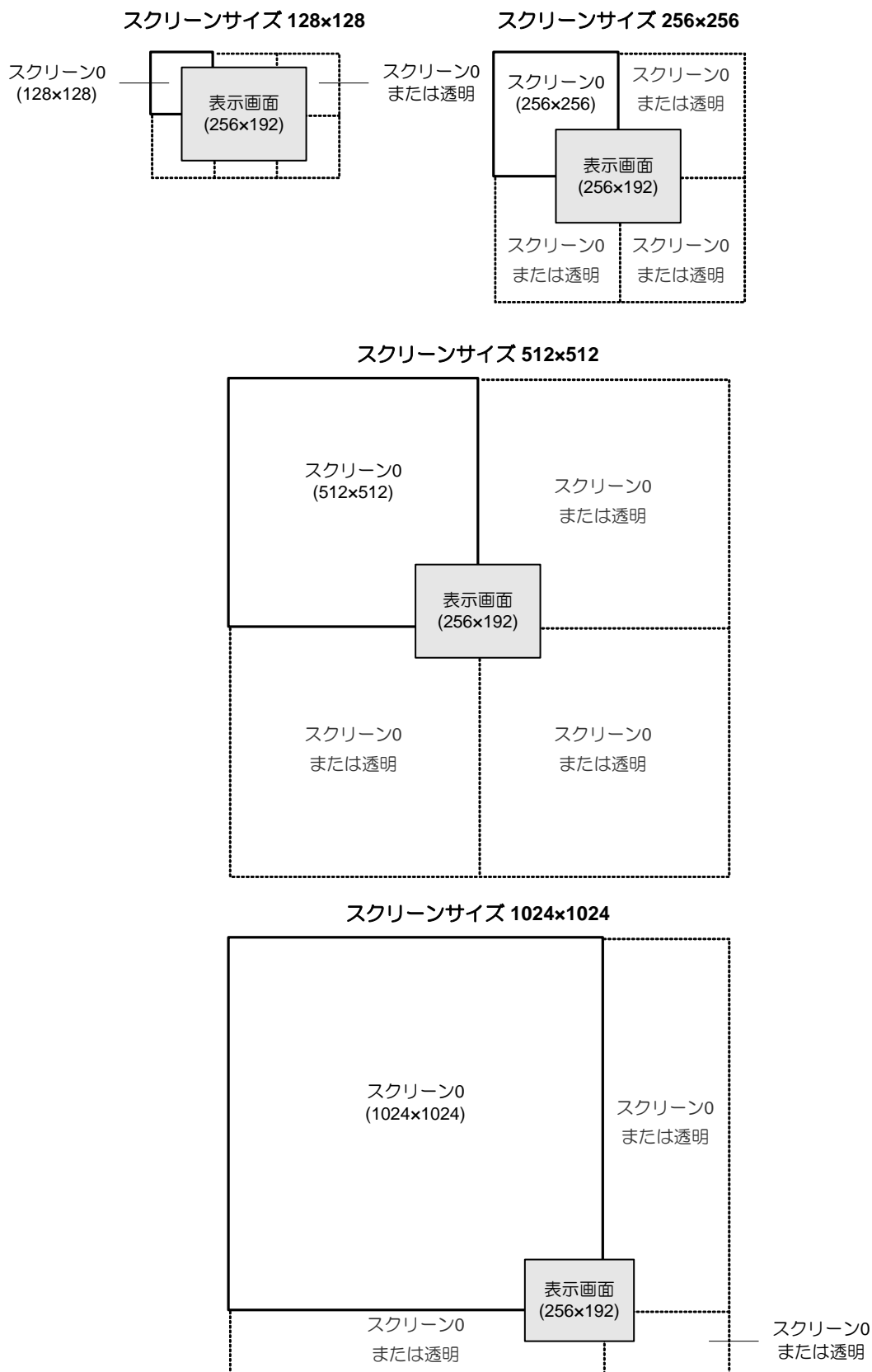


図 6-3 アフィン BG 画面のスクリーンサイズ概念図

6.2.3 キャラクタ BG

キャラクタ BG では、BG 画面の構成要素を 8×8 ドットの基本キャラクタとして扱います。

このため BG を表示するために、キャラクタデータが必要です。

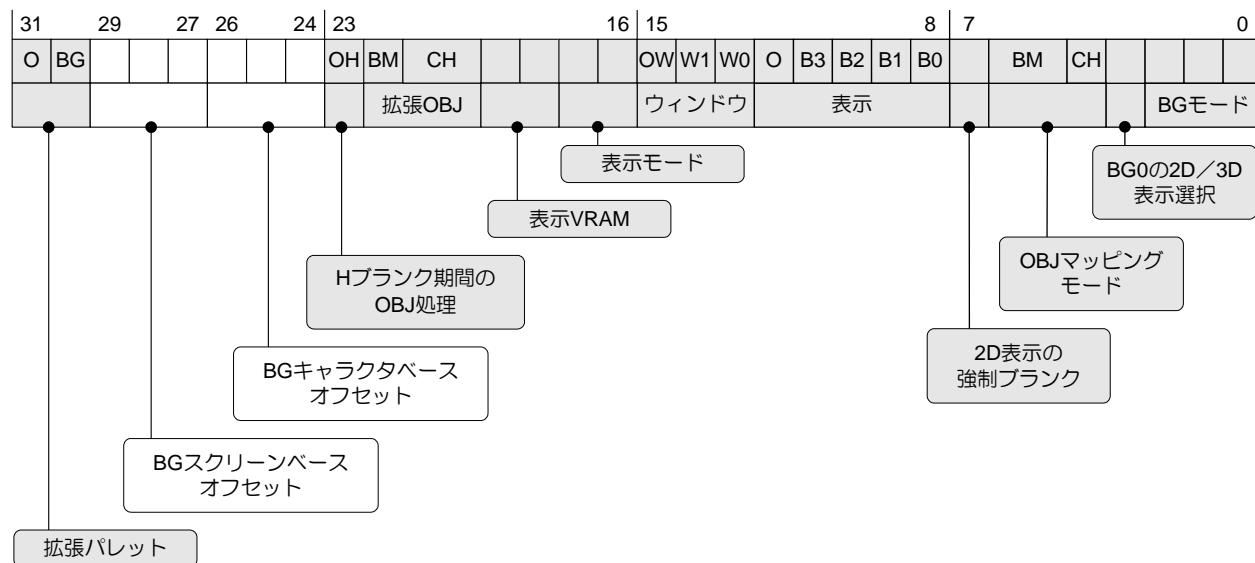
また、8×8 ドット毎にキャラクタのインデックスデータが必要となり、これをスクリーンデータと呼びます。

注意事項

2D グラフィックスエンジン B では 2D グラフィックスエンジン A とは異なり、BG スクリーンベースオフセットと BG キャラクタベースオフセットの設定がありません。

表示コントロールレジスタ

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



●[d29～d27]：BG スクリーンベースオフセット

BG コントロールレジスタで設定されたスクリーンデータベースアドレスを 64K バイト単位でオフセットします。よって、BG スクリーンデータベースアドレス値は

BG コントロールレジスタで設定した値+ (BG スクリーンベースオフセット×0x10000)

となり、最大 512K バイトの BG-VRAM 空間から任意のベースアドレスを指定できます。

●[d26～d24]：BG キャラクタベースオフセット

BG コントロールレジスタで設定されたキャラクタデータベースアドレスを 64K バイト単位でオフセットします。よって、BG キャラクタデータベースアドレス値は

BG コントロールレジスタで設定した値+ (BG キャラクタベースオフセット×0x10000)

となり、最大 512K バイトの BG-VRAM 空間から任意のベースアドレスを指定できます。

6.2.3.1 BG データの VRAM マップ

キャラクタ BG では、BG スクリーンデータと BG キャラクタデータが必要となります。
BG スクリーンデータおよび BG キャラクタデータは、RAM バンクコントロールレジスタによって BG-VRAM に割り当てられた VRAM に格納してください。
BG-VRAM は 2D グラフィックスエンジン A では最大 512K バイト、2D グラフィックスエンジン B では最大 128K バイトを割り当てることができます。

1) BG キャラクタデータ

2D グラフィックスエンジン A では、DISPCNT レジスタのキャラクタベースオフセットおよび BG コントロールレジスタのキャラクタベースブロック指定により、BG キャラクタデータを参照する先頭アドレスを設定できます。
2D グラフィックスエンジン B にはキャラクタベースオフセットの設定がありません。
BG キャラクタデータの VRAM オフセットは図 6-4 の通りです。

データ量は、登録するキャラクタデータの数およびフォーマット（カラーモード：256 色または 16 色）に依存します。

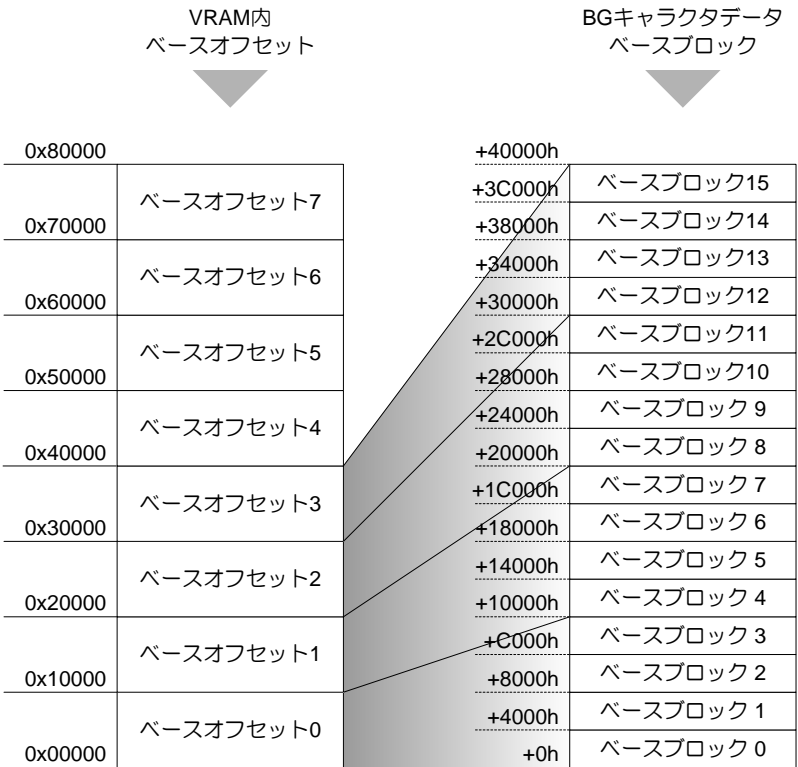


図 6-4 BG キャラクタデータの VRAM オフセット

注意事項

2D グラフィックスエンジン A では、DISPCNT レジスタのベースオフセットは BG 面毎に設定できないため、BG キャラクタデータとして使用できる VRAM 容量は最大 256K バイトとなります。
2D グラフィックスエンジン B では、BG-VRAM の割り当てサイズ制限のため、BG キャラクタデータとして使用できる VRAM 容量は最大 128K バイトとなります。

2) BG スクリーンデータ

2D グラフィックスエンジン A では、DISPCNT レジスタのスクリーンベースオフセットおよび BG コントロールレジスタのスクリーンベースブロック指定により、BG スクリーンデータを参照する先頭アドレスを設定することができます。

2D グラフィックスエンジン B には、スクリーンベースオフセットの設定がありません。

BG スクリーンデータの VRAM オフセットは図 6-5 の通りです。

データ量は、BG タイプ（テキスト BG またはアフィン BG）およびスクリーンサイズに依存します。

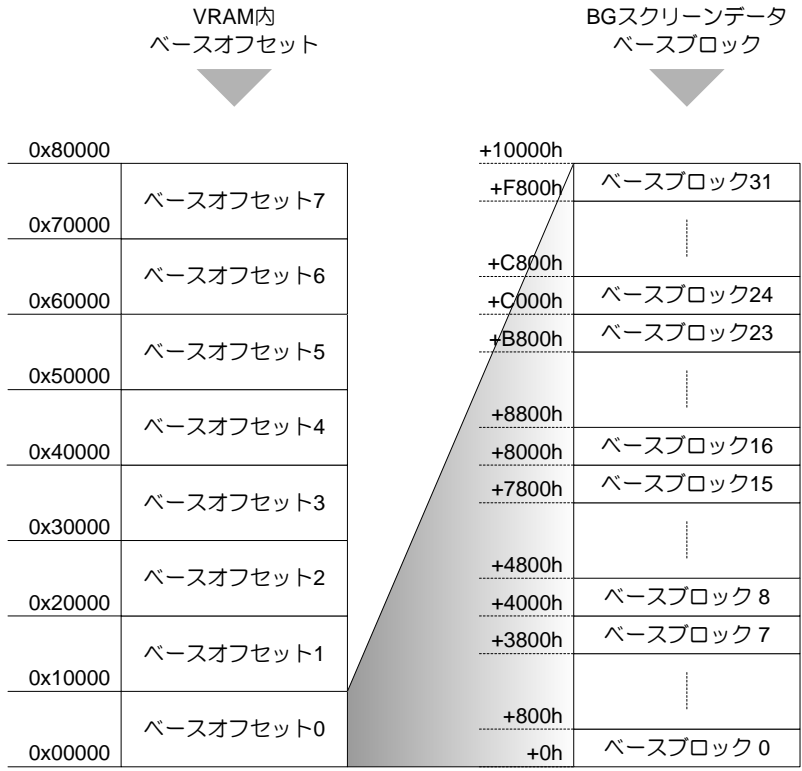


図 6-5 BG スクリーンデータの VRAM オフセット

注意事項

2D グラフィックスエンジン A では、DISPCNT レジスタのベースオフセットは BG 面毎に設定できないため、BG スクリーンデータとして使用できる VRAM 容量は最大 64K バイトとなります。

2D グラフィックスエンジン B では、BG スクリーンデータとして使用できる VRAM 容量は最大 64K バイトとなります。

6.2.3.2 テキスト BG

6.2.3.2.1 スクリーンデータフォーマット

BG スクリーンデータは、BG コントロールレジスタで指定した BG スクリーンベースブロックの先頭アドレスから格納してください。

テキスト BG 画面の BG スクリーンデータは、次を示すフォーマットで設定します。

テキスト BG 画面のスクリーンデータ

15	12			11	10	9	8	7	0					
				VF	HF									
カラーパレット				フリップ		キャラクタネーム								

- [d15～d12]：カラーパレット
キャラクタに適用するパレットを 0～15 の範囲で指定します。
カラーパレットの指定は、256 色×16 パレットのとき、もしくは、16 色×16 パレットのときに有効となり、256 色×1 パレットのときは無効となります。
- [d11～d10]：フリップ
 - VF：垂直反転フラグ、HF：水平反転フラグ

0	フリップしない
1	フリップする
- [d09～d00]：キャラクタネーム
BG コントロールレジスタで指定したキャラクタベースブロックの先頭アドレスを起点としたキャラクタの番号を指定します。

6.2.3.2.2 スクリーンデータのアドレスマッピング

1) スクリーンサイズが 256×256 ドットの時

スクリーンサイズが 256×256 ドットのときのスクリーンデータのアドレスマップは図 6-6 の通りです。

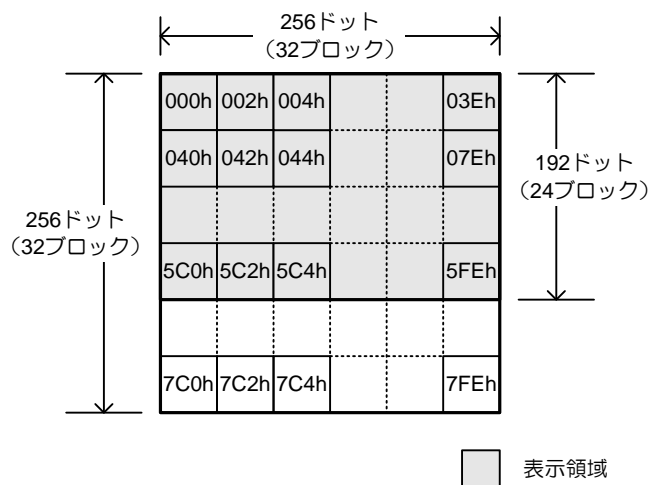


図 6-6 256×256 ドット時のアドレスマッピング (テキスト BG)

2) スクリーンサイズが 256×512 ドットの時

スクリーンサイズが 256×512 ドットのときのスクリーンデータのアドレスマップは図 6-7 の通りです。

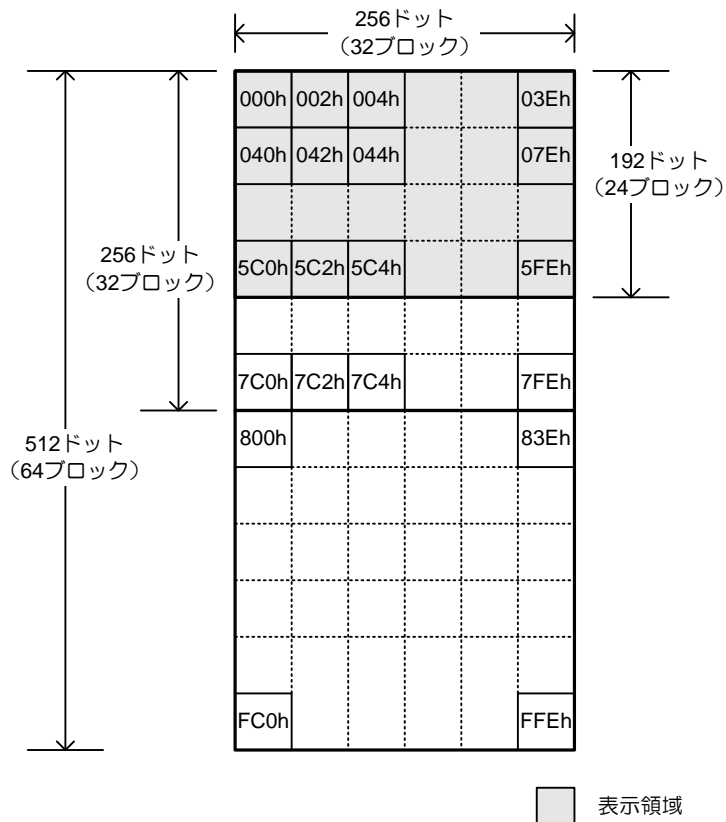


図 6-7 256×512 ドット時のアドレスマッピング (テキスト BG)

3) スクリーンサイズが 512×256 ドットするとき

スクリーンサイズが 512×256 ドットのときのスクリーンデータのアドレスマップは図 6-8 の通りです。

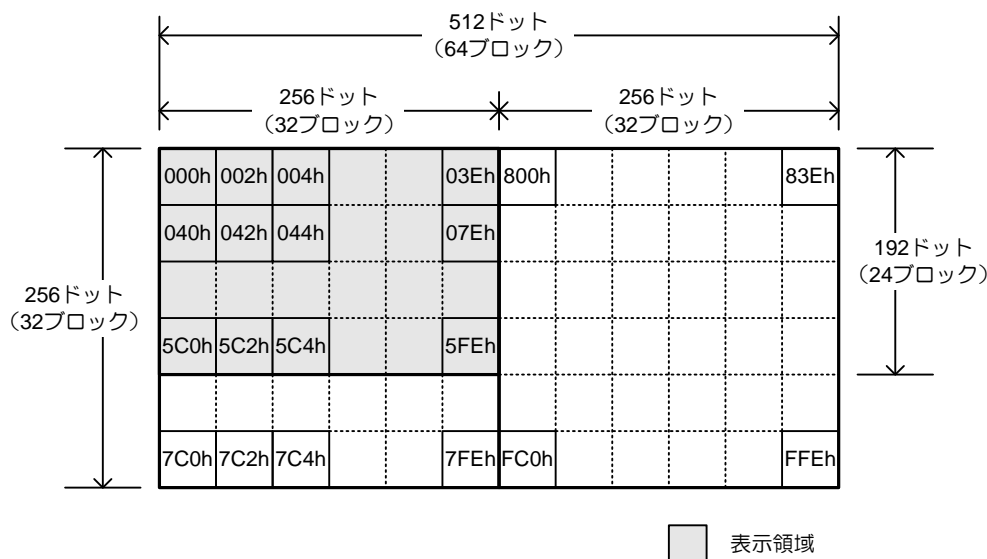


図 6-8 512×256 ドット時のアドレスマッピング (テキスト BG)

4) スクリーンサイズが 512×512 ドットするとき

スクリーンサイズが 512×512 ドットのときのスクリーンデータのアドレスマップは図 6-9 の通りです。

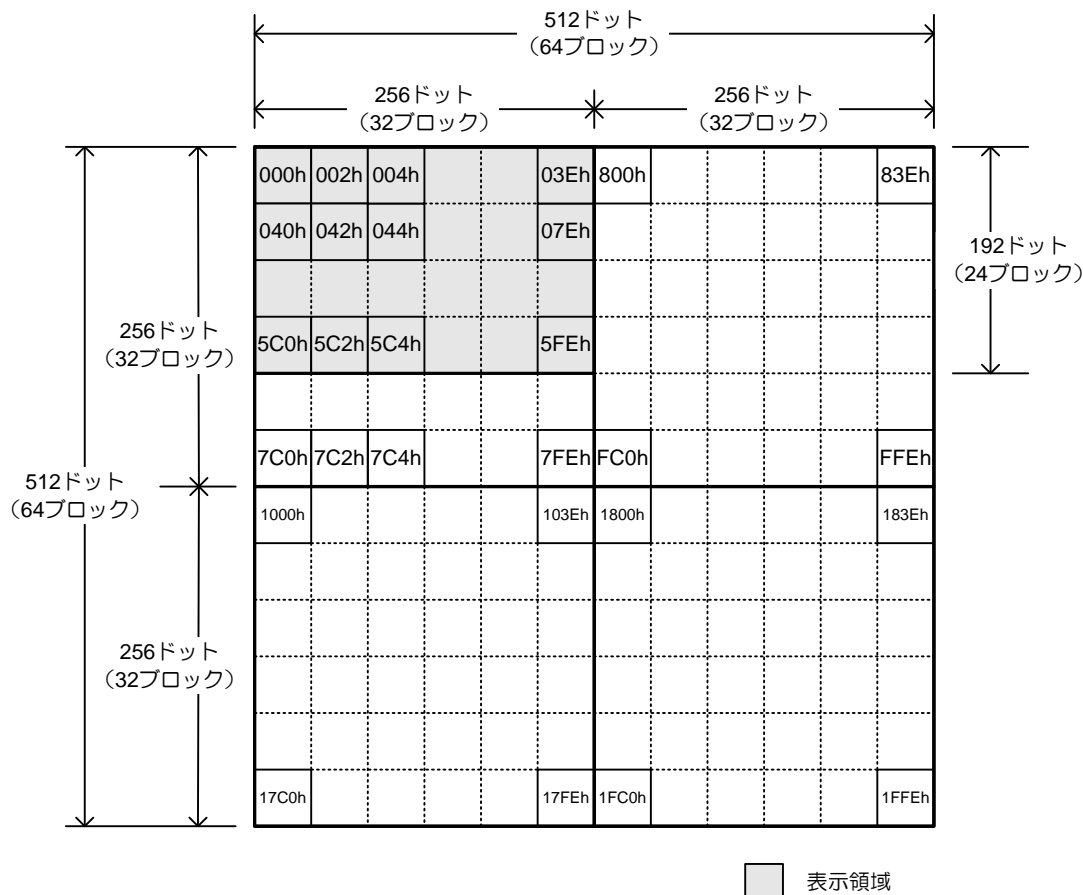


図 6-9 512×512 ドット時のアドレスマッピング (テキスト BG)

6.2.3.2.3 キャラクタデータフォーマット

テキスト BG16 色モードと、テキスト BG 256 色モードにおけるキャラクタデータフォーマットを以下に示します。
キャラクタ表示は 8×8 ドットのキャラクタを定義する場合を表しています。

6.2.3.2.3.1 16 色モード

16 色モード時のキャラクタデータフォーマット、キャラクタ表示とピクセルデータの対応、アドレスマッピング（図 6-10）を以下に示します。

16 色モード時のキャラクタデータフォーマット

15	12	11	8	7	4	3	0
P3		P2		P1		P0	
4 ピクセル分のデータ (4bit / pixel)							

キャラクタ表示

P0	P1	P2	P3				

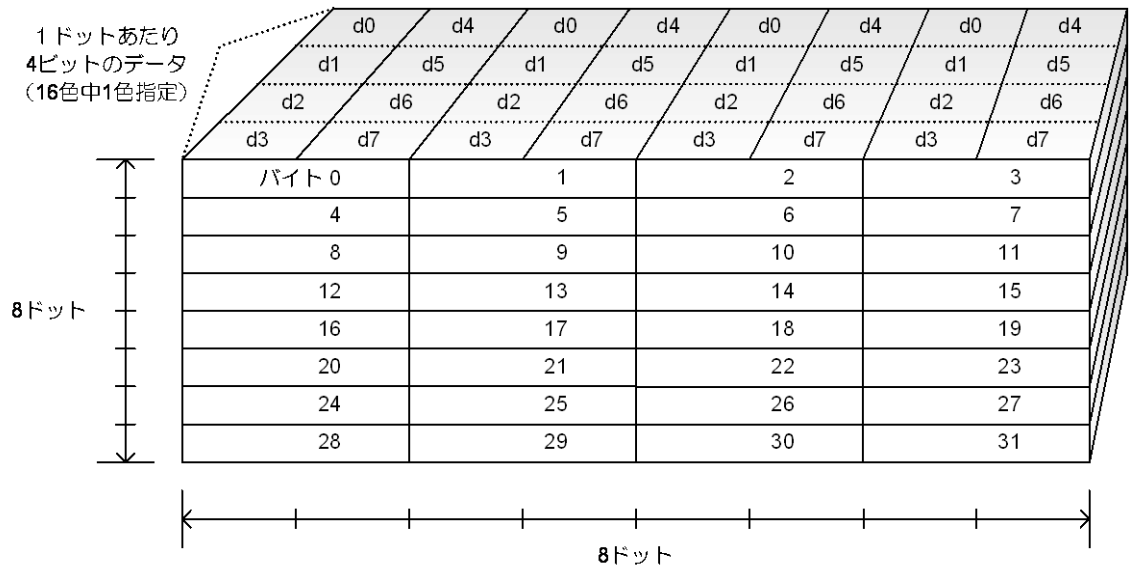


図 6-10 キャラクタデータのアドレスマッピング（テキスト BG 16 色モード）

6.2.3.2.3.2 256 色モード

256 色モード時のキャラクタデータフォーマット、キャラクタ表示とピクセルデータの対応、アドレスマッピング（図 6-11）を以下に示します。

256 色モード時のキャラクタデータフォーマット

15	12	11	8	7	4	3	0
P1				P0			
2 ピクセル分のデータ (8bit / pixel)							

キャラクタ表示

P0	P1						

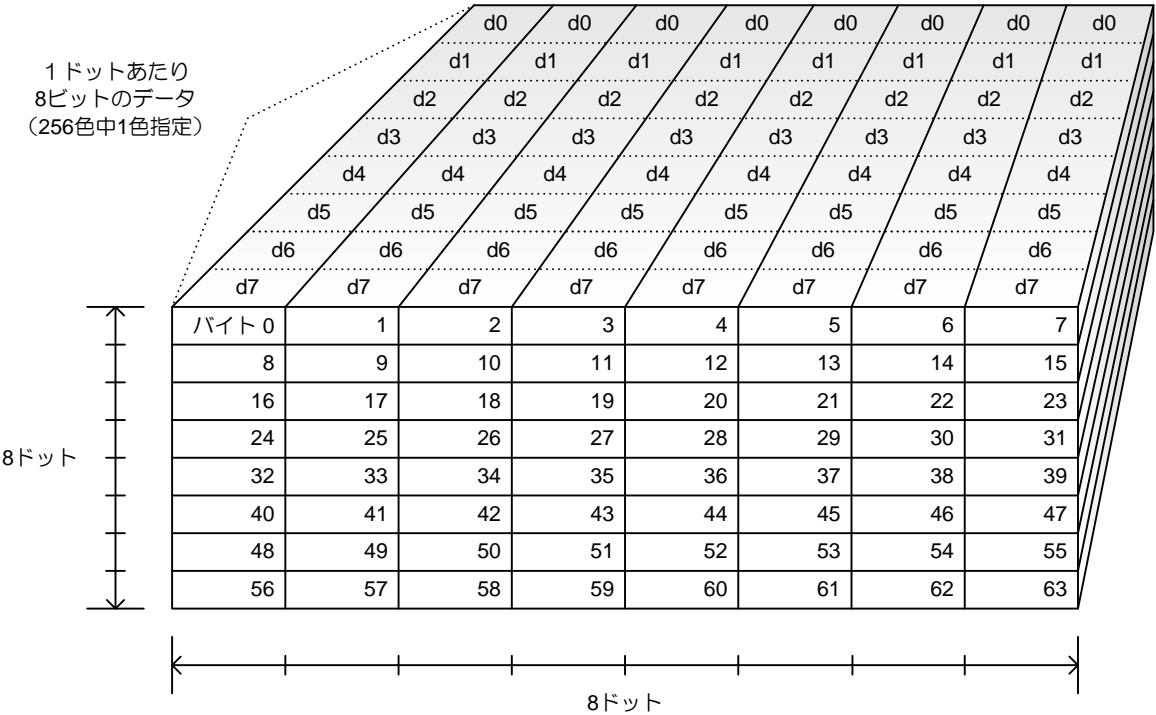


図 6-11 キャラクタデータのアドレスマッピング（テキスト BG 256 色モード）

6.2.3.3 アフィン BG

回転拡大縮小が可能なキャラクタ方式の BG タイプです。

注意事項

アフィン BG は BG2 または BG3 でのみ設定可能です。
アフィン BG 画面でのカラーモードは 256 色モード固定となります。
そのため BG コントロールレジスタのカラーモード指定は無効となります。
また、HV フリップ（垂直・水平反転）処理をすることができません。

6.2.3.3.1 スクリーンデータフォーマット

BG スクリーンデータは、BG コントロールレジスタで指定した BG スクリーンベースブロックの先頭アドレスから格納してください。
アフィン BG 画面の BG スクリーンデータは、次に示すフォーマットで設定します。

アフィン BG 画面のスクリーンデータ

7							0
キャラクタネーム							

- [d07～d00]：キャラクタネーム
BG コントロールレジスタで指定したキャラクタベースブロックの先頭アドレスを起点としたキャラクタの番号を指定します。

6.2.3.3.2 スクリーンデータのアドレスマッピング

1) スクリーンサイズが 128×128 ドットの時

スクリーンサイズが 128×128 ドットの際のスクリーンデータのアドレスマップは図 6-12 の通りです。

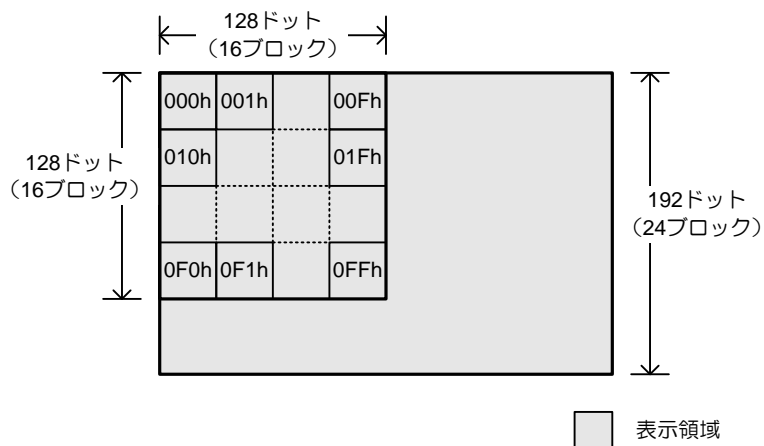


図 6-12 128×128 ドット時のアドレスマッピング (アフィン BG)

2) スクリーンサイズが 256×256 ドットの時

スクリーンサイズが 256×256 ドットの際のスクリーンデータのアドレスマップは図 6-13 の通りです。

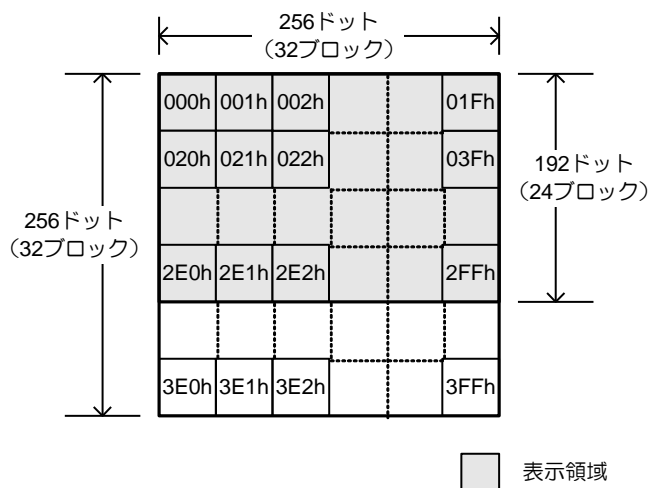


図 6-13 256×256 ドット時のアドレスマッピング (アフィン BG)

3) スクリーンサイズが 512×512 ドットの時

スクリーンサイズが 512×512 ドットの際のスクリーンデータのアドレスマップは図 6-14 の通りです。

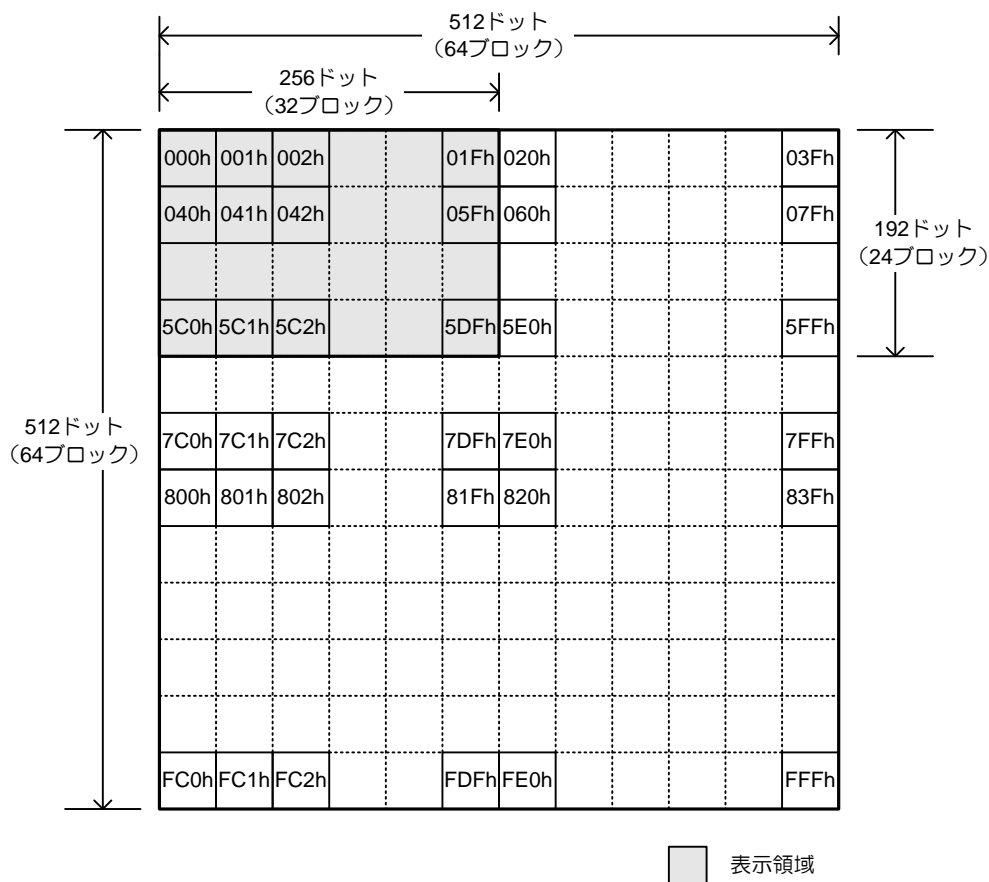


図 6-14 512×512 ドット時のアドレスマッピング (アフィン BG)

4) スクリーンサイズが 1024×1024 ドットするとき
スクリーンサイズが 1024×1024 ドットのときのスクリーンデータのアドレスマップは図 6-15 の通りです。

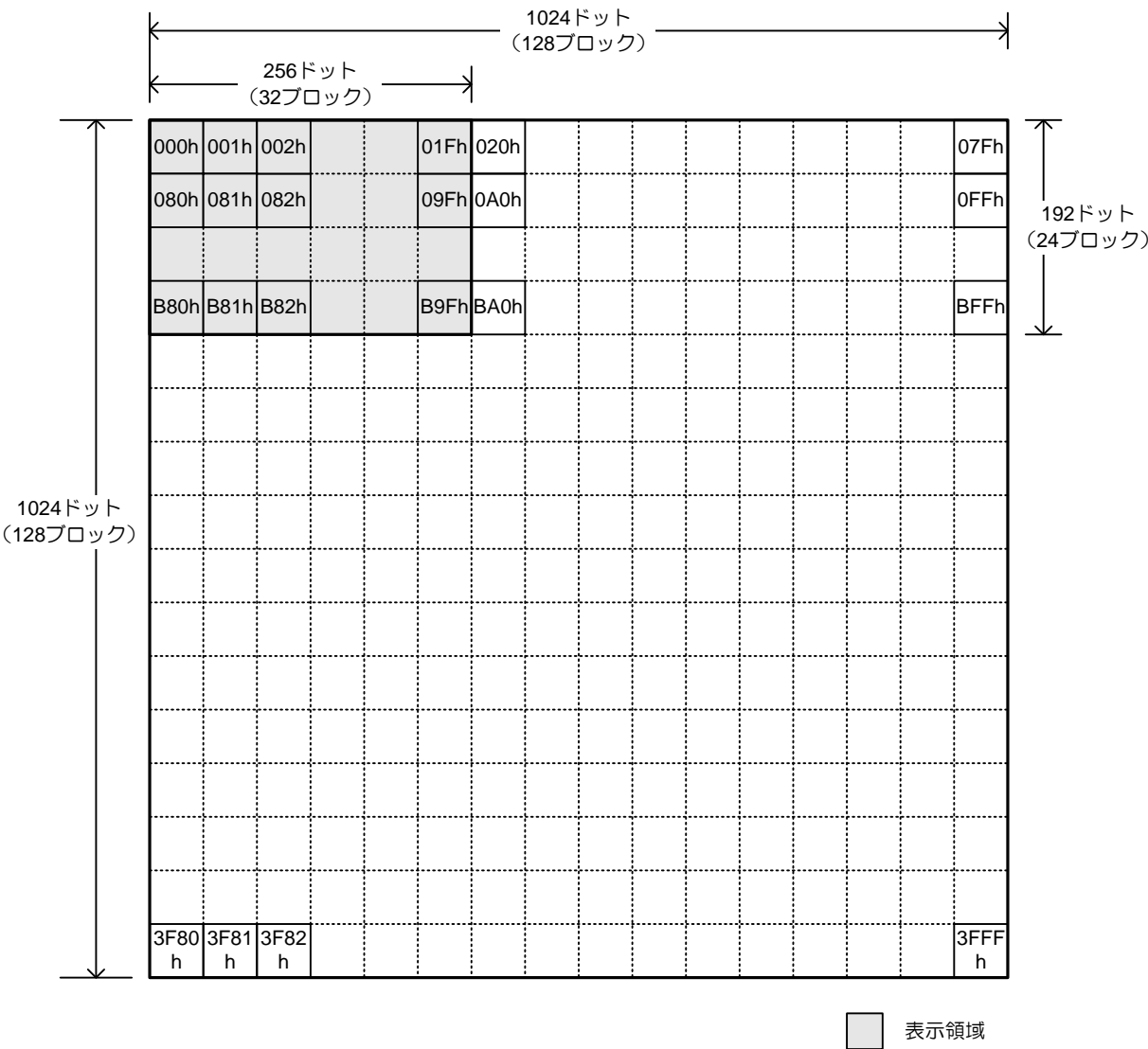


図 6-15 1024×1024 ドット時のアドレスマッピング (アフィン BG)

6.2.3.3.3 キャラクタデータフォーマット

アフィン BG 画面におけるキャラクタデータフォーマットを以下に示します。
キャラクタ表示は8×8 ドットのキャラクタを定義する場合を表しています。
図 6-16 にキャラクタデータのアドレスマッピングを示します。

キャラクタデータフォーマット

15	12	11	8	7	4	3	0
P1				P0			
2 ピクセル分のデータ (8bit / pixel)							

キャラクタ表示

P0	P1						

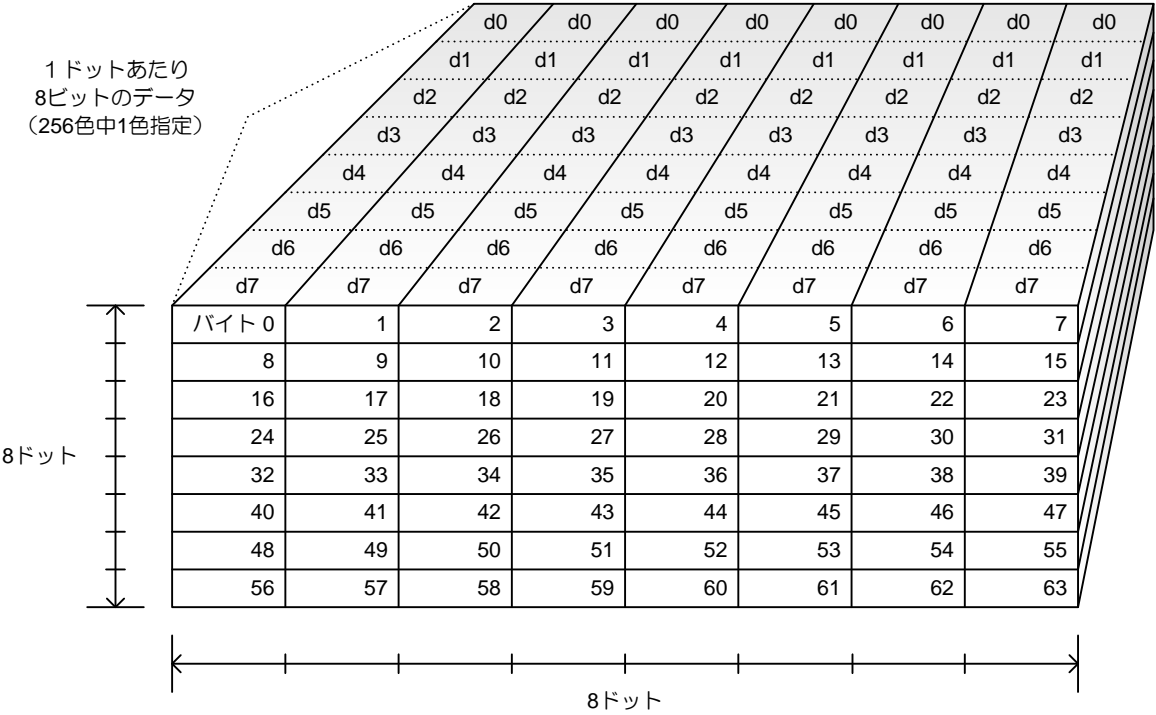


図 6-16 キャラクタデータのアドレスマッピング (アフィン BG)

6.2.3.4 256 色×16 パレットキャラクタ BG（アフィン拡張 BG）

BG コントロールレジスタで「256 色×16 パレットキャラクタ BG」を選択してください。
256 色×16 パレットキャラクタ BG は、BG タイプにアフィン拡張 BG を選択し、BG コントロールレジスタでカラーモードを 0 に設定することで選択できます。

注意事項

256 色×16 パレットキャラクタ BG は BG2 または BG3 でのみ設定可能です。

6.2.3.4.1 スクリーンデータフォーマット

256 色×16 パレット BG 画面のスクリーンデータ

15				12		11		10		9		8		7														0
						VF		HF																				
カラーパレット				フリップ		キャラクタネーム																						

- [d15～d12]：カラーパレット
拡張パレットがイネーブルの場合、キャラクタに適用する拡張パレットを 0～15 の範囲で指定します。
拡張パレットがディセーブルの場合は無効となり標準パレットが使用されます。
なお、拡張パレットのイネーブル／ディセーブルは DISPCNT [DB_DISPCNT] レジスタで設定できます。
- [d11～d10]：フリップ
 - VF：垂直反転フラグ、HF：水平反転フラグ

0	フリップしない
1	フリップする
- [d09～d00]：キャラクタネーム
BG コントロールレジスタで指定したキャラクタベースブロックの先頭アドレスを起点としたキャラクタの番号を指定します。

6.2.3.4.2 キャラクタデータフォーマット

キャラクタデータフォーマットはテキスト BG の 256 色モードキャラクタデータと同一です。

図 6-17 にキャラクタデータのアドレスマッピングを示します。

キャラクタデータフォーマット

15	12	11	8	7	4	3	0
P1				P0			
2 ピクセル分のデータ (8bit / pixel)							

キャラクタ表示

P0	P1						

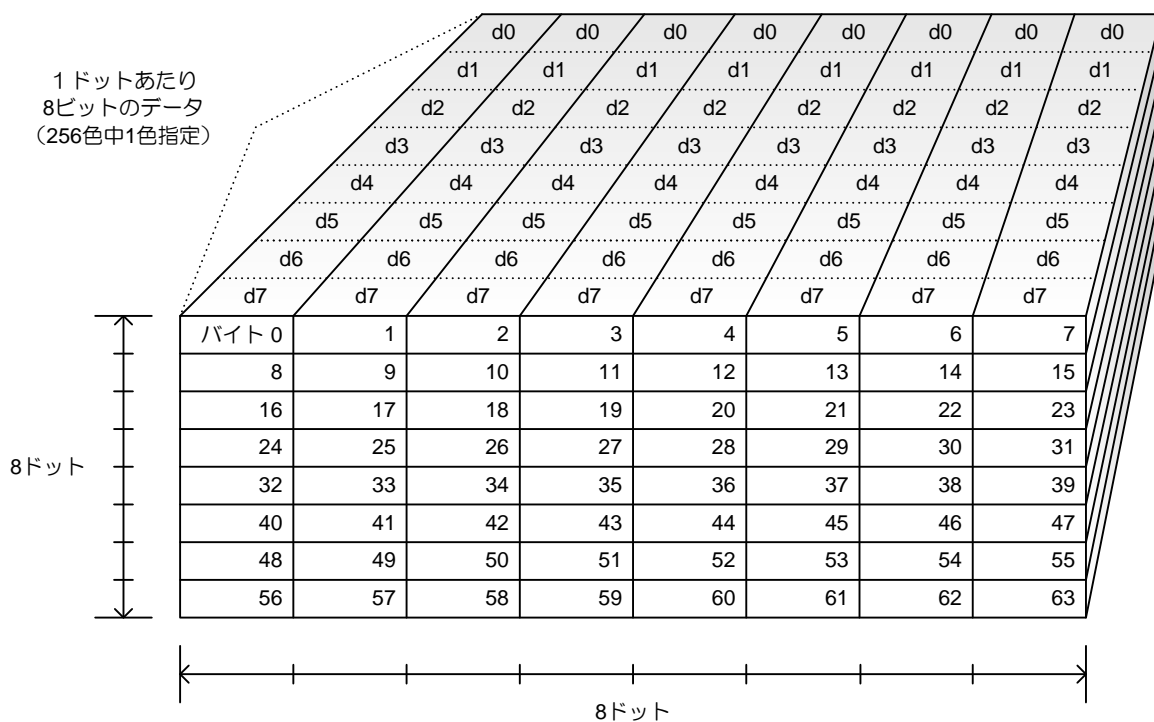


図 6-17 キャラクタデータのアドレスマッピング (256 色×16 パレットキャラクタ BG)

6.2.4 ビットマップ BG

ビットマップ BG では、BG 画面の構成要素をピクセル単位で扱い、VRAM の内容（フレームバッファ）を、画面の 1 ピクセルずつのカラーデータとして表示します。

注意事項

256 色ビットマップ BG とダイレクトカラービットマップ BG は BG2 または BG3 でのみ設定可能です。
大画面 256 色ビットマップ BG は、2D グラフィックスエンジン A の BG2 でのみ設定可能です。

6.2.4.1 256 色ビットマップ BG（アフィン拡張 BG）

BG コントロールレジスタで「256 色ビットマップ BG」を選択してください。

256 色ビットマップ BG は、BG タイプにアフィン拡張 BG を選択し、BG コントロールレジスタでカラーモードを 1 に、キャラクタベースブロックの CB0 を 0 に設定することで選択できます。

6.2.4.1.1 ピクセルデータフォーマット

256 色ビットマップ BG におけるピクセルデータフォーマットを以下に示します。

256 色ビットマップ BG・ピクセルデータフォーマット

7							0
カラーNo.							

6.2.4.1.2 ピクセルデータの VRAM マップ

BG コントロールレジスタのスクリーンベースアドレスにより、ビットマップデータの格納されている VRAM 上のアドレスを指定します（16K バイト単位）。

DISPCNT レジスタのスクリーンベースオフセットの値は無効となります。

6.2.4.2 ダイレクトカラービットマップ BG（アフィン拡張 BG）

BG コントロールレジスタで「ダイレクトカラービットマップ BG」を選択してください。

ダイレクトカラービットマップ BG は、BG タイプにアフィン拡張 BG を選択し、BG コントロールレジスタでカラーモードを 1 に、キャラクタベースブロックの CB0 を 1 に設定することで選択できます。

6.2.4.2.1 ピクセルデータフォーマット

ダイレクトカラービットマップ BG におけるピクセルデータフォーマットを以下に示します。

ダイレクトカラービットマップ BG・ピクセルデータフォーマット

15	14				10	9	8	7		5	4				0
A	BLUE					GREEN					RED				

6.2.4.2.2 ピクセルデータの VRAM マップ

BG コントロールレジスタのスクリーンベースアドレスにより、ビットマップデータの格納されている VRAM 上のアドレスを指定します（16K バイト単位）。

DISPCNT レジスタのスクリーンベースオフセットの値は無効となります。

6.2.4.3 大画面 256 色ビットマップ BG

大画面 256 色ビットマップ BG は 2D グラフィックスエンジン B では使用できません。

6.2.4.3.1 ピクセルデータフォーマット

大画面 256 色ビットマップ BG におけるピクセルデータフォーマットを以下に示します。

大画面 256 色ビットマップ BG・ピクセルデータフォーマット

7							0
カラーNo.							

6.2.4.3.2 ピクセルデータの VRAM マップ

ピクセルデータの先頭アドレスは、BG-VRAM の先頭アドレス（0x6000000 番地）に固定となります。
BG コントロールレジスタのスクリーンベースアドレスおよび DISPCNT レジスタのスクリーンベースオフセットの値は無効となります。

6.2.5 BG スクロール

テキスト BG スクリーン各々に対し、表示画面とのオフセット値をドット単位に設定することができます。オフセットレジスタはテキスト BG に対してのみ有効です。アフィン BG およびビットマップモード BG をオフセット表示するには、BG 参照開始点（「BG の回転拡大縮小（アフィン変換）」参照）を設定します。

BG オフセット設定レジスタ

(2D_A) 名称 BGxOFS (x=0~3)
アドレス 0x04000010, 0x04000014, 0x04000018, 0x0400001C 属性 W 初期値 0x00000000
(2D_B) 名称 DB_BGxOFS (x=0~3)
アドレス 0x04001010, 0x04001014, 0x04001018, 0x0400101C 属性 W 初期値 0x00000000

31								24	23							16	15									8	7							0

図 6-18 に BG スクロールのオフセット概念図を示します。

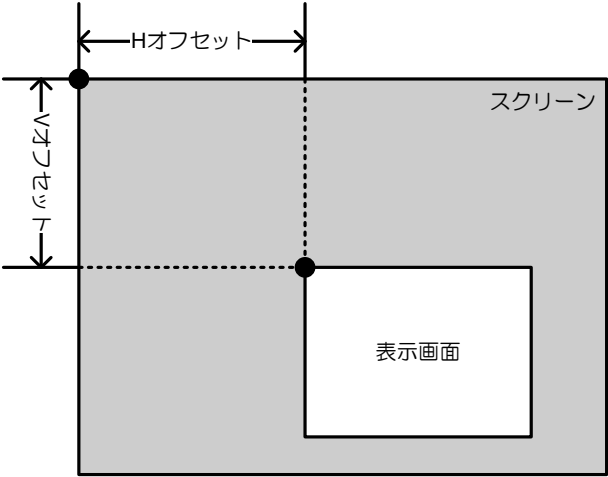
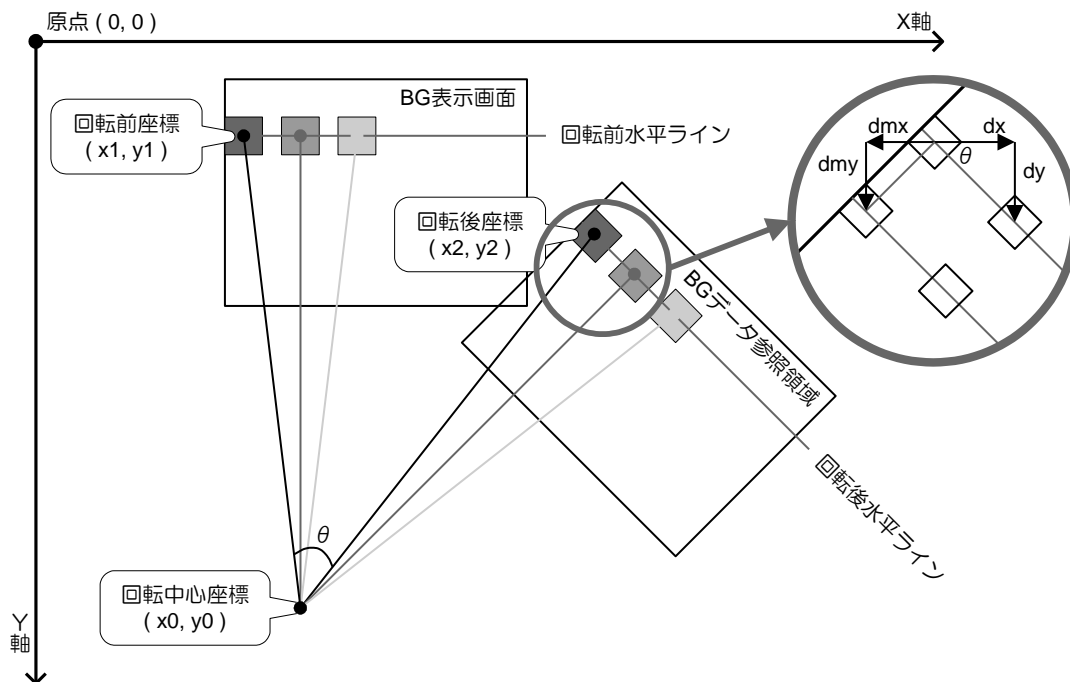


図 6-18 オフセット概念

6.2.6 BG の回転拡大縮小（アフィン変換）

BG を表示する際、BG ピクセルを最左上から順に、水平に参照していきませんが、参照方向に角度をつけることによって BG の回転表示を可能にしています。

BG の回転拡大縮小処理の概念図を図 6-19 に示します。



$$\begin{aligned}
 dx \text{ (同ライン } x \text{ 方向参照移動量)} &= (1/\alpha) \cos \theta \\
 dy \text{ (同ライン } y \text{ 方向参照移動量)} &= -(1/\beta) \sin \theta \\
 dmx \text{ (次ライン } x \text{ 方向参照移動量)} &= (1/\alpha) \sin \theta \\
 dmy \text{ (次ライン } y \text{ 方向参照移動量)} &= (1/\beta) \cos \theta
 \end{aligned}$$

ただし α は X 軸方向の拡大率、 β は Y 軸方向の拡大率

図 6-19 BG の回転拡大縮小

座標 $(x1, y1)$ のアフィン変換後の座標 $(x2, y2)$ は、次の演算式によって求められます。

$$\begin{bmatrix} x2 \\ y2 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x1 - x0 \\ y1 - y0 \end{bmatrix} + \begin{bmatrix} x0 \\ y0 \end{bmatrix}$$

$$A = \frac{1}{\alpha} \cos \theta, \quad B = \frac{1}{\alpha} \sin \theta, \quad C = -\frac{1}{\beta} \sin \theta, \quad D = \frac{1}{\beta} \cos \theta$$

● BG の回転拡大縮小処理

1) 表示画面の左上座標のアフィン変換結果を上記の式によって算出し、BG データ参照開始点を以下のレジスタに設定してください。

2D グラフィックスエンジン A BGxX, BGxY レジスタ (x=2,3)

2D グラフィックスエンジン B DB_BGxX, DB_BGxY レジスタ (x=2,3)

また、上図を参考にして BG データ参照方向を以下のレジスタに設定してください。

2D グラフィックスエンジン A BGxPA, BGxPB, BGxPC, BGxPD レジスタ (x=2,3)

2D グラフィックスエンジン B DB_BGxPA, DB_BGxPB, DB_BGxPC, DB_BGxPD, レジスタ (x=2,3)

2) 画像処理回路は、上記レジスタにセットされた BG データ参照開始点に対し、X 方向の増加分 (dx, dy) を累積加算して X 方向の座標を算出していきます。

3) ラインが進んだ場合には、参照開始点に対し、Y 方向の増加分 (dmx, dmy) を累積加算して次ラインの描画開始点座標を算出した後、2) の処理を行います。

4) ただし、H ブランク期間中に BG データ参照開始点レジスタが書き換えられた場合、そのレジスタに関わる Y 方向の累積加算は行いません。CPU がライン毎にアフィン変換パラメータや中心座標の変更を行いたいときにこのモードを使用します。

BG データ参照開始点設定レジスタ

(2D_A) 名称 BGxX (x=2, 3)					アドレス 0x04000028, 0x04000038					属性 W		初期値 0x00000000				
(2D_B) 名称 DB_BGxX (x=2, 3)					アドレス 0x04001028, 0x04001038					属性 W		初期値 0x00000000				
31		24		23		16		15		8		7		0		
										INTEGER_SX		DECIMAL_SX				
												参照開始点 (アフィン変換結果) の x 座標				

(2D_A) 名称 BGxY (x=2, 3)				アドレス 0x0400002C, 0x0400003C				属性 W	初期値 0x00000000					
(2D_B) 名称 DB_BGxY (x=2, 3)				アドレス 0x0400102C, 0x0400103C				属性 W	初期値 0x00000000					
31				24	23		16	15		8	7		0	
				S	INTEGER_SY							DECIMAL_SY		
				参照開始点 (アフィン変換結果) の y 座標										

BG データ参照方向設定レジスタ

(2D_A) 名称 BGxPA (x=2, 3)		アドレス 0x04000020, 0x04000030		属性 W	初期値 0x0100
(2D_B) 名称 DB_BGxPA (x=2, 3)		アドレス 0x04001020, 0x04001030		属性 W	初期値 0x0100
15	8		7	0	
S	INTEGER_DX		DECIMAL_DX		
同ライン x 方向参照移動量 dx					

(2D_A) 名称 BGxPB (x=2, 3)		アドレス 0x04000022, 0x04000032		属性 W	初期値 0x0000
(2D_B) 名称 DB_BGxPB (x=2, 3)		アドレス 0x04001022, 0x04001032		属性 W	初期値 0x0000
15	8		7	0	
S	INTEGER_DMx		DECIMAL_DMx		
次ライン x 方向参照移動量 dmx					

(2D_A) 名称 BGxPC (x=2, 3)		アドレス 0x04000024, 0x04000034		属性 W	初期値 0x0000
(2D_B) 名称 DB_BGxPC (x=2, 3)		アドレス 0x04001024, 0x04001034		属性 W	初期値 0x0000
15	8		7	0	
S	INTEGER_DY		DECIMAL_DY		
同ライン y 方向参照移動量 dy					

(2D_A) 名称 BGxPD (x=2, 3)		アドレス 0x04000026, 0x04000036		属性 W	初期値 0x0100
(2D_B) 名称 DB_BGxPD (x=2, 3)		アドレス 0x04001026, 0x04001036		属性 W	初期値 0x0100
15	8		7		0
S	INTEGER_DMY		DECIMAL_DMY		
次ライン y 方向参照移動量 dmy					

6.3 OBJ

TWL では、キャラクタ OBJ とビットマップ OBJ の 2 種類の OBJ を扱うことができます。
 キャラクタ OBJ、ビットマップ OBJ それぞれの概要を表 6-7 に示します。

項目	キャラクタ OBJ	ビットマップ OBJ
表示色数	1) 標準パレット時 16 色×16 パレット、 256 色× 1 パレット 2) 拡張パレット使用時 16 色×16 パレット (標準パレット) 256 色×16 パレット (拡張パレット)	32768 色
キャラクタ数 (8×8 ドット換算)	1) 1 次元マッピング時 1,024～8,192 個 (16 色モード) 512～4,096 個 (256 色モード) 2) 2 次元マッピング時 1,024 個 (16 色モード) 512 個 (256 色モード)	1) 1 次元マッピング時 1,024～2,048 個 2) 2 次元マッピング時 256 個
キャラクタサイズ	8×8 ～ 64×64 ドット (12 種類)	
1 画面最大表示数	128 個 (64×64 ドット換算)	
1 ライン最大表示数	128 個 (8× 8 ドット換算)	
機能	HV オフセット、HV フリップ、回転拡大縮小、半透明※ ₁ 、モザイク、優先順位指定	

※₁ OBJ に対するカラー特殊効果は「カラー特殊効果」章を参照してください。

表 6-7 OBJ 概要

● その他

OBJ ウィンドウについては「ウィンドウ」章を参照してください。

● 1 ラインに表示可能な OBJ 数

表 6-7 における OBJ の 1 ライン表示能力は、最大効率時のものです。表示 OBJ が OAM の先頭から連続に配置されている場合、1 ラインに表示可能な OBJ 数は次式にて計算することができます。

$$(\text{H ドット数} \times 6 - 6) / \text{描画サイクル数} = 1 \text{ ライン表示可能 OBJ 数 (最大 128)}$$

「H ドット数」は通常 355 ドットですが、DISPCNT [DB_DISPCNT] レジスタの H ブランク期間の OBJ 処理フラグを 1 に設定した場合は 256 ドットとなります（「LCD」章を参照）。

「× 6」は 1 ドットあたりに OBJ 描画回路が使用できるサイクル数を、「－ 6」は H ライン先頭での OBJ 描画前処理に要するサイクル数を表します。

「描画サイクル数」およびそれぞれの 1 ライン表示可能 OBJ 数は表 6-8 の通りとなります。

OBJ の H サイズ	描画サイクル数		1 ライン表示可能数	
	ノーマル OBJ※ ₁	アフィン OBJ※ ₂	ノーマル OBJ	アフィン OBJ
8	8	26	128	81
16	16	42	128	50
32	32	74	66	28
64	64	138	33	15
128 (64 の倍角)	—	266	—	7

※₁ ノーマル OBJ は OBJ アトリビュート 0 の OBJ モードがノーマル OBJ に設定された OBJ です。

※₂ アフィン OBJ は OBJ アトリビュート 0 の回転拡大縮小イネーブルフラグがイネーブルに設定された OBJ です。

表 6-8 描画サイクル数と 1 ラインに表示可能な OBJ 数

表 6-8 は最大効率時のものです。

実際、OAM には描画ラインから外れた OBJ も存在するため、その分効率が低下します。

描画ライン外の OBJ は 2 サイクルのロスになります。

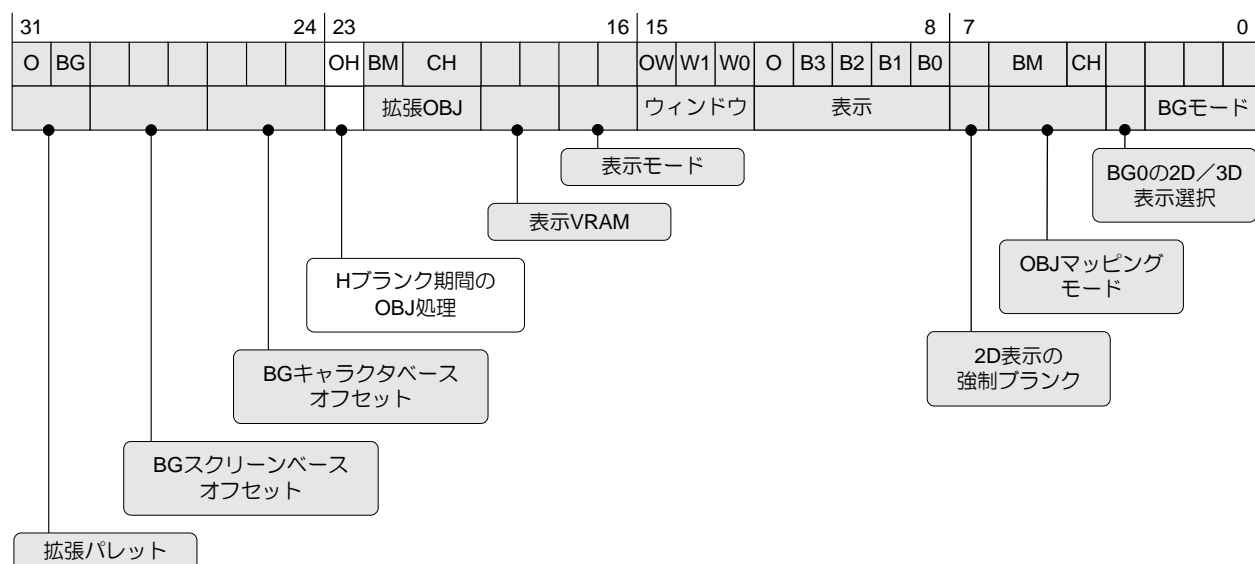
6.3.1 OBJ 表示制御

OBJ 機能の全体的な設定は、2D グラフィックスエンジン A では DISPCNT レジスタ、2D グラフィックスエンジン B では DB_DISPCNT レジスタにて行います。

OBJ ひとつひとつに対する設定は、OAM に格納する OBJ アトリビュートデータにより行います（後述）。

表示コントロールレジスタ（2D グラフィックスエンジン A）

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



● OH[d23] : H ブランク期間 OBJ 処理フラグ

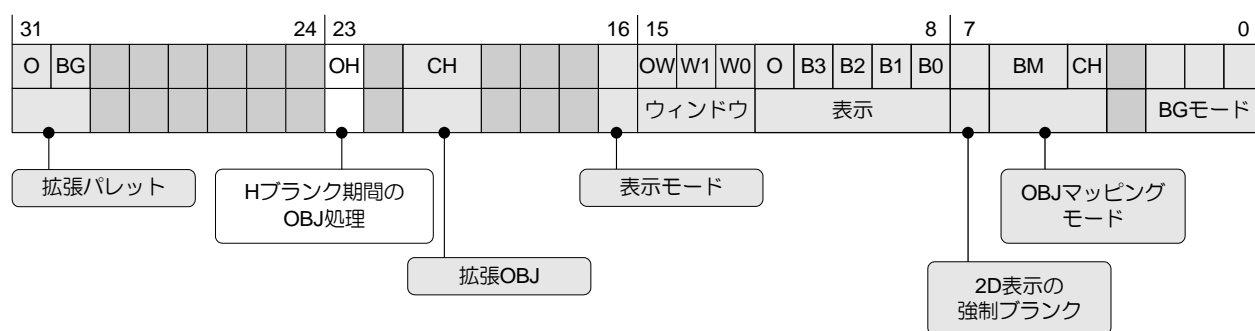
0 を設定した場合、H ラインの全期間（H ブランク期間を含む）において OBJ の描画処理を行います。

1 を設定した場合、表示期間のみ OBJ 描画処理を行い、H ブランク期間には行いません。

この場合、最大数の OBJ 表示パフォーマンスを発揮できないことになります。

表示コントロールレジスタ 1（2D グラフィックスエンジン B）

名称 DB_DISPCNT アドレス 0x04001000 属性 R/W 初期値 0x00000000



● OH[d23] : H ブランク期間 OBJ 処理フラグ

0 を設定した場合、H ラインの全期間（H ブランク期間を含む）において OBJ の描画処理を行います。

1 を設定した場合、表示期間のみ OBJ 描画処理を行い、H ブランク期間には行いません。

この場合、最大数の OBJ 表示パフォーマンスを発揮できないことになります。

6.3.2 OAM

OBJ は、OAM（オブジェクト・アトリビュート・メモリ）にデータを登録することで表示を行います。
OBJ データは、TWL プロセッサ内部の OAM（2D グラフィックスエンジン A は 0x07000000~0x070003FF 番地、2D グラフィックスエンジン B は 0x07000400~0x070007FF 番地の 1KB）に 128 個分書くことができ、LCD 上に任意の大きさの OBJ キャラクタを 128 個表示することができます。

6.3.2.1 メモリマップ

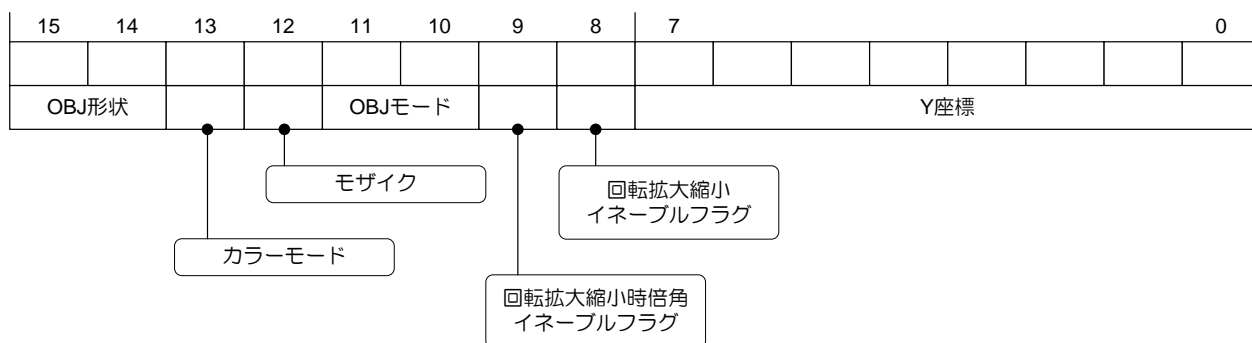
OAM には、48 ビット×128 個分の OBJ アトリビュートを書くことができます。また、OBJ において回転拡大縮小を行う場合、アフィン変換パラメータ PA, PB, PC, PD を図 6-20 の通りに合計 32 組書くことができます。



図 6-20 OAM メモリマップ（2D グラフィックスエンジン B はアドレスに+0x400）

6.3.2.2 OAM データフォーマット

OBJ アトリビュート 0



● [d15～d14] : OBJ 形状

OBJ の形状を設定します。この設定と OBJ アトリビュート 1 のサイズ指定によって OBJ の縦横ドット数を決定します。OBJ アトリビュート 1 のサイズ指定も参照してください。

00	正方形
01	横長・長方形
10	縦長・長方形
11	禁止コード

● [d13] : カラーモード

OBJ キャラクタデータを 16 色フォーマットで参照するか、256 色フォーマットで参照するかを設定します。

ダイレクトカラービットマップ OBJ 設定時は、カラーモードを必ず '0' に設定してください。

0	16 色モード
1	256 色モード

● [d12] : OBJ モザイク

0	モザイク OFF
1	モザイク ON

● [d11～d10] : OBJ モード

'00'～'10'設定時はキャラクタ OBJ です。

'10'の OBJ ウィンドウに設定した場合は、通常の OBJ として表示されず、キャラクタデータが 0 以外のドットが OBJ ウィンドウとして扱われます。このため OBJ ウィンドウは任意の形にすることができます。

OBJ ウィンドウ内の表示設定については「ウィンドウ」章を参照してください。

'11'設定時はビットマップ OBJ です。

00	ノーマル OBJ
01	半透明 OBJ
10	OBJ ウィンドウ
11	ビットマップ OBJ

● [d09] : アフィン OBJ の倍角イネーブルフラグ

0	倍角ディセーブル
1	倍角イネーブル

d08 の回転拡大縮小イネーブルフラグが 1 の場合に、OBJ フィールドを倍角にして表示することができます。

倍角 OBJ フィールドを使うことによって、OBJ を回転させた場合に OBJ の一部が欠けることなく表示できます。また、OBJ を拡大させた場合にも、2 倍までであれば OBJ の一部が欠けることなく表示できます。（図 6-21 参照）

d08 の回転拡大縮小イネーブルフラグが 0 の場合にこのビットを '1' にすると、OBJ が非表示となります。



図 6-21 倍角 OBJ フィールドによる拡大・回転表示

- [d08]：回転拡大縮小イネーブルフラグ
イネーブルの場合は、OBJ アトリビュート 1 で設定したアフィン変換パラメータが参照されます。

0	ディセーブル
1	イネーブル

ディセーブル時、d09 の倍角イネーブルフラグが1'にセットされているとその OBJ は非表示となります。

- [d07～d00]：Y 座標
表示画面に対する OBJ の Y 座標を 0～255 の範囲で指定できます。

OBJ アトリビュート 1

15	14	13	12	9			8	7	0						
		VF	HF												
OBJ サイズ		アフィン変換パラメータ選択						X 座標							

● [d15～d14] : OBJ サイズ

OBJ のサイズを設定します。OBJ アトリビュート 0 における OBJ 形状の設定によって、OBJ は表 6-9 の縦横ドット数となります。

OBJ アトリビュート 1 OBJ サイズ OBJ アトリビュート 0 OBJ 形状	OBJ アトリビュート 0 OBJ 形状			
	00	01	10	11
00 (正方形)	8× 8	16×16	32×32	64×64
01 (横長方形)	16× 8	32× 8	32×16	64×32
10 (縦長方形)	8×16	8×32	16×32	32×64
11 (設定禁止)	設定禁止			

表 6-9 OBJ 形状と OBJ サイズの設定

● [d13～d09] : アフィン変換パラメータ選択

OAM のアフィン変換パラメータ PA～PD を 1 セットとして、32 セットのうちどれを参照するかを選択します。OBJ アトリビュート 0 の回転拡大縮小フラグが 0 (ディセーブル) の場合は、[d13]VF が垂直フリップ、[d12]HF が水平フリップフラグとして扱われます。

● [d08～d00] : X 座標

表示画面に対する OBJ の X 座標を 0～511 の範囲で指定します。

OBJ アトリビュート 2

15							8		7		0				
カラーパラメータ				表示優先順位			先頭キャラクタネーム								

キャラクタOBJ時 : カラーパレットNo.
 ビットマップOBJ時 : α_{OAM} 値

● [d15～d12] : カラーパラメータ

OBJ アトリビュート 0 の OBJ モード指定がキャラクタ OBJ の場合とビットマップ OBJ の場合とで意味が異なります。キャラクタ OBJ 時は OBJ のカラーパレット No. を指定します。

ビットマップ OBJ 時は、OBJ の α_{OAM} 値を指定します。 α_{OAM} 値はビットマップ OBJ の BG とのブレンディングにおいて係数に使われます（「ビットマップ OBJ」章参照）。

1) キャラクタ OBJ 時 : カラーパレット No.

キャラクタデータに適用するパレットを 16 パレット中 1 パレット指定します。

拡張パレットがディセーブル時において 256 色モード（OBJ アトリビュート 0 参照）の場合、本ビットは無効になります。

なお、拡張パレットのイネーブル/ディセーブルは DISPCNT（2D グラフィックスエンジン B では DB_DISPCNT）レジスタで設定できます。

2) ビットマップ OBJ 時 : α_{OAM} 値

α_{OAM} 値は OBJ の透明度 α の要素で、 $\alpha = \alpha_{BMP} \times (\alpha_{OAM} + 1)$ となります。

α_{BMP} はビットマップ OBJ データで設定してください（「ビットマップ OBJ データ」章参照）。

● [d11～d10] : 表示優先順位

表示の優先順位を指定します。

BG との優先順位関係については「表示優先順位」章を参照してください。

OAM アトリビュート 0 の OBJ モードを「OBJ ウィンドウ」に指定しているときは、このビットによる表示優先順位は無効となります。

ウィンドウ間の表示優先順位については「ウィンドウ」章を参照してください。

● [d09～d00]：先頭キャラクタネーム

OBJ-VRAM 上にマッピングされている OBJ キャラクタデータの先頭に位置する基本キャラクタ No.を書きます。
ビットマップ OBJ モードのときも、8×8 ドットを 1 単位とみなしてキャラクタモード時と同じように指定します。

2 次元マッピングモード時について

2 次元マッピングモードかつ 256 色モードのときには、先頭キャラクタネームの最下位ビットが 0 に固定されます。
また、OBJ-VRAM は 32K バイトまでの領域しか参照されません。

1 次元マッピングモード時について

1 次元マッピングモードの場合、OBJ-VRAM の容量を拡張することができます（RAM バンクコントロールレジスタ 0,1 および表示コントロールレジスタ参照）。

先頭キャラクタネームの設定範囲（10 ビット）で OBJ-VRAM 全域を参照可能にするため、先頭キャラクタネームの境界は OBJ-VRAM の容量によって表 6-10、表 6-11 のように変動します。

OBJ-VRAM 容量	先頭キャラクタネーム境界
32K バイト	32 バイト
64K バイト	64 バイト
128K バイト	128 バイト
256K バイト	256 バイト

表 6-10 キャラクタ OBJ

OBJ-VRAM 容量	先頭キャラクタネーム境界
128K バイト	128 バイト
256K バイト	256 バイト

表 6-11 ビットマップ OBJ

注意事項

2D グラフィックスエンジン B では VRAM 割り当て制限のため、OBJ-VRAM の最大容量は 128K バイトとなります。
このため、キャラクタ OBJ とビットマップ OBJ の容量を 256K バイトに設定することができません。

アフィン変換パラメータ

OBJ のアフィン変換パラメータの決め方については、次の「OBJ の回転拡大縮小（アフィン変換）」章を参照してください。

アフィン変換パラメータ PA

15	14	8	7	0
S_PA	INTEGER_PA		DECIMAL_PA	
同ライン X 方向移動量 dx				

符号付き固定小数点数（符号+整数部 7 ビット+小数部 8 ビット）

アフィン変換パラメータ PB

15	14	8	7	0
S_PB	INTEGER_PB		DECIMAL_PB	
次ライン X 方向移動量 dmx				

符号付き固定小数点数（符号+整数部 7 ビット+小数部 8 ビット）

アフィン変換パラメータ PC

15	14	8	7	0
S_PC	INTEGER_PC		DECIMAL_PC	
同ライン Y 方向移動量 dy				

符号付き固定小数点数（符号+整数部 7 ビット+小数部 8 ビット）

アフィン変換パラメータ PD

15	14	8	7	0
S_PA	INTEGER_PA		DECIMAL_PA	
次ライン Y 方向移動量 dmy				

符号付き固定小数点数（符号+整数部 7 ビット+小数部 8 ビット）

6.3.2.3 OBJ の回転拡大縮小（アフィン変換）

OBJ を表示する際、OBJ キャラクタデータを最左上から順に、水平に参照していきませんが、参照方向に角度をつけることによって OBJ の回転表示を可能にしています。回転の中心は OBJ フィールドの中心（ドットの境界）に固定となっています。参照点が指定 OBJ サイズをオーバーした場合は透明色となります。

OBJ の回転拡大縮小処理の概念図を図 6-22 に示します。

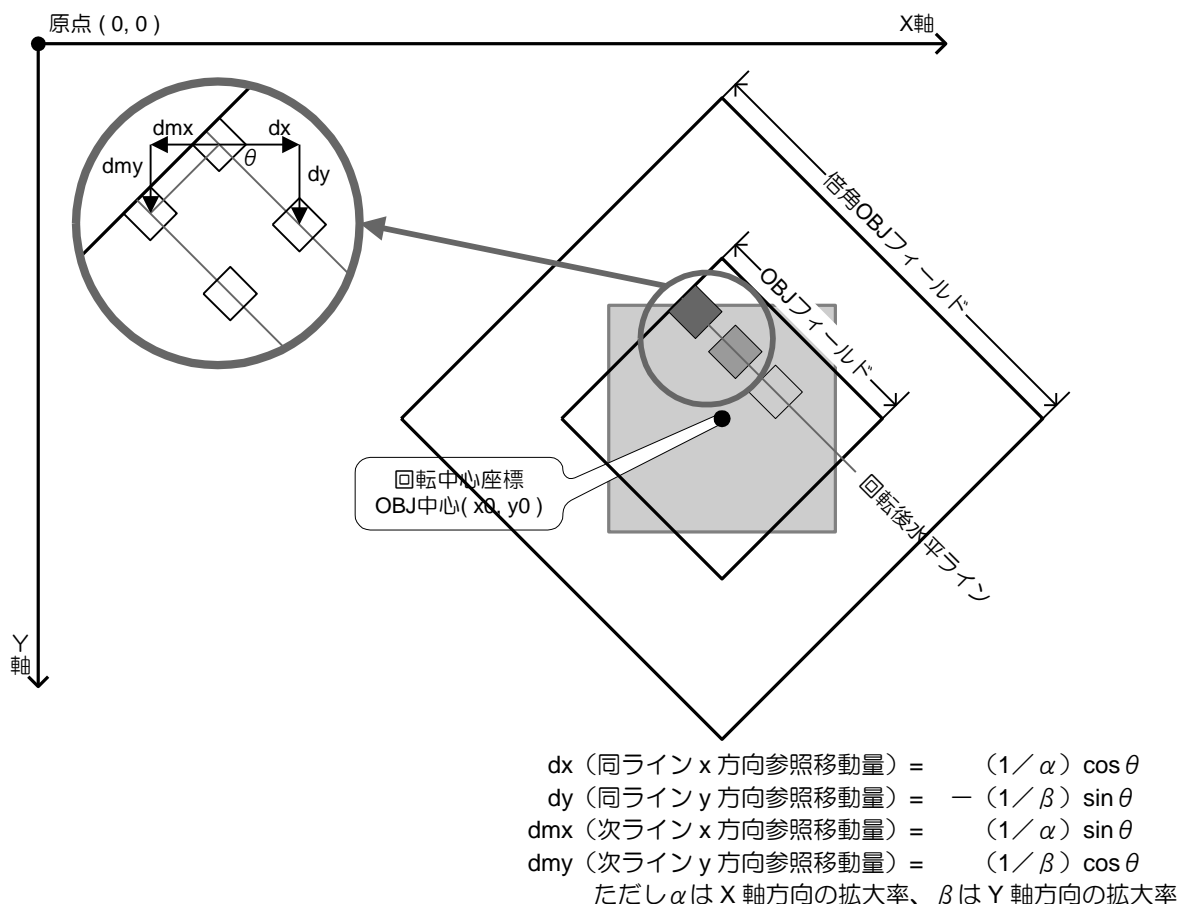


図 6-22 OBJ の回転拡大縮小

● OBJ 回転拡大縮小処理

- 1) OAM に登録する OBJ アトリビュート 1 にて、適用するアフィン変換パラメータ No. を指示します。
また、適用するアフィン変換パラメータ PA, PB, PC, PD を図 6-22 を参考に OAM で設定します。
- 2) 画像処理回路は、OBJ フィールドの中心を回転中心としたデータ参照開始点に対し、X 方向の増加分 (dx, dy) を累積加算して X 方向の座標を算出していきます。
- 3) ラインが進んだ場合には、参照開始点に対し、Y 方向の増加分 (dmx, dmy) を累積加算して次ラインの描画開始点座標を算出した後、2) の処理を行います。

6.3.3 キャラクタ OBJ

OBJ 用のキャラクタデータは、8x8 ドットを基本キャラクタとし、キャラクタ No. が割り当てられます。OBJ の大きさは 8x8~64x64 ドット（計 12 種類）を扱うことが可能です。OBJ キャラクタデータベースアドレスは、VRAM ベースアドレス固定となります。OBJ には 16 色で定義されるものと 256 色で定義されるものがあり、1 基本キャラクタの定義にはそれぞれ 32 バイト、64 バイトが必要です（どちらも BG キャラクタデータと同じフォーマット）。

OBJ キャラクタデータを 16 色フォーマットで参照するか、256 色フォーマットで参照するかは、OAM の OBJ アトリビュート 0 のカラーモードで指定します。

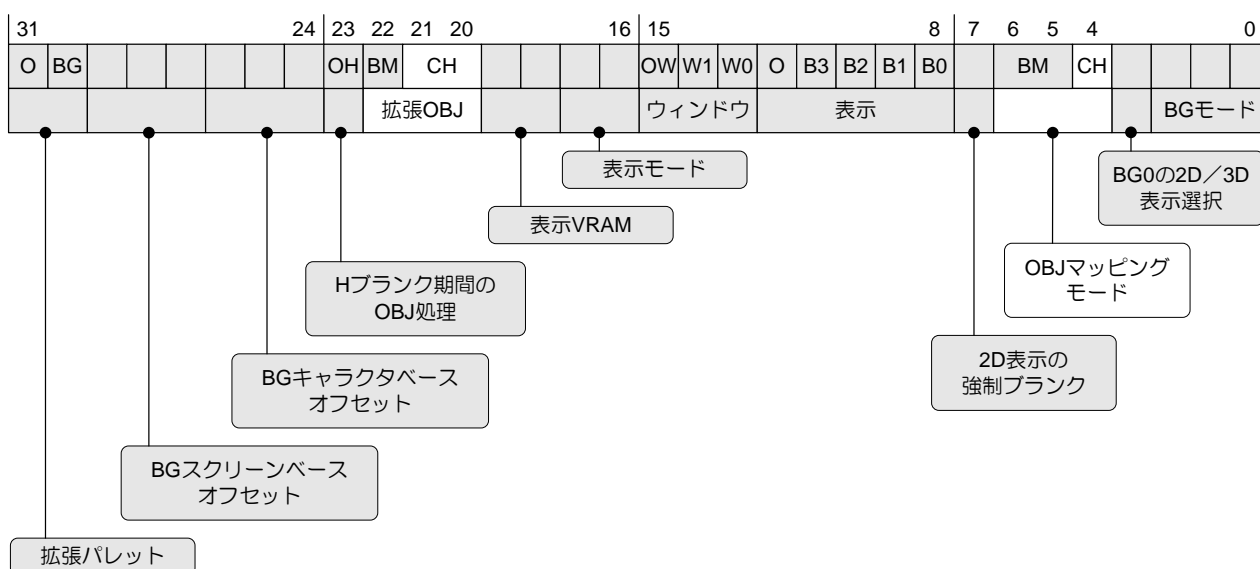
また、「16 色モードに設定した場合」および、「拡張パレットがイネーブル時に 256 色モードにした場合」は OBJ アトリビュート 2 で指定したパレットが使用されます。

なお、拡張パレットのイネーブル/ディセーブルは、2D グラフィックスエンジン A では DISPCNT レジスタ、2D グラフィックスエンジン B では DB_DISPCNT レジスタで設定できます。

キャラクタデータの VRAM マッピングは、2 次元マッピングと 1 次元マッピングの 2 種類から選択できます。

表示コントロールレジスタ（2D グラフィックスエンジン A）

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



- [d22~d20]：OBJ-VRAM 領域拡張フラグ

- CH[d21~d20]：キャラクタ OBJ の VRAM 領域拡張フラグ

OBJ キャラクタデータが一次元マッピングのときの OBJ-VRAM 容量を指定します。00 の場合 AGB と同一になります。

また、OAM の OBJ アトリビュート 2 で設定できる先頭キャラクタネームの境界は表 6-12 のようになります。

00	32K バイト（先頭キャラクタネーム境界 32 バイト）
01	64K バイト（先頭キャラクタネーム境界 64 バイト）
10	128K バイト（先頭キャラクタネーム境界 128 バイト）
11	256K バイト（先頭キャラクタネーム境界 256 バイト）

表 6-12 OBJ アトリビュート 2 の先頭キャラクタネーム境界

注意事項

OBJ-VRAM 領域設定フラグを OBJ に割り当てられた VRAM サイズを超えた設定にした場合、OBJ に割り当てられた VRAM サイズを超えた領域へアクセスしないでください。

- [d06～d04]：OBJ データ・マッピングモード
- CH[d04]：キャラクタ OBJ データ・マッピングモード

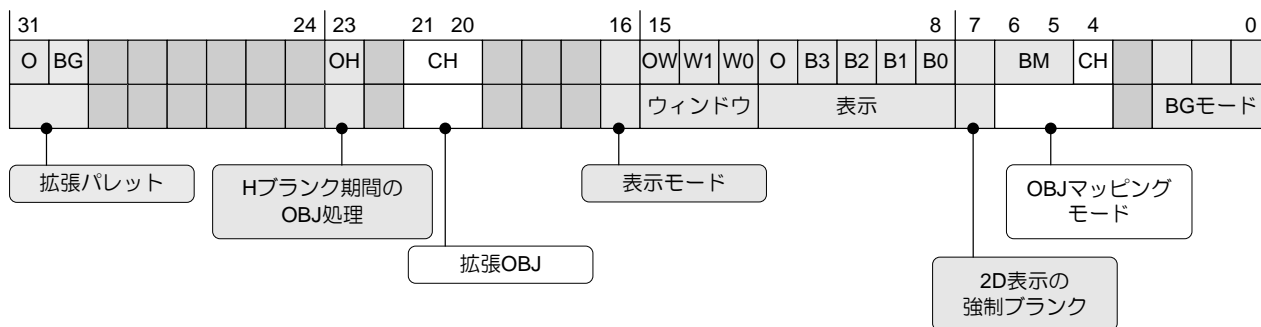
0	2 次元マッピング
1	1 次元マッピング

2 次元マッピング時、OBJ-VRAM 容量は 32K バイトまでしか参照されません。

1 次元マッピング時には、OBJ-VRAM 領域拡張フラグにより 32～256K バイトまでの容量を設定できます。よって 1 次元マッピングの方が、OBJ-VRAM にたくさんの OBJ キャラクタを定義しておくことができます。

表示コントロールレジスタ 1 (2D グラフィックスエンジン B)

名称 DB_DISPCNT アドレス 0x04001000 属性 R/W 初期値 0x00000000



- [d21～d20]：OBJ-VRAM 領域拡張フラグ
 - CH[d21～d20]：キャラクタ OBJ の VRAM 領域拡張フラグ
- OBJ キャラクタデータが 1 次元マッピングのときの OBJ-VRAM 容量を指定します。00 の場合 AGB と同一になります。
- また、OAM の OBJ アトリビュート 2 で設定できる先頭キャラクタネームの境界は表 6-13 のようになります。

00	32K バイト (先頭キャラクタネーム境界 32 バイト)
01	64K バイト (先頭キャラクタネーム境界 64 バイト)
10	128K バイト (先頭キャラクタネーム境界 128 バイト)
11	256K バイト (先頭キャラクタネーム境界 256 バイト)

表 6-13 OBJ アトリビュート 2 の先頭キャラクタネーム境界

注意事項

2D グラフィックスエンジン B では VRAM への割り当てサイズが最大 128K バイトに制限されています。OBJ-VRAM 領域設定フラグを OBJ に割り当てられた VRAM サイズを超えた設定にした場合、OBJ に割り当てられた VRAM サイズを超えた領域へアクセスしないでください。

- [d06～d04]：OBJ データ・マッピングモード
- CH[d04]：キャラクタ OBJ データ・マッピングモード

0	2 次元マッピング
1	1 次元マッピング

2 次元マッピング時、OBJ-VRAM 容量は 32K バイトまでしか参照されません。

1 次元マッピング時には、OBJ-VRAM 領域拡張フラグにより 32～128K バイトまでの容量を設定できます。よって 1 次元マッピングの方が、OBJ-VRAM にたくさんの OBJ キャラクタを定義しておくことができます。

6.3.3.1 キャラクタデータフォーマット

キャラクタ OBJ のキャラクタデータフォーマットを以下に示します。
キャラクタ表示は8×8 ドットのキャラクタを定義する場合を表しています。

6.3.3.1.1 16 色モード

16 色モード時のキャラクタデータフォーマット、キャラクタ表示とピクセルデータの対応、アドレスマッピング（図 6-23）を以下に示します。

16 色モード時のキャラクタデータ

15	12	11	8	7	4	3	0
P3		P2		P1		P0	
4 ピクセル分のデータ (4bit / pixel)							

キャラクタ表示

P0	P1	P2	P3				

キャラクタデータのアドレスマッピング

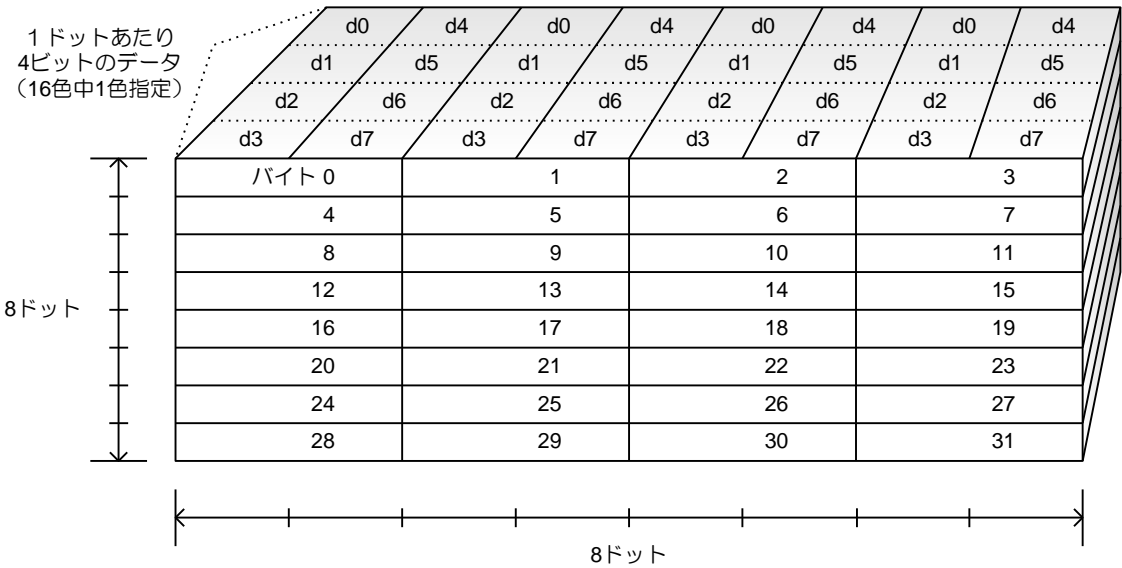


図 6-23 キャラクタデータのアドレスマッピング（キャラクタ OBJ 16 色モード）

6.3.3.1.2 256 色モード

256 色モード時のキャラクタデータフォーマット、キャラクタ表示とピクセルデータの対応、アドレスマッピング（図 6-24）を以下に示します。

256 色モード時のキャラクタデータ

15	12	11	8	7	4	3	0
P1				P0			
2 ピクセル分のデータ (8bit / pixel)							

キャラクタ表示

P0	P1						

キャラクタデータのアドレスマッピング

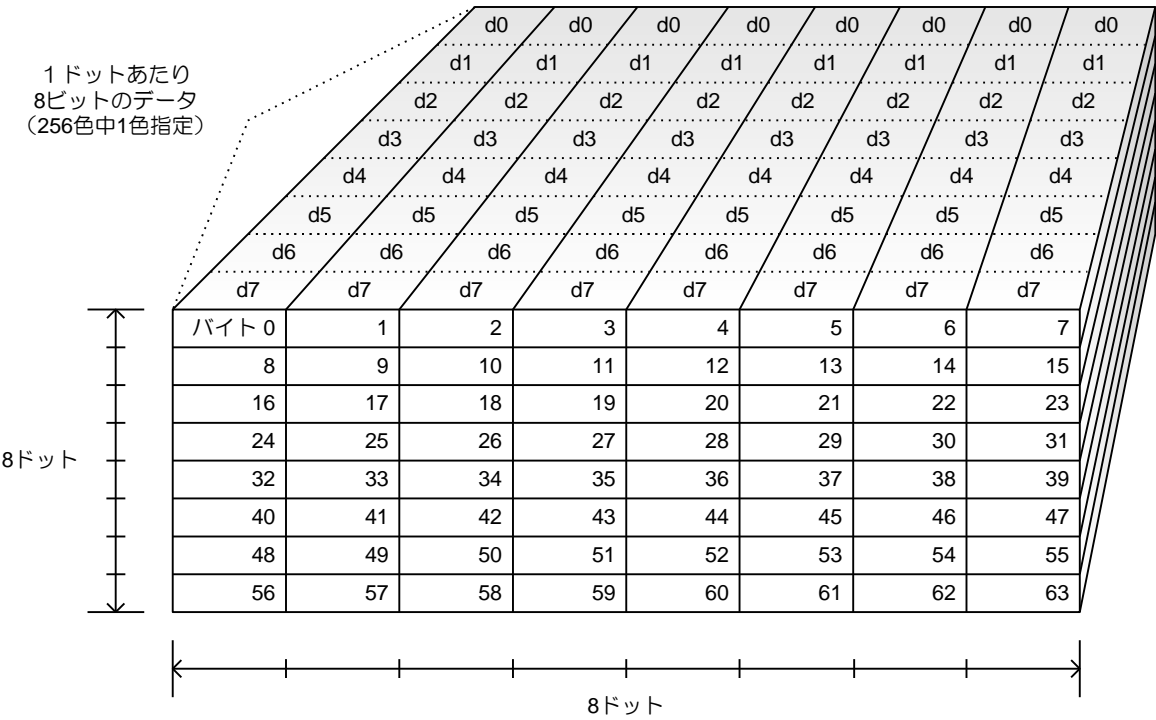


図 6-24 キャラクタデータのアドレスマッピング（キャラクタ OBJ 256 色モード）

6.3.3.2 キャラクタ OBJ データ・マッピングモード

6.3.3.2.1 2次元マッピング

2次元マッピングモード時に 256 色×1 パレットのキャラクタを表示する場合、キャラクタネームの指定が図 6-25 のように偶数に限定されます。

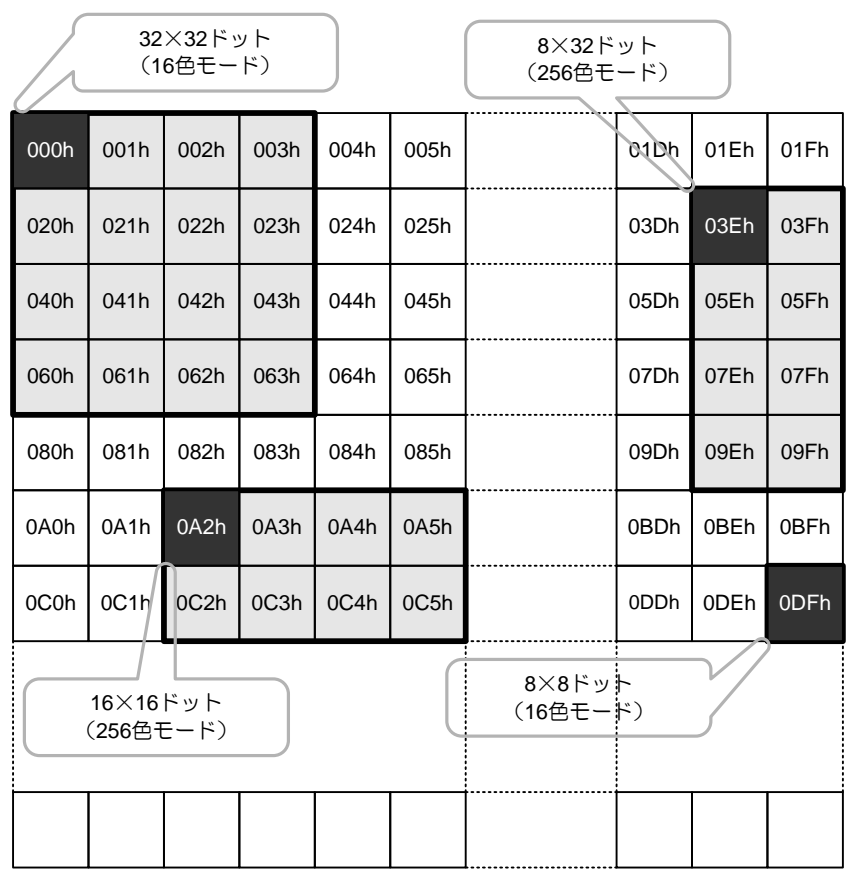


図 6-25 2次元マッピング

6.3.3.2.2 1次元マッピング

キャラクタを構成するデータの格納番地は図 6-26、図 6-27 のように、キャラクタ毎に連続しています。

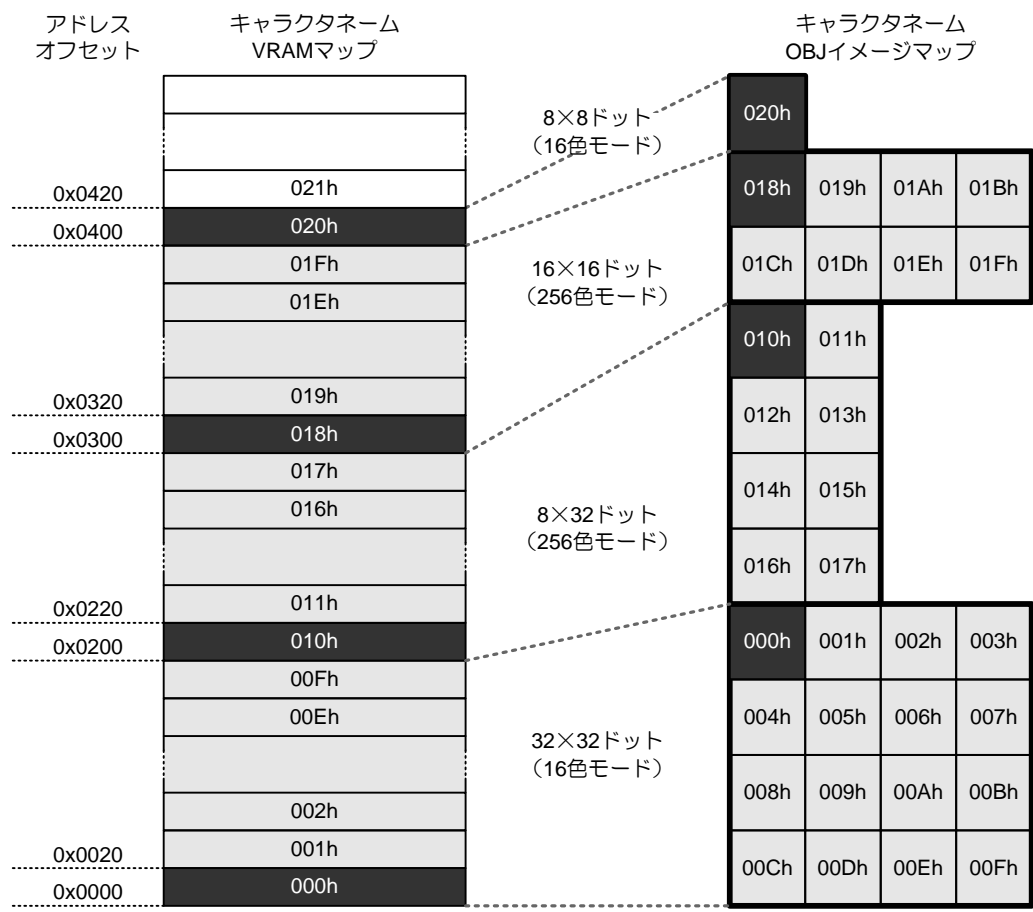


図 6-26 キャラクタネーム境界が 32 バイトのときの 1 次元マッピング

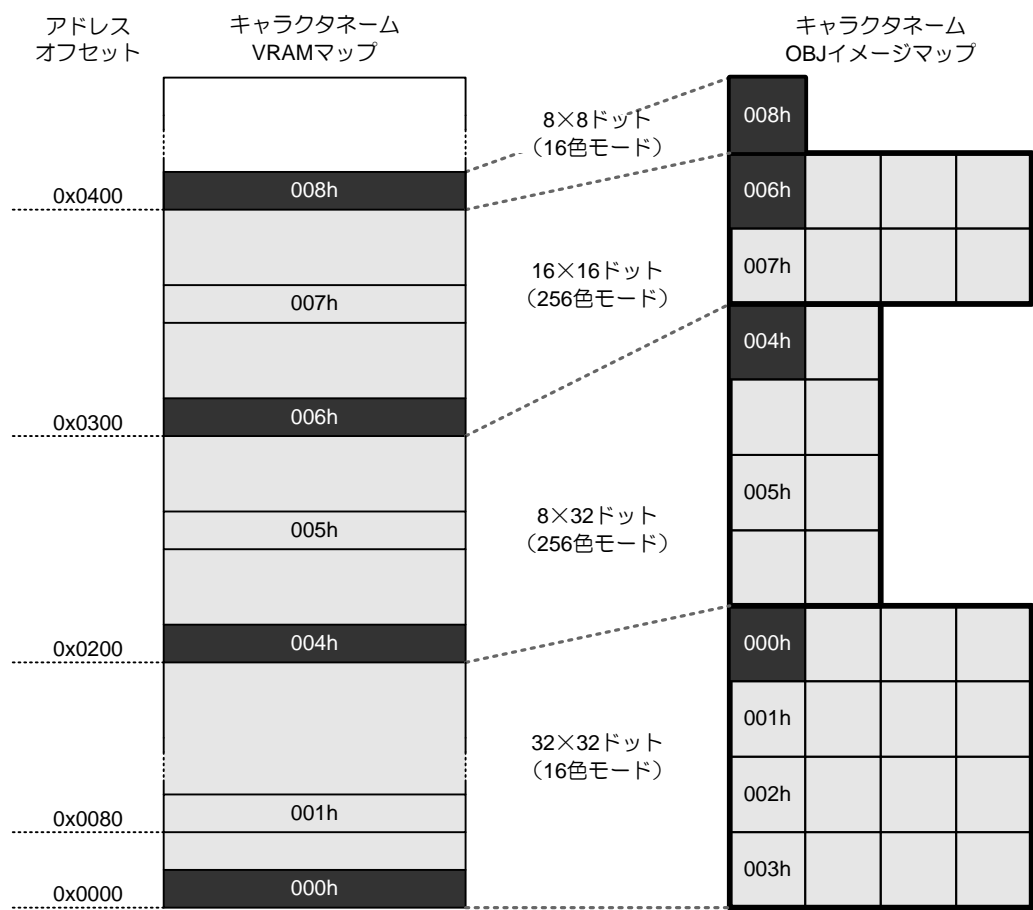


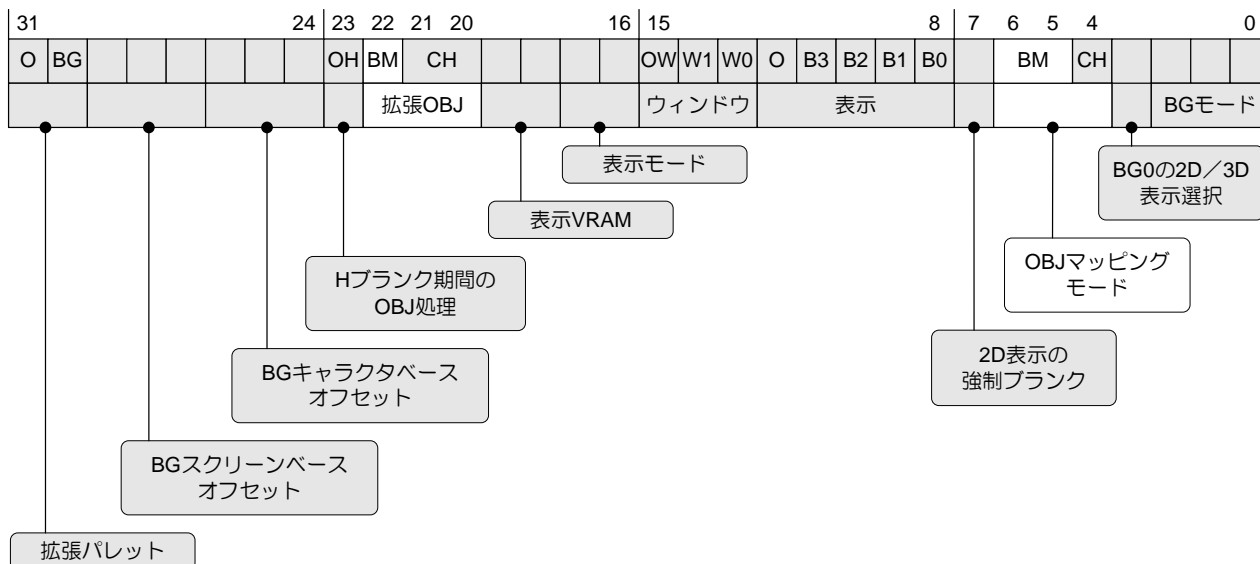
図 6-27 キャラクタネーム境界が 128 バイトのときの 1 次元マッピング

6.3.4 ビットマップ OBJ

ビットマップ OBJ の VRAM 拡張フラグとマッピングモードは、2D グラフィックスエンジン A では DISPCNT レジスタで、2D グラフィックスエンジン B では DB_DISPCNT レジスタで設定します。

表示コントロールレジスタ（2D グラフィックスエンジン A）

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



- [d22～d20]：OBJ-VRAM 領域拡張フラグ

- BM[d22]：ビットマップ OBJ の VRAM 拡張フラグ

OBJ ビットマップデータが 1 次元マッピングのときの OBJ-VRAM 容量を指定します。

0	128K バイト（先頭キャラクターネーム境界 128 バイト）
1	256K バイト（先頭キャラクターネーム境界 256 バイト）

- [d06～d04]：OBJ データ・マッピングモード

- BM[d06～d05]：ビットマップ OBJ データ・マッピングモード

00	横 128 ドットで 2 次元マッピング
01	横 256 ドットで 2 次元マッピング
10	1 次元マッピング
11	設定禁止

2 次元マッピング時は、OBJ-VRAM 領域拡張フラグは無視されます。

よって、OAM の OBJ アトリビュート 2 で指定する 10 ビットのキャラクターネームで指定できるアドレスの範囲で OBJ-VRAM が参照されます。

1 次元マッピング時には、OBJ-VRAM 領域拡張フラグにより容量を設定できます。

6.3.4.1 ビットマップ OBJ データ

ビットマップ OBJ データ

15	14	10	9	8	7	5	4	0
A	BLUE			GREEN			RED	
α_{BMP}	P0							

● A[d15] : α_{BMP}

α_{BMP} 値は OBJ の透明度 α の要素で、 $\alpha = \alpha_{\text{BMP}} \times (\alpha_{\text{OAM}} + 1)$ となっています。

α_{OAM} は OBJ アトリビュート 2 で設定してください（「OAM データフォーマット」参照）。

OBJ 表示

P0							

6.3.4.2 BG とのブレンディング

ビットマップ OBJ は半透明 OBJ と同様に、第 2 対象面の BG とブレンディング表示することができます。

$\alpha_{\text{OAM}}=0$ の場合、OBJ は全領域が透明となり、描画されません。

α_{OAM} が 0 以外の場合は、下記式に従ってブレンディング表示されます。

$$C = \frac{C_{\text{OBJ}} \times \alpha + C_{\text{BG}} \times (16 - \alpha)}{16}$$

$\alpha = \alpha_{\text{BMP}} \times (\alpha_{\text{OAM}} + 1)$ とする。

α_{BMP} はビットマップ OBJ データ、 α_{OAM} は OAM の OBJ アトリビュート 2 で設定した値です。

C : ブレンディング結果のカラー（計算結果の小数点以下は四捨五入されます）

C_{OBJ} : 第 1 対象面のビットマップ OBJ カラー

C_{BG} : 第 2 対象面の BG カラー

6.3.4.3 ビットマップ OBJ データ・マッピングモード

6.3.4.3.1 横 128 ドットで 2 次元マッピング

横 128 ドット時のビットマップ OBJ データの VRAM 上での 2 次元マップは図 6-28 の通りです。

	ドット 0	1	2	3		125	126	127
ライン 0	0h	2h	4h	6h		FAh	FCh	FEh
1	100h	102h	104h	106h		1FAh	1FCh	1FEh
2	200h	202h					2FCh	2FEh
3	300h	302h					3FCh	3FEh
4	400h							4FEh

図 6-28 ビットマップ OBJ データの VRAM 2 次元マップ (横 128 ドット時)

キャラクタネームはビットマップデータの 8×8 ドット (128 バイト分) を 1 単位とします。
 キャラクタネームの VRAM 上での 2 次元イメージマップは図 6-29 の通りです。

ドット ライン	0～7	8～15	16～23	24～31		104～ 111	112～ 119	120～ 127
0～7	0h	1h	2h	3h		Dh	Eh	Fh
8～15	10h	11h	12h	13h		1Dh	1Eh	1Fh
16～23	20h	21h	22h	23h		2Dh	2Eh	2Fh
24～31	30h	31h	32h	33h		3Dh	3Eh	3Fh
32～39	40h	41h	42h	43h		4Dh	4Eh	4Fh

図 6-29 キャラクタネームの VRAM 2 次元イメージマップ

6.3.4.3.2 横 256 ドットで 2 次元マッピング

横 256 ドット時のビットマップ OBJ データの VRAM 上での 2 次元マップは図 6-30 の通りです。

	ドット 0	1	2	3		253	254	255
ライン 0	0h	2h	4h	6h		1FAh	1FCh	1FEh
1	200h	202h	204h	206h		3FAh	3FCh	3FEh
2	400h	402h					5FCh	5FEh
3	600h	602h					7FCh	7FEh
4	800h							9FEh

図 6-30 ビットマップ OBJ データの VRAM 2 次元マップ (横 256 ドット時)

キャラクタネームはビットマップデータの 8×8 ドット (128 バイト分) を 1 単位とします。

キャラクタネームの VRAM 上での 2 次元イメージマップは図 6-31 の通りです。

ドット ライン	0~7	8~15	16~23	24~31		232~ 239	240~ 247	248~ 255
0~7	0h	1h	2h	3h		1Dh	1Eh	1Fh
8~15	20h	21h	22h	23h		3Dh	3Eh	3Fh
16~23	40h	41h	42h	43h		5Dh	5Eh	5Fh
24~31	60h	61h	62h	63h		7Dh	7Eh	7Fh
32~39	80h	81h	82h	83h		9Dh	9Eh	9Fh

図 6-31 キャラクタネームの VRAM 2 次元イメージマップ

6.3.4.3.3 1次元マッピング

キャラクターネーム

2D グラフィックスエンジン A では、表示コントロールレジスタ「DISPCNT」における「ビットマップ OBJ の VRAM 拡張フラグ」によって、キャラクターネームで指定可能な OBJ-VRAM の範囲およびキャラクターネーム境界は表 6-14 のように変わります。

2D グラフィックスエンジン B では、VRAM 割り当て制限のために、OBJ-VRAM の指定可能範囲が 128K バイト、キャラクターネーム境界は 128 バイトに固定されます。

ビットマップ OBJ の VRAM 拡張フラグ	OBJ-VRAM の指定可能範囲	キャラクターネーム境界
0	128K バイト	128 バイト
1	256K バイト	256 バイト

表 6-14 キャラクターネーム境界

（「ビットマップ OBJ の VRAM 拡張フラグ」が 0 の場合の例）

OBJ アトリビュート 2 の「先頭キャラクターネーム」が 4Ch だった場合、4Ch×128 バイト = 0x2600 番地から定義されているビットマップ OBJ データが参照されます。

ビットマップ OBJ データの 1 次元 VRAM マッピング

キャラクターネーム境界の先頭アドレスから、キャラクタのサイズ分のビットマップデータとしてマッピングしてください。

8×8 ドット単位ではありません。

8×8 ドット、16×16 ドットキャラクタ時の 1 次元マップは図 6-32、図 6-33 の通りとなります。

なお、図中の“C+xxh”はキャラクターネーム境界の先頭アドレスからのオフセットを示します。

	0	1	2	3	4	5	6	7
0	C+0h	C+2h	C+4h	C+6h	C+8h	C+Ah	C+Ch	C+Eh
1	C+10h	C+12h	C+14h	C+16h	C+18h	C+1Ah	C+1Ch	C+1Eh
2	C+20h							C+2Eh
3	C+30h							C+3Eh
4	C+40h							C+4Eh
5	C+50h							C+5Eh
6	C+60h	C+62h	C+64h	C+66h	C+68h	C+6Ah	C+6Ch	C+6Eh
7	C+70h	C+72h	C+74h	C+76h	C+78h	C+7Ah	C+7Ch	C+7Eh

図 6-32 8×8 ドットキャラクタ時の VRAM 1 次元マップ

	0	1	2		13	14	15
0	C+0h	C+2h	C+4h		C+1Ah	C+1Ch	C+1Eh
1	C+20h	C+22h	C+24h		C+3Ah	C+3Ch	C+3Eh
2	C+40h						C+5Eh
13	C+1A0h						C+1BEh
14	C+1C0h	C+1C2h	C+1C4h		C+1DAh	C+1DCh	C+1DEh
15	C+1E0h	C+1E2h	C+1E4h		C+1FAh	C+1FCh	C+1FEh

図 6-33 16×16 ドットキャラクタ時の VRAM 1 次元マップ

6.4 バックドロップ

LCD に表示される 2D グラフィックスは、OBJ、BG、バックドロップから構成されます。

OBJ は比較的小さい画像ですが、多数表示することができ、主に画面上を動き回るキャラクタを表示させるのに用います。

BG は、OBJ と同等の機能を持っていますが、面積が広くメモリの消費量が大きいため表示できる面数が少なくなっています。BG は、常に表示させておきたいものや背景など大きい画像を表示させるのに用います。

TWL では、LCD 上で表示する OBJ や BG がない領域は単一色で塗りつぶされます。

これは、図 6-34 のように常に最背面に位置する単一色の面があるという概念として表され、この面をバックドロップと呼びます。バックドロップには OBJ や BG のような機能はなく、単一色で塗りつぶされただけの面です。

この色はパレットを操作することで変更可能です（「カラーパレット」章参照）。

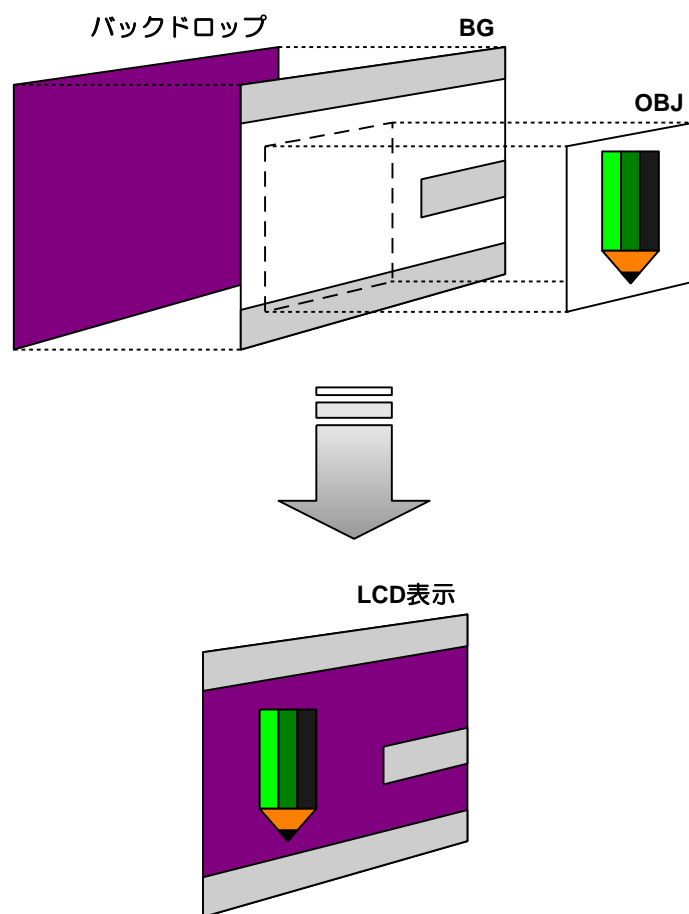


図 6-34 バックドロップ概念図

6.5 カラーパレット

TWL には、標準で BG および OBJ パレット専用割り当てられた RAM（パレット RAM）があります。そして、ここに格納されたデータを標準パレットと呼びます。

標準パレットだけを使用しても OBJ や BG を表示することができますが、拡張パレットを使用することで 256 色×16 パレットを使用でき、表現を高めることができます。拡張パレットは、RAM コントロールレジスタで VRAM を割り当て、DISPCNT レジスタ（2D グラフィックスエンジン B では DB_DISPCNT レジスタ）で拡張パレットイネーブルフラグをイネーブルに設定することで使用可能となります。

6.5.1 標準パレット

2D グラフィックスエンジン A と 2D グラフィックスエンジン B には OBJ と BG それぞれに標準パレット RAM が割り当てられています。

各パレットのカラー 0 は設定に関わらず強制的に透明色として扱われます。

バックドロップ面は、BG パレットの先頭に設定されたカラー（パレット 0 のカラー 0）が使用されます。

なお、標準パレット RAM は 2D グラフィックスエンジン内にあるためパワーコントロールレジスタ（POWCNT）でそれぞれの 2D グラフィックスエンジンがイネーブルになっていなければ書き込むことができません。

標準パレット RAM のアドレスは図 6-35、16 色×16 パレット時のカラー指定は図 6-36、256 色×1 パレット時のカラー指定は図 6-37 の通りです。

0x050003FF	OBJ パレット RAM (512 バイト)
0x05000200	
0x050001FF	BG パレット RAM (512 バイト)
0x05000000	

図 6-35 標準パレット RAM のアドレス (2D グラフィックスエンジン B は+0x400)

パレット先頭	パレット 15	カラー 15

	パレット 2	カラー 2
	パレット 1	カラー 1
	パレット 0	カラー 0

図 6-36 16 色×16 パレット

パレット先頭	パレット	カラー 255
		...
		カラー 2
		カラー 1
		カラー 0

図 6-37 256 色×1 パレット

カラーデータのフォーマットを以下に示します。

カラーデータフォーマット

15	14	10	9	8	7	5	4	0
GREEN	BLUE			GREEN			RED	
カラーデータ								

注意事項

カラーデータの 15 ビット目は GREEN の最下位ビットとして使用されます。BLUE と RED の最下位ビットは 0 で拡張され、RGB 各 6 ビットで計算されます。

6.5.2 拡張パレット

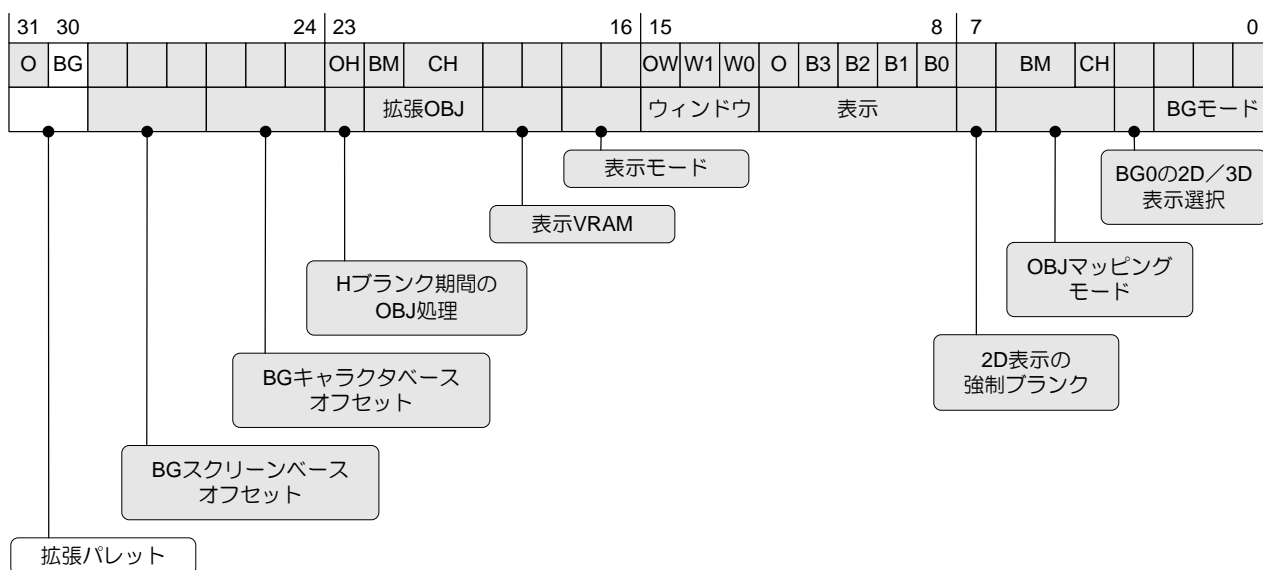
DISPCNT レジスタ（2D グラフィックスエンジン B では DB_DISPCNT レジスタ）の拡張パレットイネーブルフラグを設定し、RAM バンクコントロールレジスタを設定すると 256 色×16 パレット（8K バイト）の VRAM を、OBJ と各 BG 面それぞれに割り当てることができます。

割り当てた場合、パレットスロットは CPU バスにマッピングされません。パレットデータを書き換える場合は、一旦 LCDC に割り当てする必要があります。

6.5.2.1 BG 拡張パレット

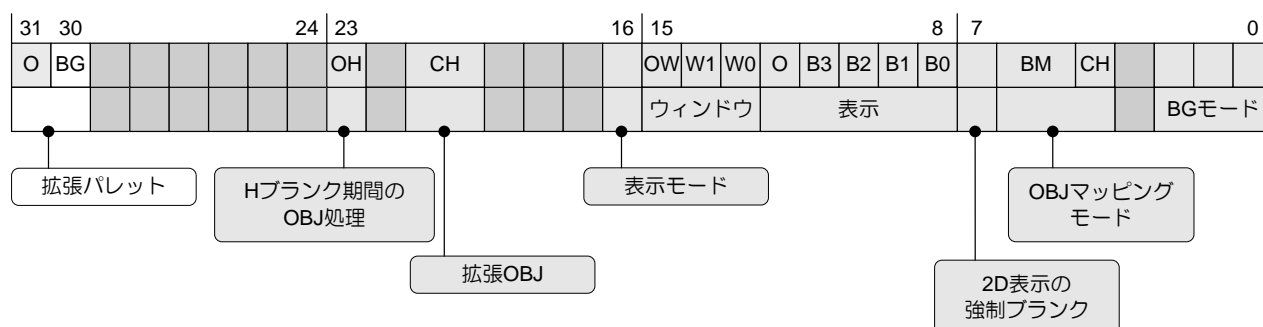
表示コントロールレジスタ（2D グラフィックスエンジン A）

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



表示コントロールレジスタ 1（2D グラフィックスエンジン B）

名称 DB_DISPCNT アドレス 0x04001000 属性 R/W 初期値 0x00000000



- [d31,d30]：拡張パレットイネーブルフラグ
 - BG [d30]：BG 拡張パレット
- 256 色×16 パレットで表示可能な BG 面において有効なフラグです。

0	ディセーブル (256 色×1 パレット)
1	イネーブル (256 色×16 パレット)

BG 拡張パレットがイネーブル時であっても、256 色×16 パレットに対応していない BG 面では常に標準パレットが使用されます。また、バックドロップ面は常に標準パレットのカラー0 が使用されます。

BG 拡張パレットを使用する場合には、VRAM を BG 拡張パレットスロットに割り当てる必要があります。VRAM を拡張パレットスロットに割り当てる方法については、RAM バンクコントロールレジスタを参照してください。

BG 拡張パレットスロット

BG 拡張パレットはスロット 0～3 の最大 32K バイト割り当てることができます。
BG0 は BG0 コントロールレジスタによってスロット 0 か 2 のどちらを使用するか選択でき、BG1 は BG1 コントロールレジスタによってスロット 1 か 3 のどちらを使用するか選択できます。
BG2 はスロット 2 のみ使用でき、BG3 はスロット 3 のみ使用できます。
よって BG0 をスロット 0 に、BG1 をスロット 1 に設定すれば、BG 面毎に独立して拡張パレットを使用できます。また、スロット 2 および 3 を全 BG 面で共有すれば、BG 拡張パレットは 16K バイトに節約できます。

各パレットのカラー0 は設定に関わらず強制的に透明色として扱われます。
バックドロップ面は、標準 BG パレットの先頭に設定されたカラー（パレット 0 のカラー0）が使用されます。

BG 拡張パレットスロットのメモリマップは図 6-38 の通りです。

0x00008000	
0x00006000	スロット 3
0x00004000	スロット 2
0x00002000	スロット 1
0x00000000	スロット 0

図 6-38 BG 拡張パレットスロット メモリマップ

BG タイプ別の使用可能パレット一覧は表 6-15 の通りです。

カテゴリ	BG タイプ		色／ パレット	BG 面	使用可能なパレット領域				
					標準	拡張パレットスロット			
					0	1	2	3	
キャラクタ BG	テキスト BG		16／16	BG0～3	○				
			256／16	BG0	○	○		○	
				BG1	○		○		○
				BG2	○			○	
	アフィン BG		256／1	BG2～3	○				
	拡張 BG	256 色×16 パレット BG	256／16	BG2	○			○	
				BG3	○				○
ビットマップ BG	拡張 BG	256 色	256／1	BG2～3	○				
		ダイレクトカラー	32,768	BG2～3					
		大画面 256 色		256／1	BG2	○			

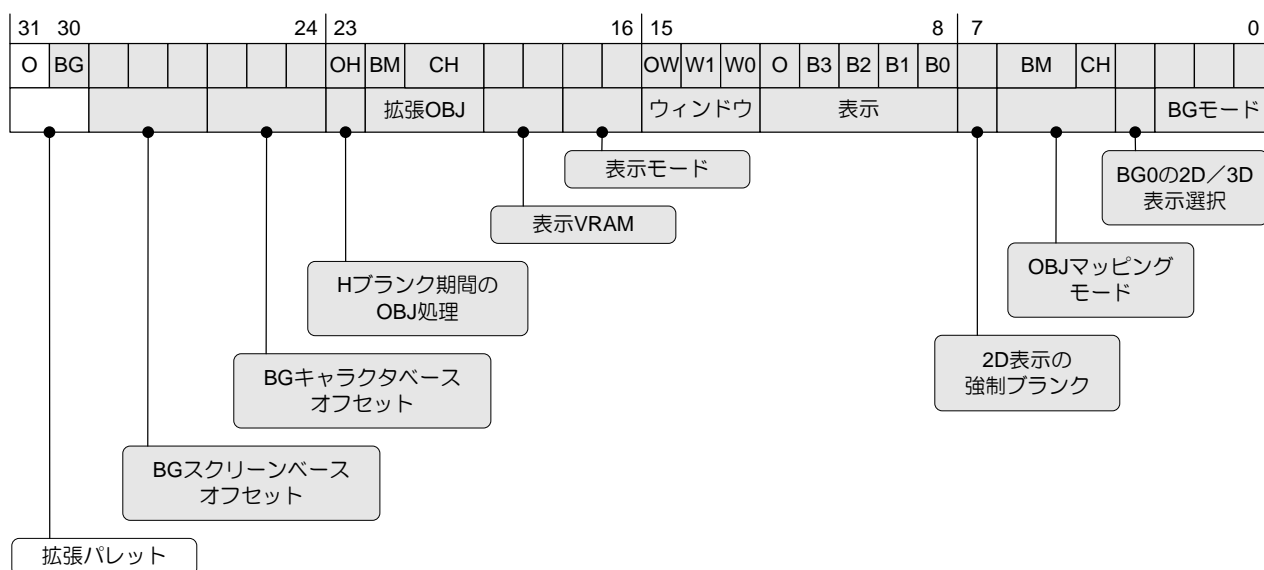
表 6-15 BG タイプ別の使用可能パレット一覧

表 6-15 の通り、BG0 および BG1 については拡張パレットのスロット No.を選択することができます。
2D グラフィックスエンジン B では BG タイプに大画面 256 色ビットマップ BG を設定できません。
選択方法は、「BG コントロール」章の BG コントロールレジスタを参照してください。

6.5.2.2 OBJ 拡張パレット

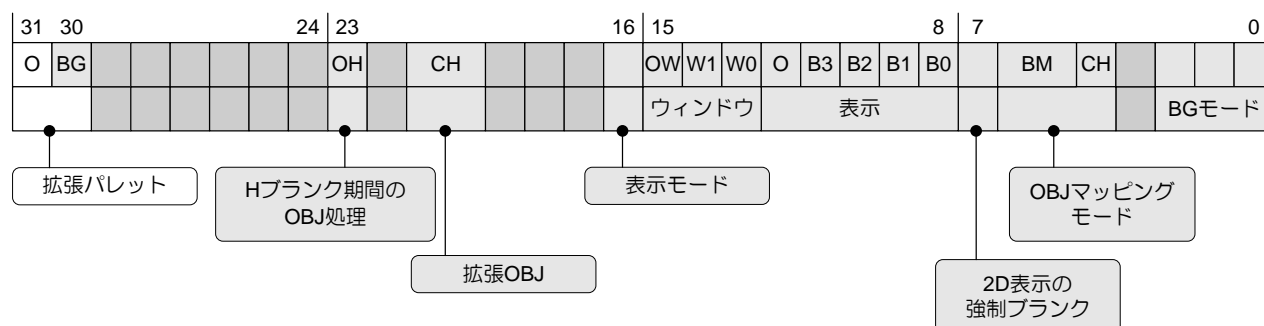
表示コントロールレジスタ (2D グラフィックスエンジン A)

名称 DISPCNT アドレス 0x04000000 属性 R/W 初期値 0x00000000



表示コントロールレジスタ 1 (2D グラフィックスエンジン B)

名称 DB_DISPCNT アドレス 0x04001000 属性 R/W 初期値 0x00000000



- [d31,d30] : 拡張パレットイネーブルフラグ

- O [d31] : OBJ 拡張パレット

0	ディセーブル (256 色×1 パレット)
1	イネーブル (256 色×16 パレット)

OBJ 拡張パレットがイネーブル時であっても、16 色モード OBJ では常に標準パレット (パレット RAM) が使用されます。それぞれのパレットにおける詳細については、「カラーパレット」章を参照してください。

OBJ 拡張パレットを使用する場合には、VRAM を OBJ 拡張パレットスロットに割り当てる必要があります。VRAM を OBJ 拡張パレットスロットに割り当てる方法については、RAM バンクコントロールレジスタを参照してください。

OBJ 拡張パレットスロット

OBJ 拡張パレットには 16K バイトの VRAM を割り当てますが、拡張パレットとしては 8K バイトしか使用しません。
図 6-39 のように、スロット 0 のみ拡張パレットとして使われ、スロット 1 は無効となります。

各パレットのカラー 0 は設定に関わらず強制的に透明色として扱われます。
バックドロップ面は、標準 BG パレットの先頭に設定されたカラー（パレット 0 のカラー 0）が使用されます。

0x00004000	
0x00002000	スロット 1（無効領域）
0x00000000	スロット 0

図 6-39 OBJ 拡張パレットスロット メモリマップ

6.6 ウィンドウ

ウィンドウ機能により、BG や OBJ の各面が表示される領域を限定したり、カラー特殊効果が行われる領域を限定することができます。TWL はウィンドウ 0, 1 および OBJ ウィンドウの 3 種類のウィンドウを使用できます。

(OBJ ウィンドウの設定については「OBJ」章の OBJ アトリビュート 0 を参照してください)

ウィンドウ内側コントロールレジスタ

(2D_A) 名称 WININ アドレス 0x04000048 属性 R/W 初期値 0x0000

(2D_B) 名称 DB_WININ アドレス 0x04001048 属性 R/W 初期値 0x0000

ウィンドウ 1 内側								ウィンドウ 0 内側							
15	13		12	8				7	5		4	0			
		EFCT	OBJ	BG3	BG2	BG1	BG0			EFCT	OBJ	BG3	BG2	BG1	BG0

ウィンドウ外側および OBJ ウィンドウ内側コントロールレジスタ

(2D_A) 名称 WINOUT アドレス 0x0400004A 属性 R/W 初期値 0x0000

(2D_B) 名称 DB_WINOUT アドレス 0x0400104A 属性 R/W 初期値 0x0000

(ED_0) 名称 ED_WINDOW1 形式 6x6 16bit 属性 RW 初期値 0x0000																
15		13		12		8		7		5		4		0		
			EFCT	OBJ	BG3	BG2	BG1	BG0			EFCT	OBJ	BG3	BG2	BG1	BG0
OBJ ウィンドウ内側								ウィンドウ (0, 1 および OBJ ウィンドウ) の外側								

- [d13], [d05] : EFCT : カラー特殊効果イネーブルフラグ

0	ディセーブル
1	イネーブル

- OBJ, BG3～0[d12～d08], [d04～d00] : 表示イネーブルフラグ

0	非表示
1	表示

ウィンドウの外側の表示コントロールは、ウィンドウ 0, 1 および OBJ ウィンドウのいずれかが表示されているときに有効です。

ウィンドウ位置設定レジスタ

(2D_A) 名称 WINxH (x=0, 1) アドレス 0x04000040, 0x04000042 属性 W 初期値 0x0000

(2D_B) 名称 DB_WINxH (x=0, 1) アドレス 0x04001040, 0x04001042 属性 W 初期値 0x0000

15	8							7	0						
ウィンドウ左上 x 座標								ウィンドウ右下 x 座標							

- [d15～d08] : ウィンドウ左上 x 座標
0～255 の範囲で設定してください。

- [d07～d00] : ウィンドウ右下 x 座標
0～255 の範囲で設定してください。

(2D_A) 名称 WINxV (x=0, 1) アドレス 0x04000044, 0x04000046 属性 W 初期値 0x0000

(2D_B) 名称 DB_WINxV (x=0, 1) アドレス 0x04001044, 0x04001046 属性 W 初期値 0x0000

15								8							7													0	
ウィンドウ左上 y 座標																ウィンドウ右下 y 座標													

- [d15～d08]：ウィンドウ左上 y 座標
0～191 の範囲で設定してください。
- [d07～d00]：ウィンドウ右下 y 座標
0～192 の範囲で設定してください。

ウィンドウの範囲

ウィンドウの左上座標を (lx, ly)、右下座標を (rx, ry) とすると、LCD 座標 (0,0) ～ (255,191) におけるウィンドウの範囲は (lx, ly) ～ (rx-1, ry-1) になります。

ウィンドウを LCD の右端に寄せるには右下座標の X 座標に 0 を設定し、左端に寄せるには左上座標の X 座標に 0 を設定することになります。しかし、どちらの X 座標も 0 に設定した場合はウィンドウが表示されませんので、ウィンドウの幅を LCD 幅一杯にすることができません。LCD 幅一杯のウィンドウを出すには、もう一枚ウィンドウ（または OBJ ウィンドウ）を使用してください。

※TWL モードのシステム設定により 2D グラフィックスエンジンの回路修正を有効にした場合は、左上と右下の X 座標を 0 にすることでウィンドウ幅を LCD 幅一杯にすることができます（SCFG_EXT レジスタで設定します）。

ウィンドウの形

ウィンドウ 0,1 は矩形の設定しかできませんが、H ブランク期間にウィンドウ位置設定レジスタを書き換えることによって変形させることは可能です（図 6-40 参照）。

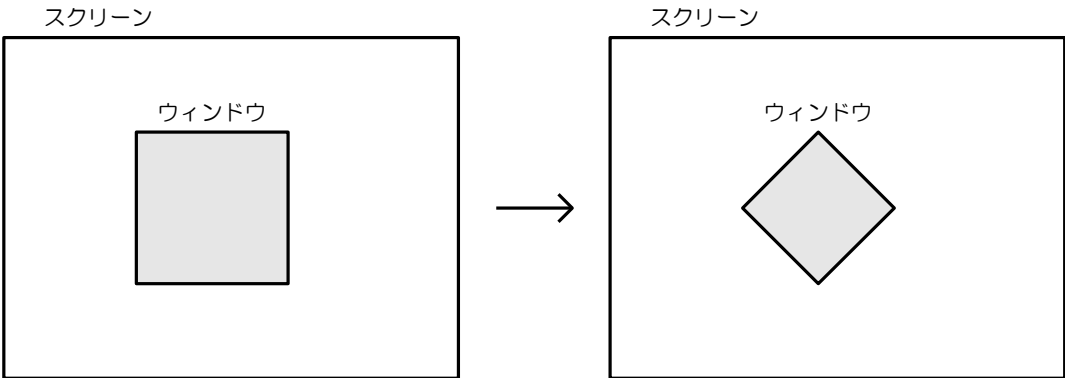


図 6-40 ウィンドウの変形例

6.6.1 ウィンドウの優先順位

図 6-41 のように、ウィンドウ 0 は常にウィンドウ 1 よりも優先表示され、OBJ ウィンドウは最も優先順位が低くなります。

優先順位は変更することができません。

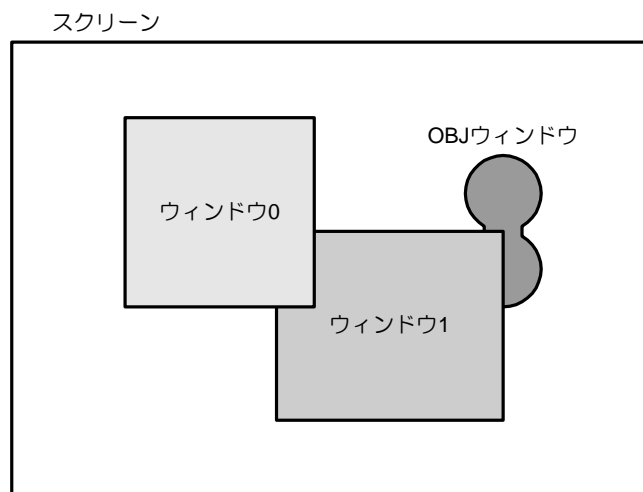


図 6-41 ウィンドウ 0、1 および OBJ ウィンドウの優先順位

ウィンドウに関する注意事項 1

ウィンドウの左上 Y 座標が 0~6 のとき、左上 Y 座標が強制的に 0 になったかのような表示になってしまいます。

この問題を回避するには、以下に示す 2 つの回避方法のうちいずれかを行ってください。

※TWL モードのシステム設定により 2D グラフィックスエンジンの回路修正を有効にした場合は、この問題は回避されます（SCFG_EXT レジスタで設定します）。

回避方法 1：次の 1) と 2) の両方の処理をソフトウェアで行ってください。

- 1) V ブランク処理において、V カウントが 256 に達する前にウィンドウの Y 座標を 7 以上に設定します
- 2) V カウント一致割り込みもしくは H ブランク割り込みで V カウント値を確認し、V カウント値が 262 であることを確認後、ウィンドウの Y 座標に本来の値を設定します。

回避方法 2：次の 1) と 2) の両方の処理をソフトウェアで行ってください。

- 1) V カウント一致割り込みもしくは H ブランク割り込みで V カウント値を確認し、V カウント値が 262 であることを確認後、表示コントロールレジスタのウィンドウの表示／非表示フラグを次のように設定します。
 ウィンドウの左上 Y 座標が 0 の場合 : 1 (ウィンドウ表示)
 ウィンドウの左上 Y 座標が 0 以外の場合 : 0 (ウィンドウ非表示)
- 2) H ブランク割り込みで V カウント値を読み出し、(ウィンドウの左上 Y 座標-1) と比較します。
 等しいことが確認できた場合、H ブランク内において、表示コントロールレジスタのウィンドウ表示／非表示フラグを 1 (ウィンドウ表示) に設定します。

ウィンドウに関する注意事項 2

右下 X 座標を 0 に設定したラインを描画した直後の H ブランクでは、まだ現在のウィンドウ描画が終わっていません。この状態で次のウィンドウ座標を設定してしまうと、次のウィンドウの右下 X 座標までウィンドウ描画が続いてしまうことになります。

これを回避するには、ウィンドウ描画が確実に終了するのを待ってから次のウィンドウ座標を設定する必要があります。右下 X 座標が 0 のとき、ウィンドウ描画が終わるのは、ステータスフラグで H ブランクフラグが 1→0 に変化（もしくは V カウント値が+1）してからシステムクロック（33MHz）換算で 3 クロック後です。このタイミングから、H カウントが次のウィンドウの左上 X 座標に達するまでの間がウィンドウ外になりますので、この期間中に次の座標をセットしてください。なお、H ブランク起動 DMA でウィンドウ座標を設定する場合は、この回避方法を採用することができません。

6.7 カラー特殊効果

OBJ や BG は、 α ブレンディングおよびフェードイン・アウトのカラー特殊効果機能を備えています。また、これらの効果を施す領域をウィンドウにより限定することができます（「ウィンドウ」章参照）。カラー特殊効果を表 6-16 に示します。

カラー特殊効果	効果
α ブレンディング	選択された 2 面に演算を施し、16 段階の半透明処理を行います。 ただし抜きの部分（透明なピクセル）に対して処理は行われません。
輝度アップ/ダウン （フェードイン・アウト）	選択された面に演算を施し、16 段階の輝度変化処理を行います。 ただし抜きの部分（透明なピクセル）に対して処理は行われません。

表 6-16 カラー特殊効果とその効果

カラー特殊効果コントロールレジスタ

(2D_A) 名称 BLDCNT アドレス 0x04000050 属性 R/W 初期値 0x0000

(2D_B) 名称 DB_BLDCNT アドレス 0x04001050 属性 R/W 初期値 0x0000

[ED_B] 名称 ED_EDBON1 中 6 次 6x6 1001000 属性 100W 防刺値 6x6000																
15		13			8			7		6		5		0		
			BD	OBJ	BG3	BG2	BG1	BG0			BD	OBJ	BG3	BG2	BG1	BG0
		第 2 対象面						特殊効果選択		第 1 対象面						

カラー特殊効果は、BLDCNT [DB_BLDCNT] レジスタにより設定しますが、2 面間で処理を行う α ブレンディングにおいては、対象 2 面の優先順位が適切でなければなりません。また、OBJ に関しては OAM にて個別に半透明 OBJ 指定を行い、BLDCNT [DB_BLDCNT] レジスタでは OBJ 全体に対するカラー特殊効果指定を行います。

● [d07～d06]：特殊効果選択

特殊効果選択		種類	カラー特殊効果処理内容
d07	d06		
0	0	特殊効果なし	通常カラー特殊効果処理は行いません。 ただし半透明 OBJ、ビットマップ OBJ、3D 面を描画した BG0 面については、その直後が第 2 対象面であった場合のみ 16 段階の半透明処理（ α ブレンディング）を行います。
0	1	α ブレンディング	第 1 対象面の直後が第 2 対象面の場合のみ 16 段階の半透明処理（ α ブレンディング）を行います。第 1 対象面のバックドロップ面ビットはオフ（[d05]=0）にしてください。 第 1 対象面で OBJ=1 の場合は、OBJ の種類に関わらずすべての OBJ に対して処理が実行されます。OBJ=0 の場合は、半透明 OBJ、ビットマップ OBJ、3D 面を描画した BG0 面の場合のみ処理が実行されます。
1	0	輝度アップ※	第 1 対象面に輝度変更アップ（徐々に輝度を上げる）が実行されます。 第 1 対象面指定で OBJ=1 の場合は、ノーマル OBJ に対してのみ輝度変更アップ処理が実行されます。 半透明 OBJ、ビットマップ OBJ、3D 面を描画した BG0 面については、その直後が第 2 対象面であった場合は常に α ブレンディング処理が実行されます。
1	1	輝度ダウン※	第 1 対象面に輝度変更ダウン（徐々に輝度を下げる）が実行されます。 第 1 対象面指定で OBJ=1 の場合は、ノーマル OBJ に対してのみ輝度変更ダウン処理が実行されます。 半透明 OBJ、ビットマップ OBJ、3D 面を描画した BG0 面については、その直後が第 2 対象面であった場合は常に α ブレンディング処理が実行されます。

表 6-17 特殊効果とその処理内容

※表 6-17 のように、「半透明 OBJ」、「ビットマップ OBJ」、「3D 面を描画した BG0 面」については、その直後に第 2 対象面が設定されていた場合は特殊効果の選択に関わらず常に第 2 対象面との α ブレンディング処理が行われます。

2) 輝度アップ、輝度ダウン

カラー特殊効果・輝度変更係数レジスタ

(2D_A) 名称 BLDY アドレス 0x04000054 属性 W 初期値 0x0000

(2D_B) 名称 DB_BLDY アドレス 0x04001054 属性 W 初期値 0x0000

15							8	7						0
														EVY

輝度変更処理に使用する係数を、BLDY [DB_BLDY] レジスタの EVY で設定します。

EVY は 1/16 倍され、ピクセル色の係数として下式の計算に使われます (EVY の値が 16 以上のときは 16 とみなされます)。

● 輝度変更アップ演算

$$\text{表示色 (R)} = \text{第 1 ピクセル (R)} + (31 - \text{第 1 ピクセル (R)}) \times (\text{EVY} / 16)$$

$$\text{表示色 (G)} = \text{第 1 ピクセル (G)} + (31 - \text{第 1 ピクセル (G)}) \times (\text{EVY} / 16)$$

$$\text{表示色 (B)} = \text{第 1 ピクセル (B)} + (31 - \text{第 1 ピクセル (B)}) \times (\text{EVY} / 16)$$

● 輝度変更ダウン演算

$$\text{表示色 (R)} = \text{第 1 ピクセル (R)} - \text{第 1 ピクセル (R)} \times (\text{EVY} / 16)$$

$$\text{表示色 (G)} = \text{第 1 ピクセル (G)} - \text{第 1 ピクセル (G)} \times (\text{EVY} / 16)$$

$$\text{表示色 (B)} = \text{第 1 ピクセル (B)} - \text{第 1 ピクセル (B)} \times (\text{EVY} / 16)$$

輝度変更演算結果の小数点以下は四捨五入されます。

6.8 モザイク

モザイクのサイズは、MOSAIC レジスタ（2D グラフィックスエンジン B では DB_MOSAIC レジスタ）にて設定します。
BG 毎の ON/OFF は、BG コントロールレジスタのモザイクフラグにて行います。

モザイクサイズレジスタ

(2D_A) 名称 MOSAIC アドレス 0x0400004C 属性 W 初期値 0x0000
(2D_B) 名称 DB_MOSAIC アドレス 0x0400104C 属性 W 初期値 0x0000

15	8	7	0
V サイズ	H サイズ	V サイズ	H サイズ
OBJ モザイクサイズ		BG モザイクサイズ	

モザイクサイズ値は、通常時の表示ドットに対して何ドット大きいドットを表示するか、を指定します。
モザイクの表示では、画面の最も左上のドットからモザイクサイズおきのドットが採用され、それ以外のドットはモザイクによって上書きされます（図 6-43 参照）。
つまり、モザイクサイズ値が 0 であれば、モザイク ON であっても通常表示となります。

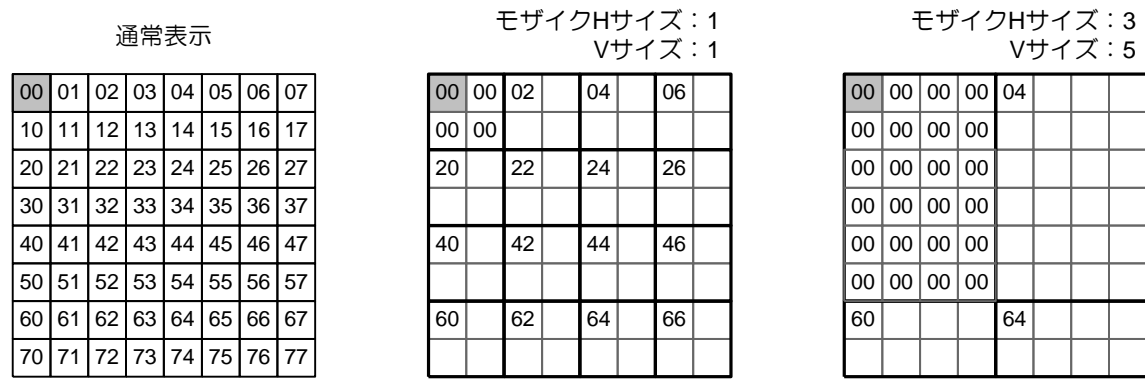


図 6-43 モザイクサイズによる表示の変化

6.9 表示優先順位

● BG 間の表示優先順位

BG 同士の優先順位を BG コントロールレジスタで 4 段階に任意設定可能です。

同じ優先指定のとき、BG ナンバーの若いものが優先されます。

バックドロップ面は、常に優先順位が最低に固定されています。

● OBJ 間の表示優先順位

OBJ 同士の優先順位を OAM の OBJ アトリビュートデータで 4 段階に任意設定可能です。

同じ優先指定のとき、OBJ ナンバーの若いものが優先されます。

● BG – OBJ 間の表示優先順位

優先順位番号が同じであれば、OBJ の方が優先されます（図 6-44 参照）。

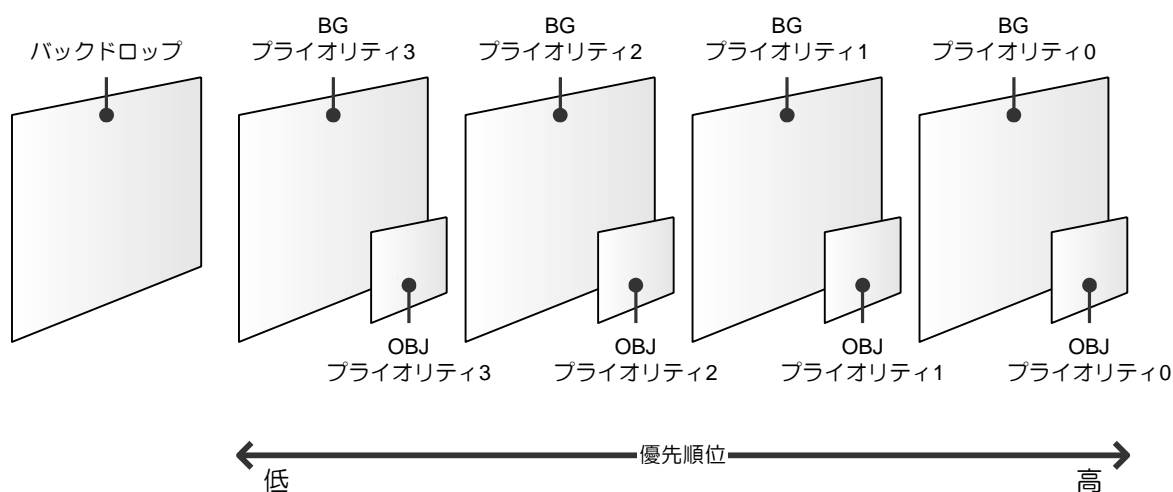


図 6-44 表示優先順位

7. 3D グラフィックス

TWL の 3D グラフィックスエンジンはシステム設定によって、NITRO にあった不具合を修正できること以外には特に変更はありません。修正される機能については、「ジオメトリエンジン」、「レンダリングエンジン」または「NITRO との違い」を参照してください。

3D グラフィックス描画のためのハードウェア・ブロック図を図 7-1 に示します。

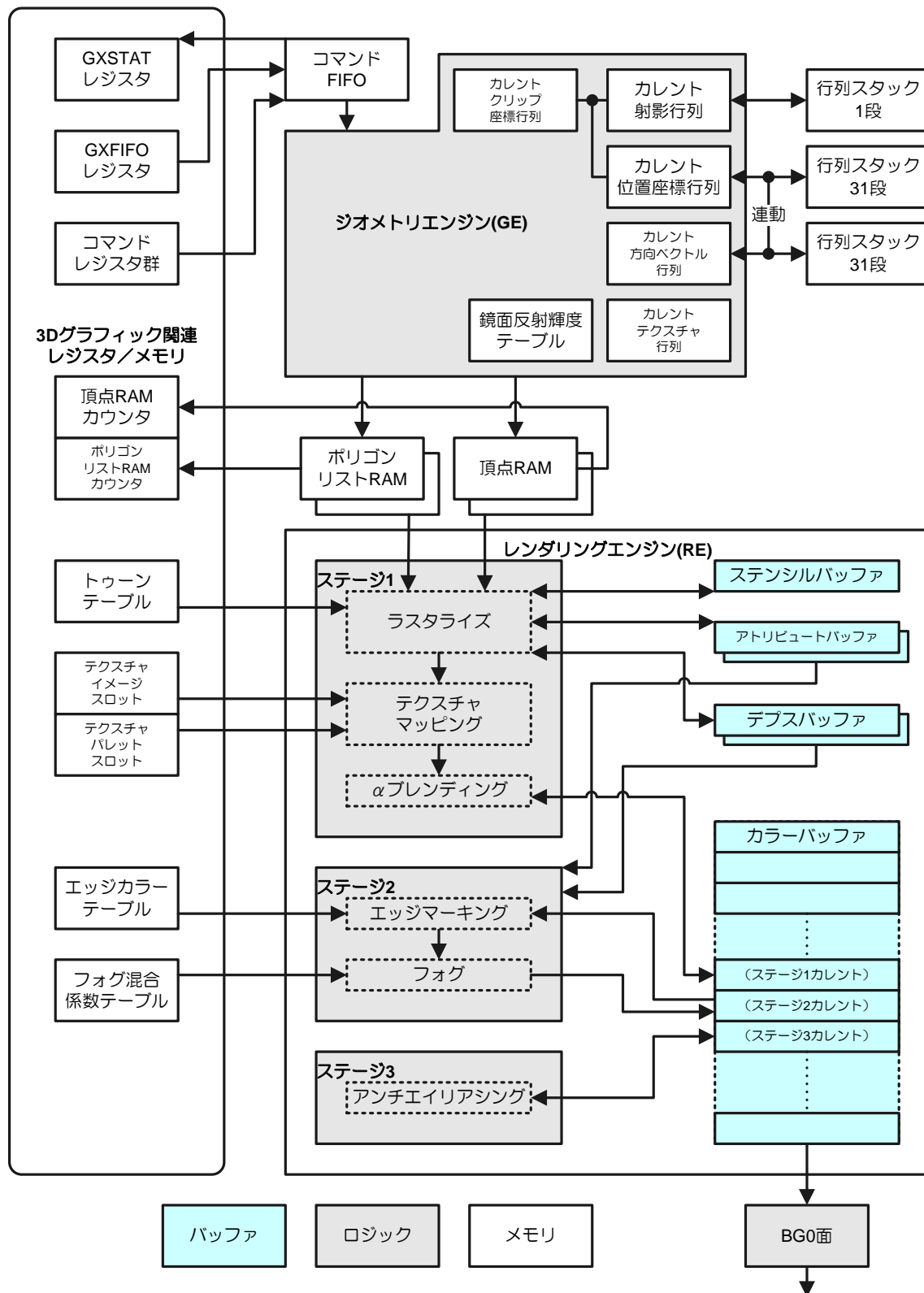


図 7-1 3D グラフィックスのハードウェア ブロック図

- ポリゴンリスト RAM、頂点 RAM
ジオメトリエンジンからレンダリングエンジンに渡すデータは、ポリゴンリスト RAM および頂点 RAM に格納されます。
ポリゴンリスト RAM および頂点 RAM の容量は表 7-1 の通りです。

RAM	容量
ポリゴンリスト RAM	2048 ポリゴン
頂点 RAM	6144 頂点

表 7-1 ポリゴンリスト RAM および頂点 RAM の容量

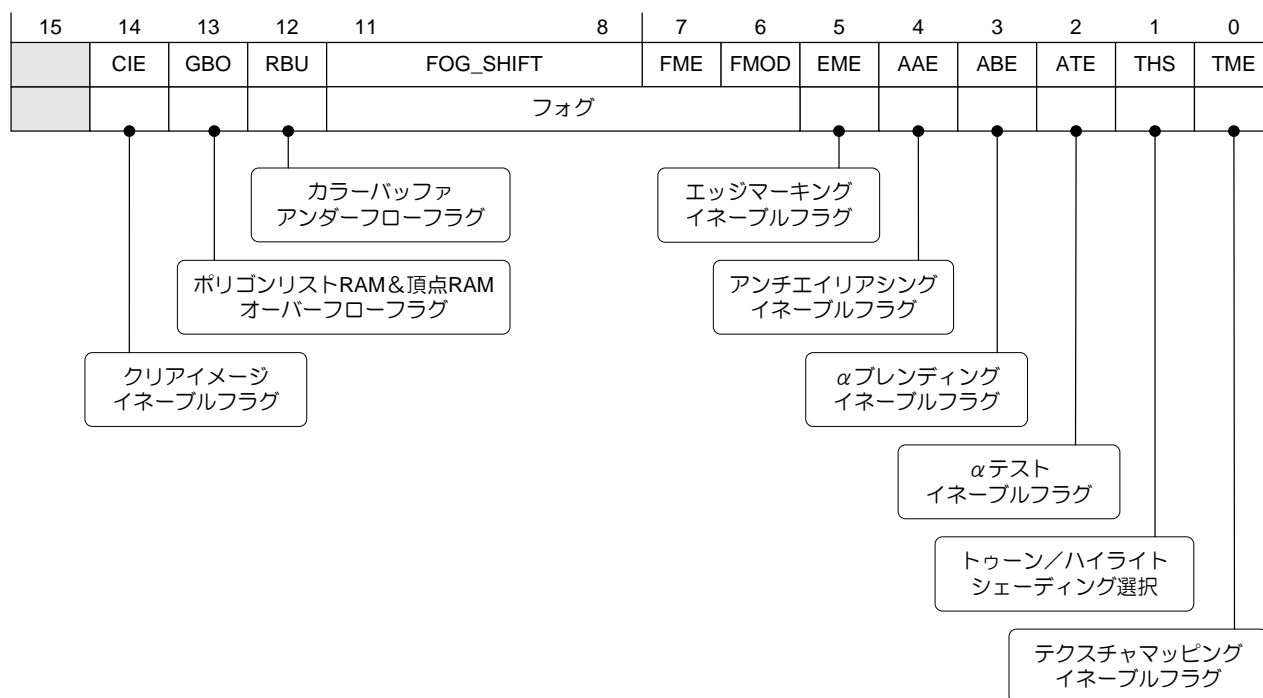
ポリゴンリスト RAM の容量はちょうど三角形ポリゴン 2048 個分ですが、四角形ポリゴンでは 1 ポリゴン当たり 4 つの頂点を持つため、格納できるポリゴン数は少なくなります。
連結四角形ポリゴンであれば、隣接したポリゴン間で頂点が共有されるため、最大で 1706 個まで格納することができます。

- レンダリングエンジン内のバッファ
ステンシルバッファ、アトリビュートバッファ、デプスバッファ、カラーバッファはピクセル毎の情報を格納するメモリです。
図 7-1 中のバッファ 1 ブロックは LCD 水平ライン 256 ドット分のラインバッファを表しています。
詳細については、「レンダリングエンジン」を参照してください。

7.1 3D 表示制御

3D 表示コントロールレジスタ

名称 DISP3DCNT アドレス 0x04000060 属性 R/W 初期値 0x0000



● CIE[d14]：クリアイメージイネーブルフラグ

128K バイトの VRAM を 2 面使用し、クリアカラー、クリア α 、クリアデプス、クリアフォグの各値を設定します。ただしクリア α 値は 1 ビット指定のため、透明／不透明の選択のみ可能です。

使用する VRAM2 面は、RAM コントロールレジスタ 0 でクリアイメージバッファに割り当てておく必要があります。この機能を使用した場合でも、クリアポリゴン ID についてはレジスタの値が使われます。

「レンダリングバッファ初期化」の「クリアイメージによる初期化」を参照してください。

● GBO[d13]：ポリゴンリスト RAM & 頂点 RAM ・オーバーフローフラグ

ジオメトリエンジンが処理したポリゴン数や頂点数が多過ぎ、ポリゴンリスト RAM および頂点リスト RAM がオーバーフローした場合に 1 となります。

一度オーバーフローが発生すると、解消しても 1 のままです。

1 をライトすることでリセットできます。

0	オーバーフローなし
1	オーバーフロー発生

ポリゴンリスト RAM または頂点 RAM がオーバーフローした場合、オーバーフローが起こったポリゴンより後のポリゴンに関してはポリゴンリスト RAM や頂点 RAM に登録されません。

● RBU[d12]：カラーバッファ・アンダーフローフラグ

表示に対してレンダリングが間に合わず、カラーバッファがアンダーフロー（レンダリングエンジンのラインズオーバー）が起こった場合に 1 となります。

一度カラーバッファのアンダーフローが発生すると、解消しても 1 のままです。

1 をライトすることでリセットできます。「レンダリング方式」を参照してください。

0	アンダーフローなし
1	アンダーフロー発生

● [d11~d06] : フォグ

霧がかかったような効果を表現する機能です。

詳細は「レンダリングエンジン」を参照してください。

● FOG_SHIFT[d11~d08] : フォグシフト

フォグで使用するデプス値は 24 ビットのうちの上位 15 ビット（これをフォグデプス値とします）を使用します。フォグテーブルはフォグデプス値の 5 ビットをインデックスにして参照されます。

フォグシフトが 0 のときにはフォグデプス値の d14~d10 がインデックスとして参照され、フォグシフト値が 1 増える毎に参照ビットは 1 ビットずつ右へシフトされていきます。

11~15 の値は設定禁止です。

● FME[d07] : フォグマスタースイネーブルフラグ

0	ディセーブル
1	イネーブル

● FMOD[d06] : フォグモード

1 を選択することによって、3D のオブジェクトが 2D の背景に溶け込んでいくような表現ができます。

0	ピクセルのカラー値と α 値に フォグブレンディング
1	ピクセルの α 値のみに フォグブレンディング

● EME[d05] : エッジマーキングイネーブルフラグ

ポリゴン ID が異なるポリゴンエッジに対して指定色で輪郭線を描画する機能です。

詳細は「レンダリングエンジン」を参照してください。

0	ディセーブル
1	イネーブル

● AAE[d04] : アンチエイリアシングイネーブルフラグ

ポリゴンエッジに対して後ろにあるポリゴンのカラー値と混合する機能です。

詳細は「レンダリングエンジン」を参照してください。

レンダリング結果をキャプチャし、ビットマップ OBJ 等として使う場合はディセーブルにしておいた方が自然な表示となります。

0	ディセーブル
1	イネーブル

● ABE[d03] : α ブレンディングイネーブルフラグ

カラーバッファのカラーとフラグメントカラーをフラグメントの α 値に応じてブレンドします。

詳細は「レンダリングエンジン」を参照してください。

0	ディセーブル
1	イネーブル

● ATE[d02] : α テストイネーブルフラグ

α 値が設定値よりも小さいピクセルの描画をスキップすることができます。

詳細は「レンダリングエンジン」を参照してください。

0	ディセーブル
1	イネーブル

● THS[d01] : トゥーン／ハイライトシェーディング選択

PolygonAttr コマンドでトゥーン／ハイライトシェーディングに指定されたポリゴンのシェーディングモードを選択します。

● TMOD : トゥーン／ハイライトポリゴンのモード

0	トゥーンシェーディング
1	ハイライトシェーディング

● TME[d00] : テクスチャマッピングマスタースイネーブルフラグ

テクスチャマッピングを行うか、行わないか選択できます。

0	ディセーブル
1	イネーブル

7.2 ジオメトリエンジン

TWL モードではシステム設定によって、NITRO で問題の発生していた以下の機能について修正された回路を使用することができます。ただし、NITRO 互換モードでは、注意事項に従って問題を回避する必要があります。

- コマンド FIFO：「ジオメトリコマンド」を参照してください。
- 1 ドットポリゴン：「ポリゴン属性」を参照してください。
- クリッピング機能：「ポリゴン」を参照してください。
- テクスチャイメージパラメータ：「テクスチャマッピング」を参照してください。
- ボックステスト：「テスト」を参照してください。

7.2.1 概要

ジオメトリエンジンの仕様一覧を表 7-2 に示します。

動作周波数	33.514MHz
座標変換	最大 400 万頂点／秒
行列演算	4×4 行列演算および行列スタック
クリッピング	6 面クリッピング
ライティング	ライト：平行光源×4 マテリアル：反射光（拡散反射、鏡面反射、環境反射）、放射光
その他の機能	バックフェイスカリング、1 ドットポリゴン※表示指定、 ボックスカリングテスト、テクスチャ座標変換、 鏡面反射輝度分布調整

※1 ドットポリゴン = ポリゴンを構成する全頂点の座標 (x, y) が同一座標に集約されたポリゴン

表 7-2 ジオメトリエンジンの仕様一覧

7.2.2 座標系

3 次元空間の座標系では、XY 軸に対する Z 軸方向の決め方が右手座標系と左手座標系の 2 通りあります。

基本的に TWL は右手座標系になっています。
ただし、射影行列にて Z 値を反転するため、クリップ座標以降は左手座標系になります。

右手座標系では、X、Y、Z の各軸は図 7-2 のような関係となります。

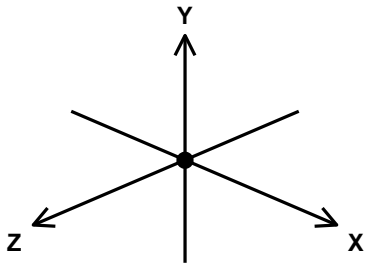


図 7-2 右手座標系

7.2.3 座標変換

OpenGL ではモデルをビュー座標に配置する際、モデルと視界(ビュー)の変換を含めたモデルビュー行列を乗算します。続いて射影行列を乗算し視体積に基づいて、モデルがどのような大きさで見えるか決定します。

TWL では、ハードウェアの負荷軽減のため射影行列と位置座標行列をあらかじめ合成したクリップ座標行列を適用し、一度の座標変換でクリップ座標へ変換しています。

その後、クリップ座標値 (x, y, z, w) を w だけ平行移動した後に $2w$ で除算 (透視除算) して正規化スクリーン座標を求め、ビューポート変換によって BG スクリーン座標にスケーリング変換します。

頂点の法線ベクトルやライトベクトルについては、OpenGL ではモデルビュー行列の逆転置行列を求め変換を行いますが、TWL ではベクトルが正規化されていることを前提に、回転成分のみの行列 (直交行列) を適用した変換を行います。

この変換を「方向ベクトル変換」と呼びます。

OpenGL ではモデルビュー行列を設定するだけで頂点位置座標と方向ベクトルをビュー座標へ変換しますが、TWL では頂点位置座標と方向ベクトルを別々の行列で変換するため、それぞれ「位置座標行列」・「方向ベクトル行列」と定義しています。

TWL での座標変換フロー図を図 7-3 に示します。

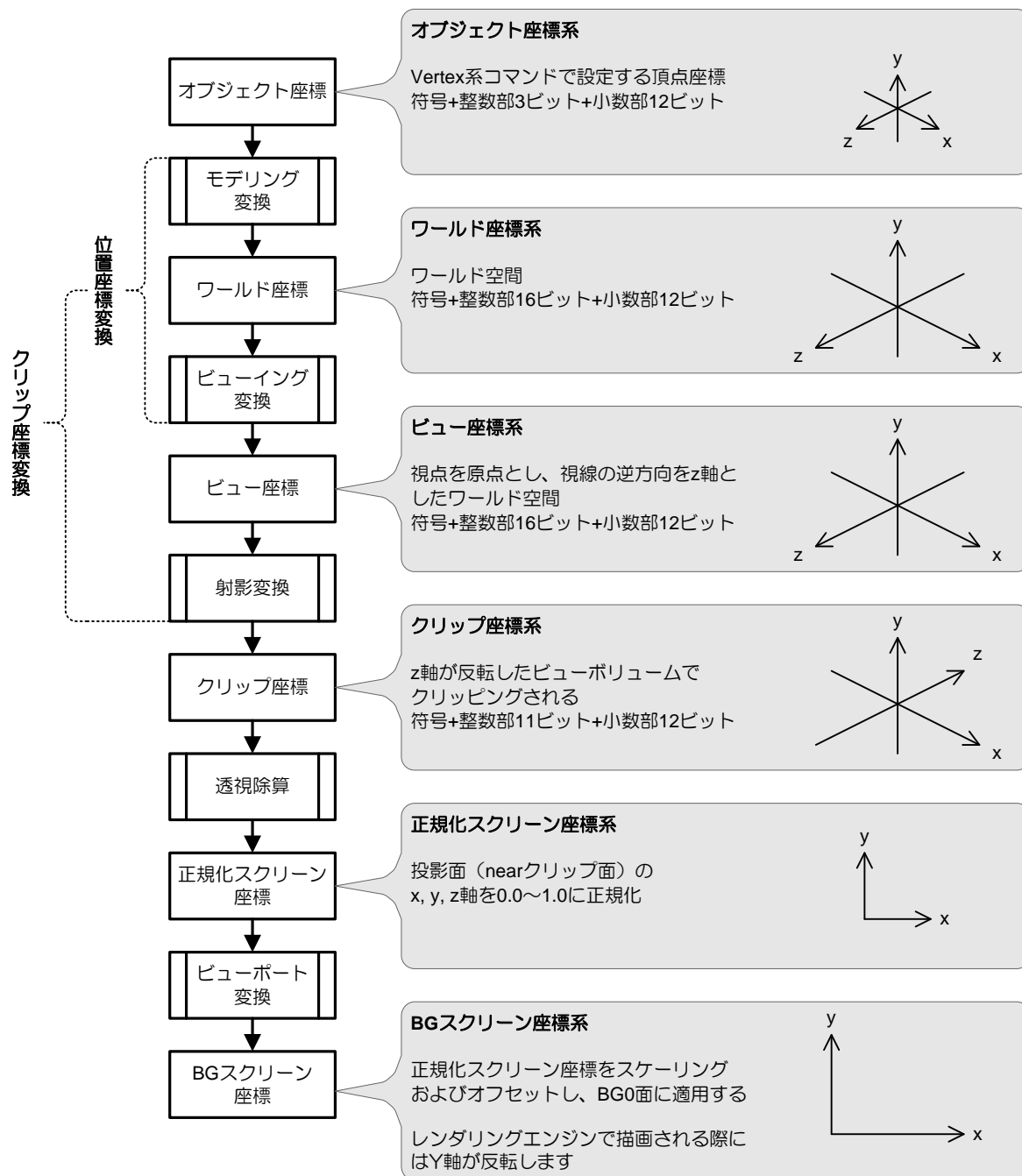


図 7-3 座標変換フロー図

7.2.4 射影変換

視点から見たポリゴンオブジェクトの遠近感をビューボリューム（視体積）により定義します。
透視射影（図 7-4）と正射影（図 7-5）のビューボリュームおよび射影行列は下記ようになります。

1) 透視射影

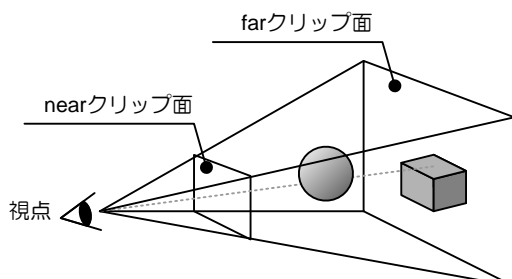


図 7-4 透視射影

2) 正射影

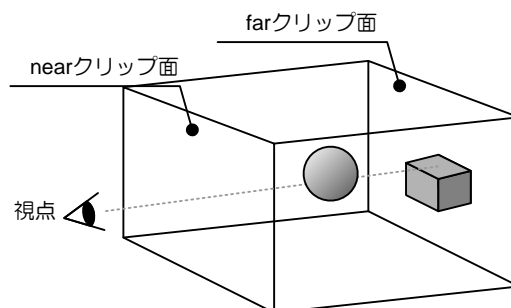


図 7-5 正射影

1-a) 左右非対称の透視射影

$$\text{Frustum} = \begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ \frac{r+l}{r-l} & \frac{t+b}{t-b} & -\frac{f+n}{f-n} & -1 \\ 0 & 0 & -\frac{2fn}{f-n} & 0 \end{bmatrix} \times \text{scaleW}$$

$$\text{Ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{2}{f-n} & 0 \\ -\frac{r+l}{r-l} & -\frac{t+b}{t-b} & -\frac{f+n}{f-n} & 1 \end{bmatrix}$$

1-b) 左右対称の透視射影

$$\text{Perspective} = \begin{bmatrix} \frac{\cos \theta}{\text{asp} \cdot \sin \theta} & 0 & 0 & 0 \\ 0 & \frac{\cos \theta}{\sin \theta} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -1 \\ 0 & 0 & -\frac{2fn}{f-n} & 0 \end{bmatrix} \times \text{scaleW}$$

t: near クリップ面上辺の y 座標

b: near クリップ面下辺の y 座標

r: near クリップ面右辺の x 座標

l: near クリップ面左辺の x 座標

n: 視点から near クリップ面までの距離

f: 視点から far クリップ面までの距離

θ : 縦 (y) 方向の視界角度 (画角) $\div 2$

asp: 縦に対する横の視界の割合 (縦横比: 視界での幅 \div 高さ)

scaleW: ビューボリュームの精度調整パラメータ (クリップ座標空間のスケーリングを変更し、透視除算後の正規化スクリーン座標の精度を向上させるために使用します)

7.2.5 デプスバッファリング

1) 透視射影時

透視射影行列のパラメータは次のように設定されます（要素 p0～p5 の詳細は前述の「射影変換」を参照してください）。

$$M = \begin{bmatrix} p0 & 0 & 0 & 0 \\ 0 & p2 & 0 & 0 \\ p1 & p3 & p4 & -1 \\ 0 & 0 & p5 & 0 \end{bmatrix} \times \text{scaleW}$$

ビュー座標を $(x, y, z, 1)$ とすると、クリップ座標 (x', y', z', w') は次のようになります。

$$x' = (p0 \times x + p1 \times z) \times \text{scaleW}$$

$$y' = (p2 \times y + p3 \times z) \times \text{scaleW}$$

$$z' = (p4 \times z + p5 \times 1) \times \text{scaleW}$$

$$w' = -z \times \text{scaleW}$$

クリップ座標の各成分を w' だけ平行移動した後に $2w'$ で透視除算すると、正規化スクリーン座標 $(x'', y'', z'', 1)$ が求められます。その成分 z'' は下記のようになります。

$$z'' = \frac{z' + w'}{2w'} = \frac{(p4 \times z + p5 - z) \times \text{scaleW}}{-2z \times \text{scaleW}} = \frac{1}{2} - \frac{p4}{2} - \frac{p5}{2z}$$

p4 および p5 は、透視射影行列の要素であるため「射影変換」の 1-a) 行列を適用すると、

$$z'' = \frac{f}{(f - n)} \left(1 + \frac{n}{z}\right) \quad (-f \leq z \leq -\text{near}) \text{ となります。}$$

z バッファリング時には、この z'' に 0x7FFF を乗算した値 z''' がデプス値となります。

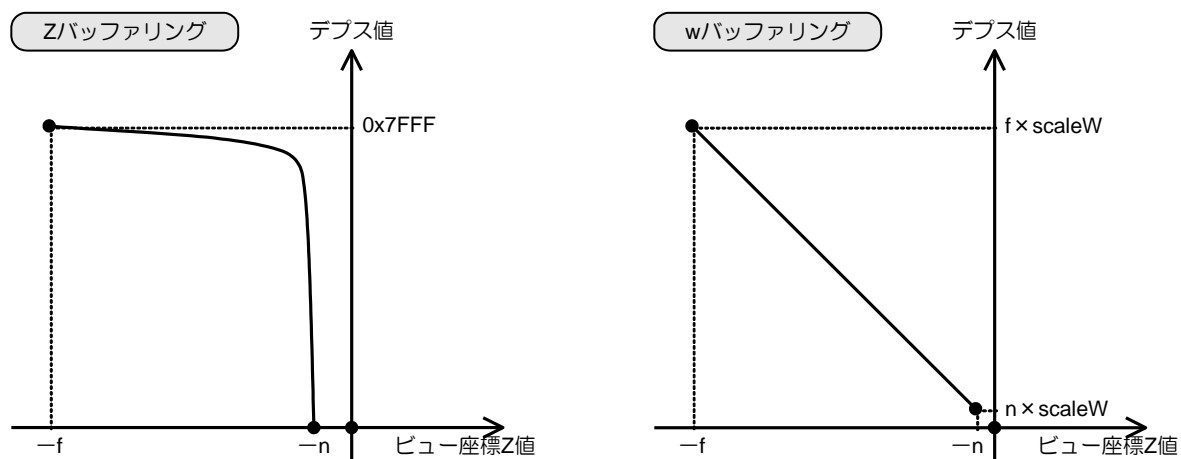
この場合デプス値 z''' がビュー座標 z の逆数に比例するため、ビュー座標空間にて位置座標が視点に近いほど精度が良く、視点から離れるほど精度が悪化することになります。

これは広い空間を表現したい場合に遠方の描画が不正確になりやすいことを意味しています。

この問題を解決するため、TWL ではクリップ座標 w' をデプス値とする「w バッファリング」をサポートしています。

$w' = -z \times \text{scaleW}$ ($-f \leq z \leq -\text{near}$) よりビュー座標 z の定数倍をデプス値とすることになるため、遠方の描画精度が改善されます。

逆に近傍の精度は z バッファリングに比べて悪化することになりますが、視体積空間を広くしたり、scaleW を調節することによって視体積の分解精度をある程度確保すれば、近傍においてもそれほど問題になりません。（図 7-6 参照）



f: 視点から far クリップ面までの距離

n: 視点から near クリップ面までの距離

図 7-6 Z バッファリングと w バッファリング（透視射影時）

2) 正射影時

正射影行列のパラメータは次のように設定されます（要素 p0～p5 の詳細は前述の「射影変換」を参照してください）。

$$M = \begin{bmatrix} p0 & 0 & 0 & 0 \\ 0 & p2 & 0 & 0 \\ 0 & 0 & p4 & 0 \\ p1 & p3 & p5 & 1 \end{bmatrix} \times \text{scaleW}$$

ビュー座標を $(x, y, z, 1)$ とすると、クリップ座標 (x', y', z', w') は次のようになります。

$$x' = (p0 \times x + p1 \times 1) \times \text{scaleW}$$

$$y' = (p2 \times y + p3 \times 1) \times \text{scaleW}$$

$$z' = (p4 \times z + p5 \times 1) \times \text{scaleW}$$

$$w' = 1 \times \text{scaleW}$$

クリップ座標の各成分を w' だけ平行移動した後に $2w'$ で透視除算すると、正規化スクリーン座標 $(x'', y'', z'', 1)$ が求められます。 $\text{scaleW} = 1$ の場合には、1 平行移動して 2 で割ることになるため、クリップ座標系の $-1.0 \sim 1.0$ を正規化スクリーン座標系の $0.0 \sim 1.0$ へ変換することになります。

正規化スクリーン座標の成分 z'' は下記のようになります。

$$z'' = \frac{z' + w'}{2w'} = \frac{(p4 \times z + p5 + 1) \times \text{scaleW}}{2 \times \text{scaleW}} = \frac{p4 \times z + p5 + 1}{2}$$

p4 および p5 は、正射影行列の要素であるため「射影変換」の 2) 行列を適用すると、

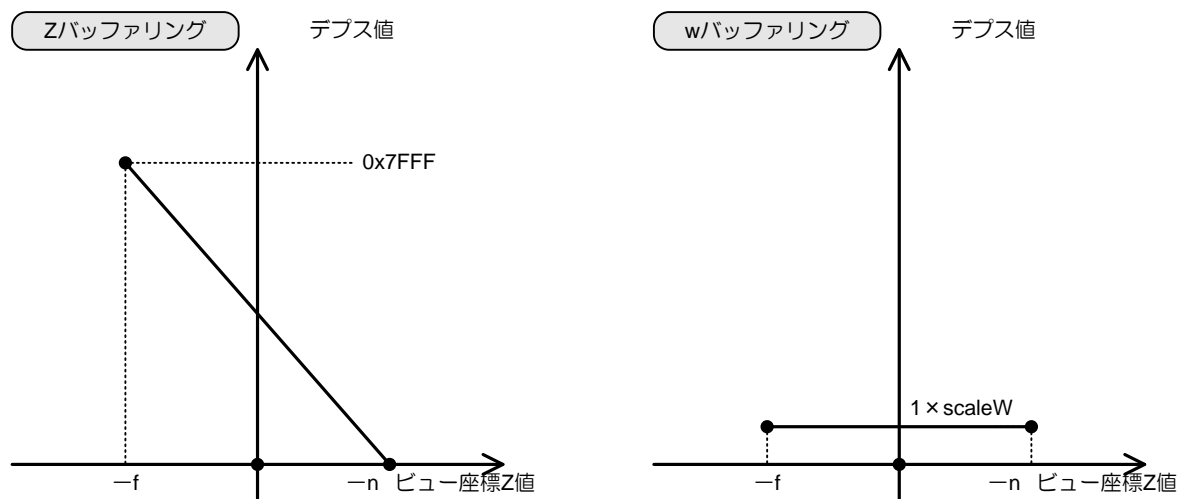
$$z'' = \frac{-1}{(f - n)}(z + n) \quad (-f \leq z \leq -n) \text{ となります。}$$

z バッファリング時には、この z'' に $0x7FFF$ を乗算した値 z''' がデプス値となります。

この場合デプス値 z''' がビュー座標 z に比例するため、遠方の描画精度の問題は発生しません。

w バッファリングを適用した場合、デプス値となるクリップ座標 w' は常に $1 \times \text{scaleW}$ という固定値になります。

このため、通常は正射影時において w バッファリングは使用しません。（図 7-7 参照）



f: 視点から far クリップ面までの距離

n: 視点から near クリップ面までの距離

図 7-7 Z バッファリングと w バッファリング（正射影時）

● デプス値のフォーマット

TWL では、デプスバッファが 1 ピクセル当たり 24 ビットになっているため、デプス値をその範囲内へ収めなければなりません。

Z バッファリング時は、near クリップ面から far クリップ面までの距離を 24 ビットの精度で表現した値がデプス値となります。

W バッファリング時は、ビュー座標系での視点からの距離をクリップ座標系の 24 ビット精度（符号+整数部 11 ビット+小数部 12 ビット）の範囲へ収めなければならず、クリップ座標の W 値を W だけ平行移動して 2 で割った値（整数部 12 ビット+小数部 12 ビット）がデプス値となります。

クリップ座標値が 24 ビットの範囲を超えた値にならないように注意してください。

また、フォグで使用するデプス値は 24 ビットのデプス値のうち、上位 15 ビットが使用されます。詳細は「クリアレジスタによる初期化」を参照してください。

7.2.6 ジオメトリコマンド

ジオメトリエンジンへ行列やポリゴン等のデータを送るには、コマンド FIFO に必要なコマンド列を書き込むことによって行います。

コマンド FIFO に書き込む方法としては、次の 2 種類の方法があります。

方法 1) メインプロセッサのレジスタ空間にマッピングされたコマンドレジスタ群にパラメータを書き込む

コマンドレジスタにパラメータを書き込むだけで自動的にコマンドコードとパラメータがコマンド FIFO に書き込まれるので、CPU が 1 命令ずつ処理するような場合に適しています。

方法 2) コマンド FIFO レジスタにコマンドコードとパラメータを書き込む

メインメモリ上に格納されたコマンド列を DMA 転送するなど、大量のデータをジオメトリエンジンに送る際に適しています。

注意事項

ジオメトリコマンドレジスタ群や、コマンド FIFO は 32 ビットアクセス専用です。

CPU、DMA のどちらを使用して書き込む場合でも必ず 32 ビット幅でアクセスしてください。

コマンド FIFO がフルのときに書き込みを行うと、FIFO に 32 ビットの空きが発生するまでウェイトが入ります。

ウェイト中は他のバスマスタであってもバスを使用することはできません。

この状態を避けるには次の方法があります。

● DMA の「ジオメトリコマンド FIFO 起動モード」を使用してコマンドを転送する

ジオメトリコマンド FIFO 起動モードは、コマンド FIFO が半分未満になると DMA が起動し 112 ワード※分の転送を行い、合計で設定した転送量に達すると終了するモードです（「DMA」参照）。

（※ コマンドパック時は展開前のワード数になります）

なお、ジオメトリエンジンステータスレジスタ（GXSTAT）のコマンド FIFO ステータスフラグによってコマンド FIFO の状態を確認することができます。

コマンド FIFO レジスタ

名称 GXFIFO アドレス 0x04000400 (イメージ: 0x04000404~0x0400043F) 属性 W

[illegible]

コマンド FIFO レジスタに書き込まれたデータはコマンド FIFO に送られます。

コマンド FIFO の深さは 32 ビット×256 段です。

未定義のコマンドコードを転送しないように注意してください。

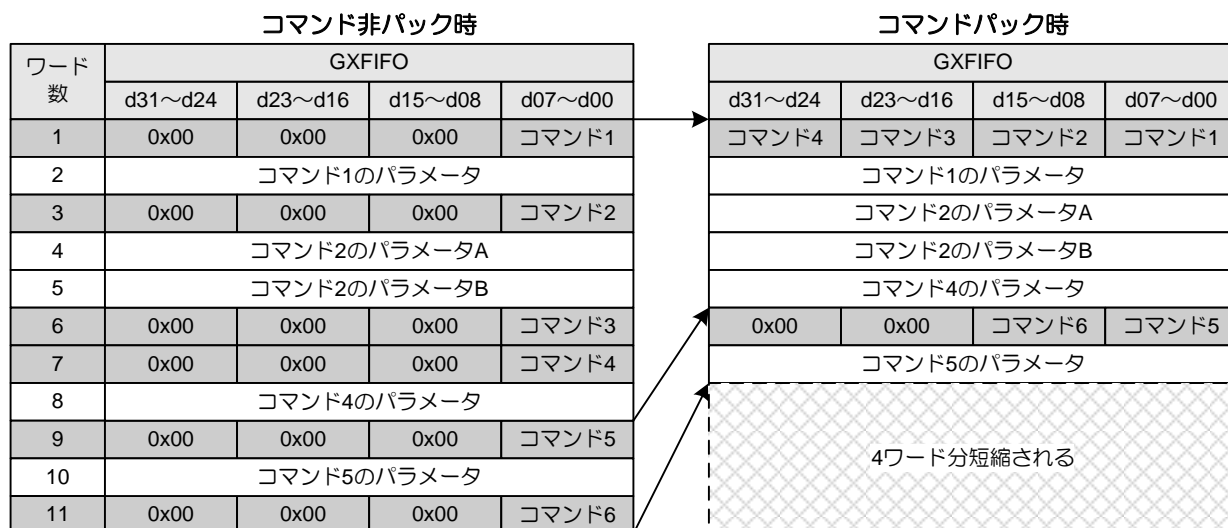
● コマンドパック

コマンド FIFO レジスタにコマンド列を転送する場合、1 ワード中に最大 4 つまでのコマンドコードをまとめることによってコマンド列を圧縮することができます。

バックされたコマンド列は、解凍された後にコマンド FIFO へ格納されます。

バックされたコマンド列は下位バイトから順にコマンド FIFO へ格納されるため、コマンドコードは下位アドレスから詰めて格納し、格納しない上位バイトはすべて 0 にしておく必要があります。

コマンド非パック時とコマンドパック時の転送コマンドの違いを図 7-8 に示します。



(コマンド 1,4,5 は 1 パラメータ、コマンド 2 は 2 パラメータ、コマンド 3,6 はパラメータなしと仮定)

図 7-8 コマンド非パック時とコマンドパック時の転送コマンドの違い

図 7-8 の例では、コマンド FIFO レジスタへ転送するデータ量がコマンド非バック時の 11 ワードに対してコマンドバック時では 7 ワード（4 ワードの短縮）で済みます。

● CPU によるジオメトリ FIFO への連続書き込み時の注意事項

STM 命令、STRD 命令によるジオメトリ FIFO への連続書き込みは次の 2 つの条件を満たした場合のみ正常に転送可能です。詳細を図 7-9 に示します。

- コマンドパックをしない
- コマンドとパラメータの一对だけを一度に書き込む

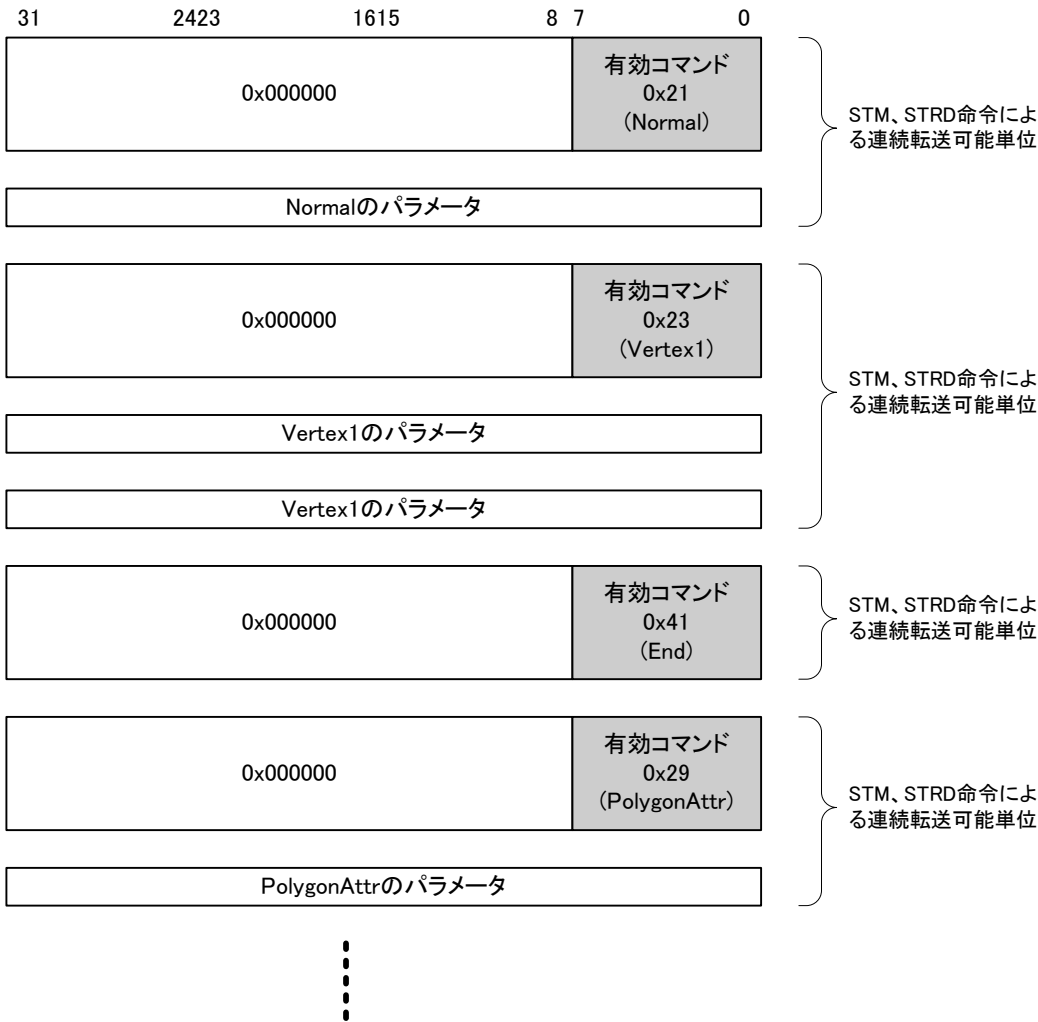


図 7-9 STM、STRD 命令によるジオメトリ FIFO への連続書き込み

上記の「STM、STRD 命令による連続転送可能単位」を一度にライトすることは可能ですが、この単位を超えての STM、STRD 命令による書き込みは行わないでください。
各「STM、STRD 命令による連続転送可能単位」間は 1 システムサイクル以上の間隔を空けてください。

● コマンドパック時のデータ列に関する注意事項

コマンドパックを使用する場合、ジオメトリ FIFO への書き込みは CPU、DMA のどちらで行うにしても、下記のいずれかの条件を満たした場合のみ正常に転送可能となります。

- コマンドパックの有効コマンド※1 最上位にパラメータなしコマンド※2 が来ないようにする。
詳細を図 7-10 および図 7-11 に示します。

※1 「有効コマンド」とは、0x10～0xff の範囲で定義されるコマンドを指します。

0x00～0x0f の範囲は無効コマンドとなり、該当しません。

※2 「パラメータなしコマンド」とは、下記 4 種類のコマンドを指します。

「PushMatrix」、「LoadIdentity」、「End」、0x10～0xff の範囲で未定義のコマンド。

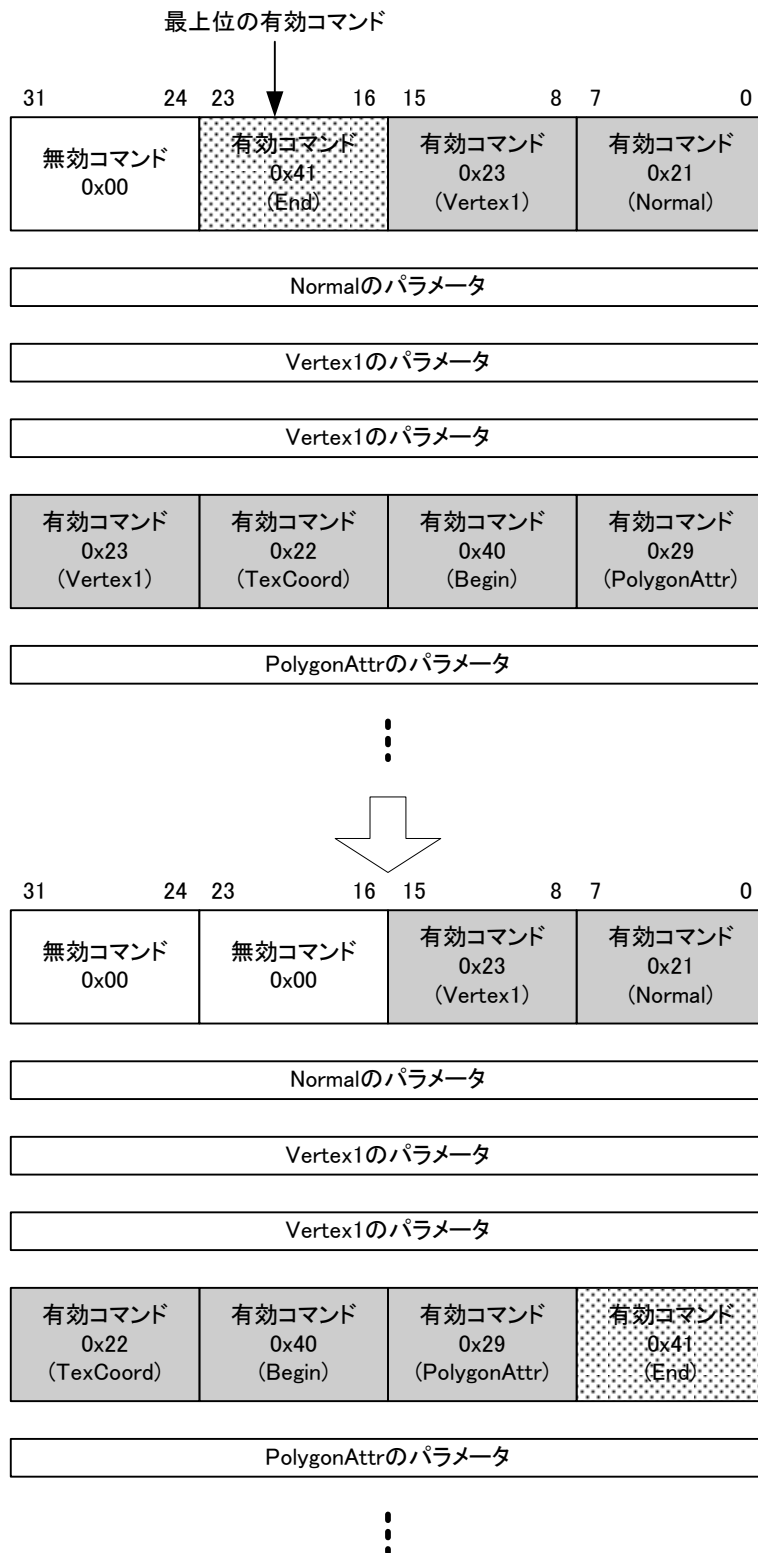


図 7-10 最上位の有効コマンドがパラメータなしコマンドにならないようにする場合 1

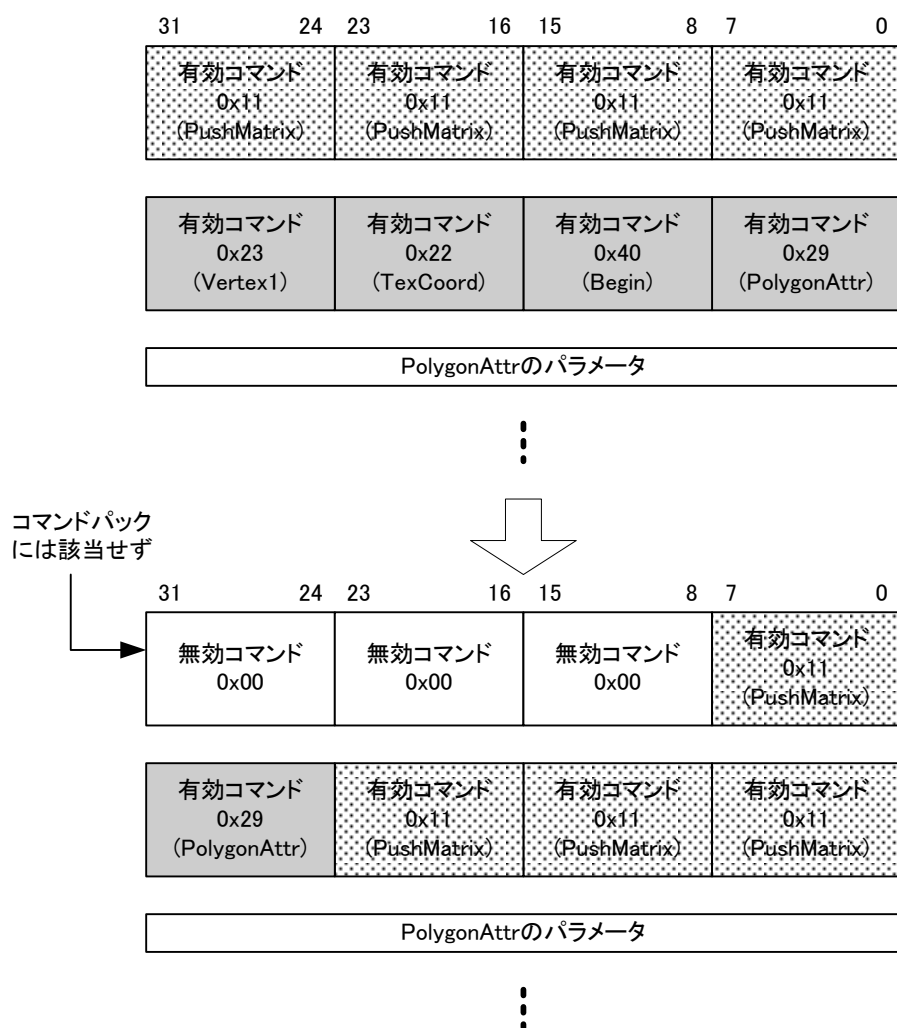


図 7-11 最上位の有効コマンドがパラメータなしコマンドにならないようにする場合 2

- コマンドパックの有効コマンド最上位にパラメータなしコマンドが来る場合、そのコマンドパックに対応するパラメータ列の最後に 32 ビットの 0 データを挿入する。

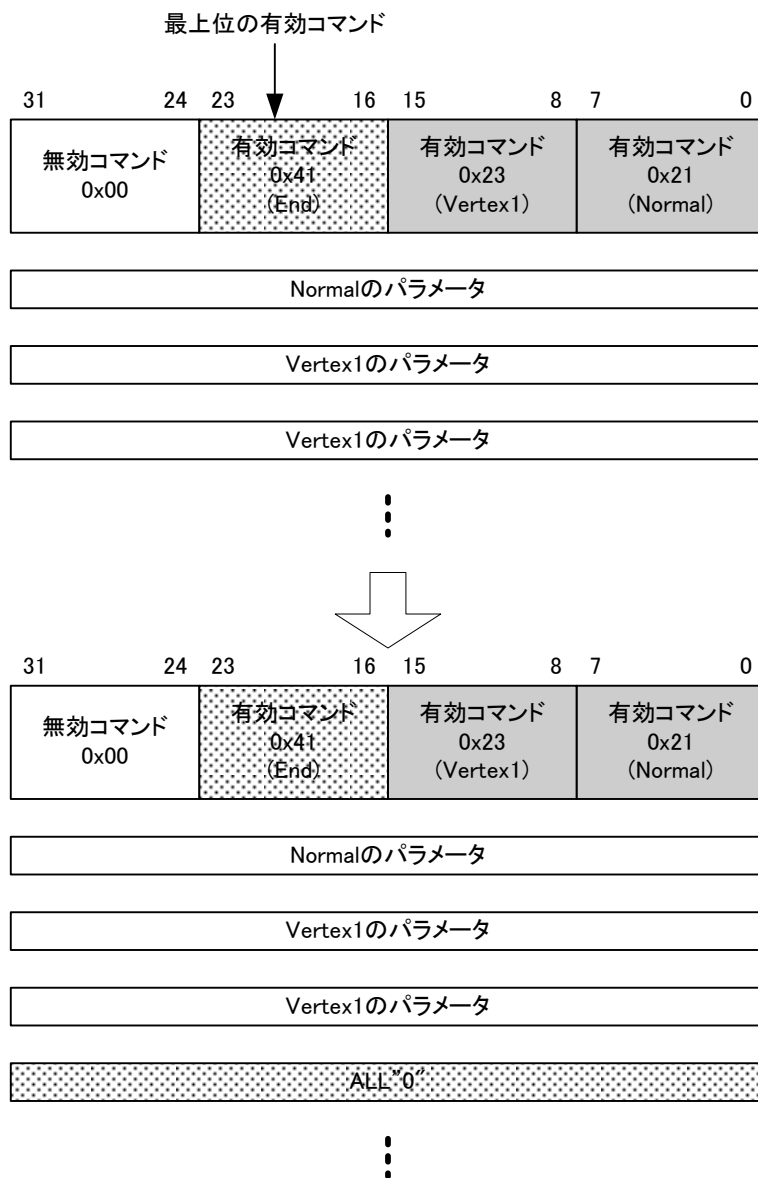


図 7-12 最上位の有効コマンドがパラメータなしコマンドの場合

ジオメトリコマンドの一覧を表 7-3 と表 7-4 に示します。

カテゴリ	機能	コマンド名	※ コマンド レジスタ アドレス	コマンド コード	パラメータ のワード数	参照頁
—	何もしない	Nop	—	0x00	0	なし
行列モード	行列モードを設定する	MatrixMode	0x440	0x10	1	185
カレント 行列に 対する演算	スタックへプッシュする	PushMatrix	0x444	0x11	0	189
	スタックからポップする	PopMatrix	0x448	0x12	1	189
	スタックの指定位置へ書き込む	StoreMatrix	0x44C	0x13	1	190
	スタックの指定位置から読み出す	RestoreMatrix	0x450	0x14	1	190
	単位行列に初期化する	Identity	0x454	0x15	0	186
	4×4 行列を設定する	LoadMatrix44	0x458	0x16	16	186
	4×3 行列を設定する	LoadMatrix43	0x45C	0x17	12	186
	4×4 行列を乗算する	MultMatrix44	0x460	0x18	16	187
	4×3 行列を乗算する	MultMatrix43	0x464	0x19	12	187
	3×3 行列を乗算する	MultMatrix33	0x468	0x1A	9	187
	スケール行列を乗算する	Scale	0x46C	0x1B	3	188
頂点情報	平行移動行列を乗算する	Translate	0x470	0x1C	3	188
	頂点カラーを直接設定する	Color	0x480	0x20	1	202
	法線ベクトルを設定する	Normal	0x484	0x21	1	202
頂点座標	テクスチャ座標を設定する	TexCoord	0x488	0x22	1	206
	頂点座標を設定する	Vertex	0x48C	0x23	2	203
	同上	VertexShort	0x490	0x24	1	203
	頂点の XY 座標を設定する	VertexXY	0x494	0x25	1	203
	頂点の XZ 座標を設定する	VertexXZ	0x498	0x26	1	203
	頂点の YZ 座標を設定する	VertexYZ	0x49C	0x27	1	203
	最後に設定した座標の差分値で 頂点を設定する	VertexDiff	0x4A0	0x28	1	204
ポリゴン属性	ポリゴン属性を設定する	PolygonAttr	0x4A4	0x29	1	198
テクスチャ 情報	テクスチャのパラメータを設定する	TexImageParam	0x4A8	0x2A	1	207
	テクスチャパレットの ベースアドレスを設定する	TexPlttBase	0x4AC	0x2B	1	210
マテリアル	環境反射色、拡散反射色を設定する	MaterialColor0	0x4C0	0x30	1	195
	放射光色、鏡面反射色を設定する	MaterialColor1	0x4C4	0x31	1	195
	鏡面反射輝度テーブルを設定する	Shininess	0x4D0	0x34	32	196
ライト	ライトの方向ベクトルを設定する	LightVector	0x4C8	0x32	1	192
	ライトカラーを設定する	LightColor	0x4CC	0x33	1	192
頂点リスト 開始／終了	頂点リストの開始を宣言する	Begin	0x500	0x40	1	201
	頂点リストの終了を宣言する	End	0x504	0x41	0	202
レンダリングエ ンジン参照デー タ切替え	レンダリングエンジンが参照する データ群をスワップする	SwapBuffers	0x540	0x50	1	183
ビューポート	ビューポートを設定する	ViewPort	0x580	0x60	1	184
テスト	直方体が視体積に入っているか テストする	BoxTest	0x5C0	0x70	3	214
	テスト用位置座標を設定する	PositionTest	0x5C4	0x71	2	215
	テスト用方向ベクトルを設定する	VectorTest	0x5C8	0x72	1	215

表 7-3 ジオメトリコマンド一覧（コマンドコード順）

※コマンドレジスタのアドレス値は、0x04000000 番地からのオフセット値です。
ジオメトリエンジンのコマンド FIFO に対して、未定義のコマンドコードを発行しないよう注意してください。

コマンド名	実行 サイクル数	発行できる タイミング	設定値が有効となる タイミング	設定値が破棄されるタイミング
Nop		制限なし		
MatrixMode	1	制限なし	コマンド実行時	次の MatrixMode コマンド実行時
PushMatrix	17			次の Push/StoreMatrix コマンド実行時
PopMatrix	36			次の Matrix 変更コマンド実行時
StoreMatrix	17			次の Push/StoreMatrix コマンド実行時
RestoreMatrix	36			次の Matrix 変更コマンド実行時
Identity	19			
LoadMatrix44	34			
LoadMatrix43	30			
MultMatrix44	35※ ₁			
MultMatrix43	31※ ₁			
MultMatrix33	28※ ₁			
Scale	22			次の Color, Normal
Translate	22※ ₁			
Color	1			
Normal	9～12 (+2)※ _{2, 5}			
TexCoord	1 (+1) ※ ₅			
Vertex	9 (+2) ※ ₅	次の Vertex 系コマンド実行時		
VertexShort	8 (+2) ※ ₅			
VertexXY	8 (+2) ※ ₅			
VertexXZ	8 (+2) ※ ₅			
VertexYZ	8 (+2) ※ ₅			
VertexDiff	8 (+2) ※ ₅			
PolygonAttr	1	Begin～End 外 のみ	Begin コマンド実行時 (Begin～End 単位で設 定値が有効) ※ ₃	次の PolygonAttr コマンド実行時
TexImageParam	1	ポリゴン単位 ※ ₄	コマンド実行時 (ポリゴン単位で設定 値が有効)	次の TexImageParam コマンド実行時
TexPlttBase	1			次の TexPlttBase コマンド実行時
MaterialColor0	4	制限なし	Normal コマンド実行時	次の MaterialColor0 コマンド実行時
MaterialColor1	4			次の MaterialColor1 コマンド実行時
Shininess	32			次の Shininess コマンド実行時
LightVector	6			次の LightVector コマンド実行時
LightColor	1			次の LightColor コマンド実行時
Begin	1		コマンド実行時	次の Begin コマンド実行時
End	1			設定値なし
SwapBuffers	392	Begin～End 外 のみ	V ブランク期間に 入ったとき	次の SwapBuffers コマンド実行時
ViewPort	1		コマンド実行時	次の ViewPort コマンド実行時
BoxTest	103			次の BoxTest コマンド実行時
PositionTest	9	次の PositionTest		
VectorTest	5	制限なし		次の VectorTest

表 7-4 ジオメトリコマンドの実行サイクル数および発行に関するタイミング一覧（コマンドコード順）

実行サイクル数は、システムクロック（33.514Mhz）換算の値であり、最小の値です。アクセスのバッティング、クリッピング、パイプラインハザード等が発生した場合は余分なサイクルが必要となります。

※1：Position・Vector 同時設定モード時は +30 サイクルかかります。

※2：イネーブル（ON）となっているライトの数によって増加します。

※3：PolygonAttr コマンドは Begin コマンドを実行することで有効になりますが、ライトイネーブルフラグを頂点カラーに反映させるには、さらに Normal コマンドを発行してライティング（照光処理）を再計算させてください。

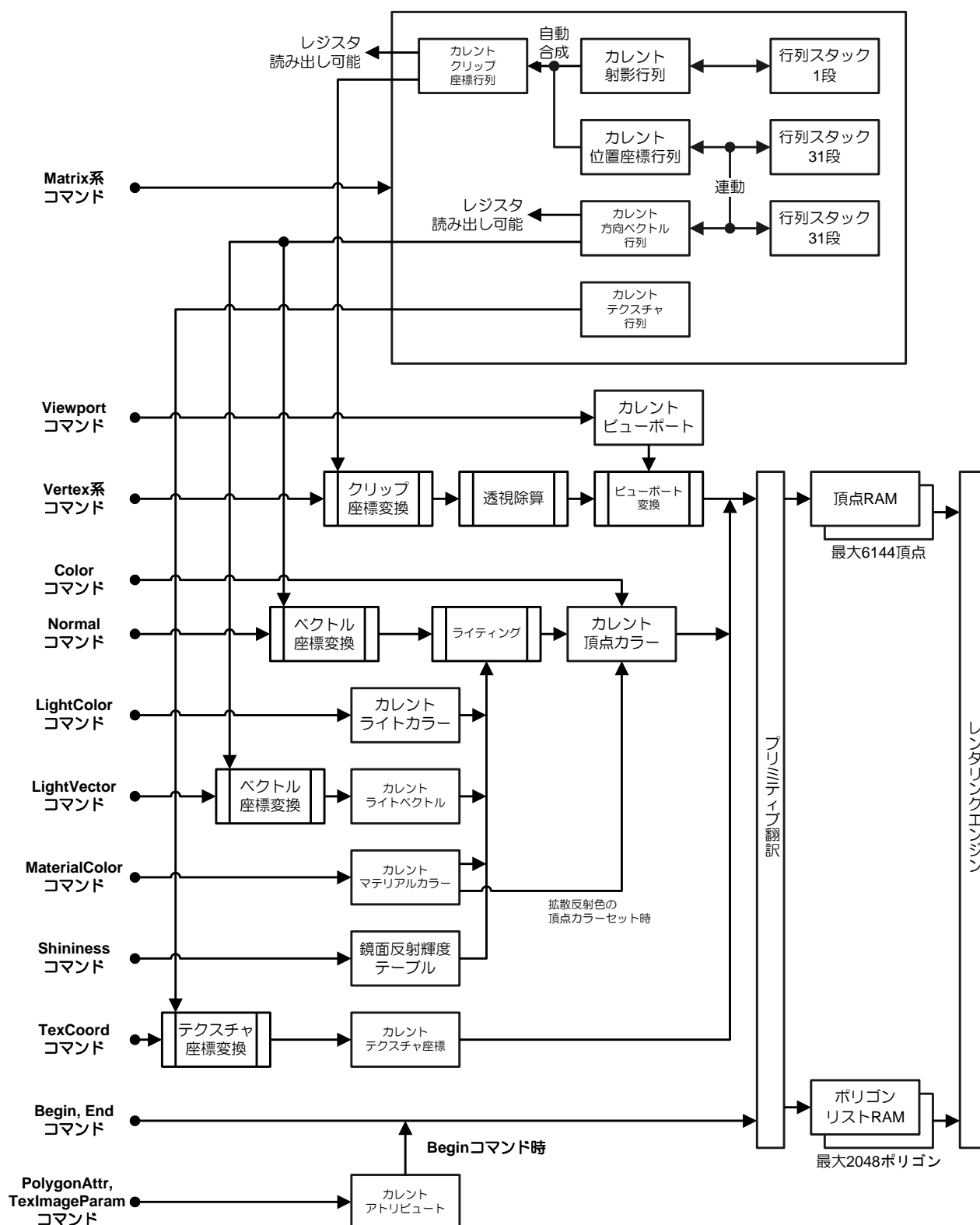
※4：「ポリゴン単位」に関して

ポリゴン単位に発行できるコマンドは、Vertex 系コマンド列のポリゴン単位の切れ目（下記●位置）で発行できません。

三角形ポリゴン	連結三角形ポリゴン	四角形ポリゴン	連結四角形ポリゴン
● Begin ● Vertex //Polygon 1 Vertex Vertex ● Vertex //Polygon 2 Vertex Vertex ● Vertex //Polygon 3 Vertex Vertex ● End ●	● Begin ● Vertex //Polygon 1 Vertex Vertex Vertex ● Vertex //Polygon 2 Vertex //Polygon 3 ● End ●	● Begin ● Vertex //Polygon 1 Vertex Vertex Vertex Vertex ● Vertex //Polygon 2 Vertex Vertex Vertex Vertex ● Vertex //Polygon 3 Vertex Vertex Vertex ● End ●	● Begin ● Vertex //Polygon 1 Vertex Vertex Vertex Vertex ● Vertex //Polygon 2 Vertex Vertex //Polygon 3 Vertex ● End ●

※5：テクスチャ座標変換モードでテクスチャ座標変換を行う場合、テクスチャ座標変換のため各ソースに相当するコマンドの実行サイクル数が表中括弧内の数字だけ増加します。

主なジオメトリコマンドの処理概要図を図 7-13 に示します。



7.2.7 レンダリングエンジン参照データのスワップ

SwapBuffers：レンダリングエンジンが参照するデータ群をスワップする

名称 SWAP_BUFFERS アドレス 0x04000540 属性 W コマンドコード 0x50

31	24	23	16	15	8	7	1	0
							DP	YS

● DP[d01]：デプスバッファリング選択フラグ

デプステストに使用する値を選択します。デプスバッファリング方式によるデプス値の違いに関しては、「ジオメトリエンジン」の「デプスバッファリング」を参照してください。

0	Z 値によるバッファリングを選択
1	W 値によるバッファリングを選択 (正射影では正常に機能しません)

● YS[d00]：半透明ポリゴンの Y ソーティング選択フラグ

半透明ポリゴンは ($1 \leq \alpha \leq 30$) または半透明テクスチャを貼ったポリゴンです。

シャドウボリュームを使用するときなどにおいて、描画順序を指定したいときはマニュアルソートモードにしてください。

0	オートソートモード
1	マニュアルソートモード

ジオメトリエンジンはレンダリングエンジンに渡すデータをポリゴンリスト RAM に書き出しますが、半透明ポリゴンについてはソートして書き出すか、ソートせず処理順のまま書き出すかを選択できます。

オートソートモードにすると、ポリゴンの LCD 上での最大 Y 値※が小さいものから順にソートされます。

最大 Y 値が同一のポリゴンについては、最小 Y 値が小さいものからソートされます。

マニュアルソートモードでは、ジオメトリエンジンに送ったポリゴン順にソートされます。

オートソートモードでは、レンダリングエンジンは最小 Y 値がスキャンライン（描画中の 1 ラインのこと）より大きいポリゴンや、最大 Y 値がスキャンラインより小さいポリゴンを参照しませんが、マニュアルソートモードではすべてのポリゴンを参照するため負荷が増加（描画効率が低下）します。

このため半透明ポリゴンが多い状況でマニュアルソートモードを使用する際は注意してください。

なお、不透明ポリゴンについては常にオートソートされます。

また、ソートモードに関わらず、常に不透明ポリゴン→半透明ポリゴンの順で描画されます。

※ポリゴンの LCD 上での最大 Y 値

LCD 上での座標は、「座標変換」の座標変換フロー図（図 7-3）における「BG スクリーン座標系」の Y 軸を反転した座標になります。

このため、ポリゴンの LCD 上での最大 Y 値は「BG スクリーン座標系」における最小 Y 値ということになります。

● 「デプスバッファリング選択フラグ」および「半透明ポリゴンの Y ソーティング選択フラグ」の反映について

これらのフラグはジオメトリエンジンにおいて次のフレームから反映されます。

しかしポリゴンリスト RAM と頂点 RAM はダブルバッファ構成になっており、1 つ前のフレームでジオメトリエンジンが格納したデータをレンダリングエンジンが描画します。

このためジオメトリエンジンが出力したデータが描画へ反映されるのは、さらに 1 フレーム遅れることになります。

● SwapBuffers コマンドの処理

SwapBuffers コマンドはどのタイミングで入力しても次の V ブランクで処理されます。（その V ブランク期間が来るまで、ジオメトリエンジンはウェイト状態になります）

SwapBuffers コマンドを発行した次の V ブランク期間開始時に、ポリゴンリスト RAM、頂点 RAM、レンダリング関連レジスタ等、レンダリングエンジンが参照するデータを切り替えます。

このため、書き込んだグラフィックスデータが描画へ反映されるのは SwapBuffers コマンドを発行した次のフレームになります。

7.2.8 ビューポート

Viewport：ビューポートを設定する

名称 VIEWPORT アドレス 0x04000580 属性 W コマンドコード 0x60

31	24	23	16	15	8	7	0
INTEGER_Y2		INTEGER_X2		INTEGER_Y1		INTEGER_X1	
Y2		X2		Y1		X1	

- Y2, X2 [d31~d24,d23~d16]：右上座標
Y2 は Y1 より大きい値を設定してください。設定範囲は 0~191 です。
X2 は X1 より大きい値を設定してください。設定範囲は 0~255 です。
- Y1, X1[d15~d08,d07~d00]：左下座標（ビューポートの原点）
Y1 は Y2 より小さい値を設定してください。設定範囲は 0~191 です。
X1 は X2 より小さい値を設定してください。設定範囲は 0~255 です。

3D イメージを描画するビューポートの BG0 面上における位置と大きさを設定します。
LCD 上の表示位置は、BG0 の H オフセット値を加算した位置になります。
2D の座標系とは原点の位置が異なることに注意してください（図 7-14 参照）。

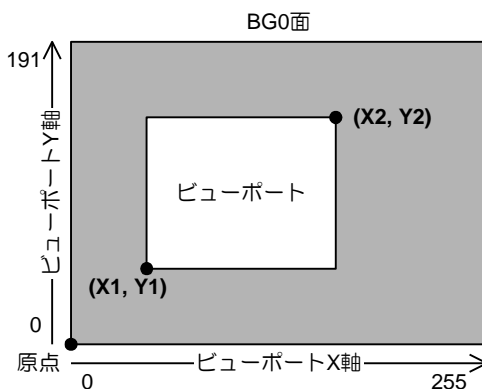


図 7-14 ビューポートの位置と大きさ

注意事項

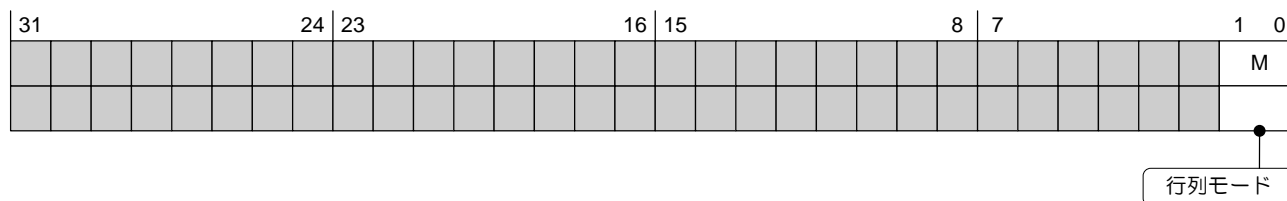
ビューポートの右端および下端に関しては、1 ドットはみ出して描画される場合があります。

7.2.9 行列

7.2.9.1 カレント行列の操作

MatrixMode：行列モードを設定する

名称 MTX_MODE アドレス 0x04000440 属性 W コマンドコード 0x10



● M[d01~d00]：行列モード

00	Projection モード	…（射影行列を操作するモード）
01	Position モード	…（位置座標行列を操作するモード）
10	Position・Vector 同時設定モード	…（位置座標行列と方向ベクトル行列を操作するモード）
11	Texture モード	…（テクスチャ行列を操作するモード）

行列コマンドで操作するカレント行列を指定します（この区分を「行列モード」と呼びます）。

● Position モードと Position・Vector 同時設定モードについて

TWL では、ハードウェアで法線の単位ベクトル化を行いません。

このため正しいライティング結果を得るには、法線にあらかじめ単位ベクトルを設定しておき、方向ベクトル行列も直交行列（変換によって方向ベクトルの長さが変わらない行列）である必要があります。

通常はモデルの変換を行う際、変換行列を位置座標行列と方向ベクトル行列の両方へ適用しますが、TWL では方向ベクトル行列を直交行列として保つ必要があるために、変換の種類によっては位置座標行列のみに適用する場合もあります。

Position モードと Position・Vector 同時設定モードはこのようなケースで使い分けてください。

（具体例）

モデルの回転を行う際、通常は Position・Vector 同時設定モードにし、MultMatrix コマンドで行列の回転成分を位置座標行列と方向ベクトル行列の両方へ適用します。

モデルのスケールリングは Scale コマンドで行うことによって、方向ベクトル行列を直交行列として維持することができ、正常なライティング結果が得られます（[Scale コマンド参照](#)）。

MultMatrix コマンドでスケール行列を適用すると方向ベクトル行列が直交行列ではなくなってしまう、ライティング結果が本来の明るさより明る過ぎたり暗過ぎたりすることになるため基本的にはこのようなことは行わない方が無難です。

Position モードは位置座標行列のみにしか行列コマンドが適用されないため、MultMatrix コマンドでスケール行列を適用しても正常なライティング結果が得られます。

回転行列を位置座標行列のみに適用したい場合にも使用できますが、回転によってライティング結果が変わらなくなり、ライトが照射されていない部分が最も明るくなるなど不自然な表示となる場合があります。

Identity：カレント行列を単位行列に初期化する

名称 MTX_IDENTITY アドレス 0x04000454 属性 W コマンドコード 0x15

31	24	23	16	15	8	7	0

LoadMatrix44：カレント行列に 4×4 行列を設定する

名称 MTX_LOAD_4x4 アドレス 0x04000458 属性 W コマンドコード 0x16

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_M44								DECIMAL_M44	
符	整数部								小数部	

符号付き固定小数点数（符号+整数部 19 ビット+小数部 12 ビット）

- M44：4×4 行列要素 m[x]（x=0～15）

行列 M は、m[0]～m[15]によって次のように設定されます。

$$M = \begin{bmatrix} m[0] & m[1] & m[2] & m[3] \\ m[4] & m[5] & m[6] & m[7] \\ m[8] & m[9] & m[10] & m[11] \\ m[12] & m[13] & m[14] & m[15] \end{bmatrix}$$

LoadMatrix43：カレント行列に 4×3 行列を設定する

名称 MTX_LOAD_4x3 アドレス 0x0400045C 属性 W コマンドコード 0x17

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_M43								DECIMAL_M43	
符	整数部								小数部	

符号付き固定小数点数（符号+整数部 19 ビット+小数部 12 ビット）

- M43：4×3 行列要素 m[x]（x=0～11）

行列 M は、m[0]～m[11]によって次のように設定されます。

$$M = \begin{bmatrix} m[0] & m[1] & m[2] & 0 \\ m[3] & m[4] & m[5] & 0 \\ m[6] & m[7] & m[8] & 0 \\ m[9] & m[10] & m[11] & 1 \end{bmatrix}$$

MultMatrix44：4×4 行列とカレント行列を乗算する

名称 MTX_MULT_4x4 アドレス 0x04000460 属性 W コマンドコード 0x18

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_M44								DECIMAL_M44	
符	整数部								小数部	

符号付き固定小数点数（符号+整数部 19 ビット+小数部 12 ビット）

● M44：4×4 行列要素 m[x]（x=0～15）

行列 M は、m[0]～m[15]によって次のように設定されます。

$$M = \begin{bmatrix} m[0] & m[1] & m[2] & m[3] \\ m[4] & m[5] & m[6] & m[7] \\ m[8] & m[9] & m[10] & m[11] \\ m[12] & m[13] & m[14] & m[15] \end{bmatrix}$$

カレント行列を C とすると、新しいカレント行列は C'=MC となります。

MultMatrix43：4×3 行列とカレント行列を乗算する

名称 MTX_MULT_4x3 アドレス 0x04000464 属性 W コマンドコード 0x19

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_M43								DECIMAL_M43	
符	整数部								小数部	

符号付き固定小数点数（符号+整数部 19 ビット+小数部 12 ビット）

● M43：4×3 行列要素 m[x]（x=0～11）

行列 M は、m[0]～m[11]によって次のように設定されます。

$$M = \begin{bmatrix} m[0] & m[1] & m[2] & 0 \\ m[3] & m[4] & m[5] & 0 \\ m[6] & m[7] & m[8] & 0 \\ m[9] & m[10] & m[11] & 1 \end{bmatrix}$$

カレント行列を C とすると、新しいカレント行列は C'=MC となります。

MultMatrix33：3×3 行列とカレント行列を乗算する

名称 MTX_MULT_3x3 アドレス 0x04000468 属性 W コマンドコード 0x1A

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_M33								DECIMAL_M33	
符	整数部								小数部	

符号付き固定小数点数（符号+整数部 19 ビット+小数部 12 ビット）

● M33：3×3 行列要素 m[x]（x=0～8）

行列 M は、m[0]～m[8]によって次のように設定されます。

$$M = \begin{bmatrix} m[0] & m[1] & m[2] & 0 \\ m[3] & m[4] & m[5] & 0 \\ m[6] & m[7] & m[8] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

カレント行列を C とすると、新しいカレント行列は C'=MC となります。

Translate : 平行移動行列とカレント行列を乗算する

名称 MTX_TRANS アドレス 0x04000470 属性 W コマンドコード 0x1C

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_TRANSLATE								DECIMAL_TRANSLATE	
符	整数部								小数部	

符号付き固定小数点数 (符号+整数部 19 ビット+小数部 12 ビット)

● TRANSLATE : 平行移動行列要素 $m[x]$ ($x=0\sim 2$)行列 M は、 $m[0]\sim m[2]$ によって次のように設定されます。

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ m[0] & m[1] & m[2] & 1 \end{bmatrix}$$

カレント行列を C とすると、新しいカレント行列は $C'=MC$ となります。**Scale : スケール (拡大縮小) 行列とカレント行列を乗算する**

名称 MTX_SCALE アドレス 0x0400046C 属性 W コマンドコード 0x1B

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_SCALE								DECIMAL_SCALE	
符	整数部								小数部	

符号付き固定小数点数 (符号+整数部 19 ビット+小数部 12 ビット)

● SCALE : スケール行列要素 $m[x]$ ($x=0\sim 2$)行列 M は、 $m[0]\sim m[2]$ によって次のように設定されます。

$$M = \begin{bmatrix} m[0] & 0 & 0 & 0 \\ 0 & m[1] & 0 & 0 \\ 0 & 0 & m[2] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

カレント行列を C とすると、新しいカレント行列は $C'=MC$ となります。

Scale コマンドは、行列モードコマンドで Position・Vector 同時設定モードが指定されていた場合でも位置座標行列のみに乗算が適用されます(方向ベクトル行列へ適用するとベクトルの方向や長さが変化することでライティングの結果が異なってしまうため)。

7.2.9.2 行列スタック

カレント行列のスタックを操作します。

ただし、位置座標行列スタックと方向ベクトル行列スタックは連動しているため、行列モードが Position モードと Position・Vector 同時設定モードのどちらの場合でも、両方のスタックに対する操作となります。

PushMatrix : カレント行列をスタックへプッシュする

名称 MTX PUSH アドレス 0x04000444 属性 W コマンドコード 0x11

[illegible]

PopMatrix：カレント行列をスタックからポップする

名称 MTX POP アドレス 0x04000448 属性 W コマンドコード 0x12

[illegible]

符号付き整数（符号+整数部5ビット）

- NUM[d05～d00]：ポップ数指定（-30～31 の値を設定可能）

行列モードで指定されている行列スタックのスタックポインタ位置から、NUM 段目の行列をポップレカレント行列として設定します。

行列モードが Projection の場合はスタックが 1 段しかないため、NUM は設定に関わらず 1 とみなされます。

通常は Begin—End の外で発行しますが、Begin—End 中の Vertex 間で発行することも可能です。

コマンド列の例 1	PushMatrix ⇒ Translate ⇒ Begin ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End ⇒ PopMatrix(1)
コマンド列の例 2	PushMatrix ⇒ Translate ⇒ PushMatrix ⇒ Begin ⇒ Vertex ⇒ PopMatrix(1) ⇒ Vertex ⇒ PopMatrix(-1) ⇒ Vertex ⇒ PopMatrix(1) ⇒ Vertex ⇒ End ⇒ PopMatrix(1)

「コマンド列の例 2」は、Vertex 間で PopMatrix を発行することによってスキニングの一種であるスティッチングやスプライトポリゴン（2D として表示するポリゴン）の変形等を実現する例です。

StoreMatrix：カレント行列をスタックの指定位置に格納する

名称 MTX_STORE アドレス 0x0400044C 属性 W コマンドコード 0x13

31	24	23	16	15	8	7	4	0
								INDEX
								格納位置

符号なし整数（整数部 5 ビット）

● INDEX[d04～d00]：格納位置（0～30 の値を設定可能）

行列スタックの INDEX 番目に行列モードで指定された行列を格納します

行列モードが Projection の場合はスタックが 1 段しかないため、INDEX は設定に関わらず 0 とみなされます。

また、PushMatrix と PopMatrix コマンドで移動した行列スタックポインタはこのコマンド発行後には移動しません。

RestoreMatrix：スタックの指定位置から行列を読み出す

名称 MTX_RESTORE アドレス 0x04000450 属性 W コマンドコード 0x14

31	24	23	16	15	8	7	5	4	0
									INDEX
									読み出し位置

符号なし整数（整数部 5 ビット）

● INDEX[d04～d00]：読み出し位置（0～30 の値を設定可能）

行列スタックの INDEX 番目の値を行列モードで指定された行列として設定します。

行列モードが Projection の場合はスタックが 1 段しかないため、NUM は設定に関わらず 0 とみなされます。

また、PushMatrix と PopMatrix コマンドで移動した行列スタックポインタはこのコマンド発行後には移動しません。

通常は Begin—End の外で発行しますが、Begin—End 中の Vertex 間で発行することも可能です。

コマンド列の例 1	StoreMatrix(i) ⇒ Translate ⇒ Begin ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End ⇒ RestoreMatrix(i)
コマンド列の例 2	StoreMatrix(i) ⇒ Translate ⇒ StoreMatrix(i+1) ⇒ Begin ⇒ Vertex ⇒ RestoreMatrix(i) ⇒ Vertex ⇒ RestoreMatrix(i+1) ⇒ Vertex ⇒ End ⇒ RestoreMatrix(i)

「コマンド列の例 2」は、Vertex 間で RestoreMatrix を発行することによってスキニングの一種であるスティッチングやスプライトポリゴン（2D として表示するポリゴン）の変形等を実現する例です。

7.2.9.3 カレント行列の読み出し

ClipMatrix：カレントクリップ座標行列を読み出す

名称 CLIPMTX_RESULT_x (x=0~15)

アドレス 0x04000640, 0x04000644, 0x04000648, 0x0400064C, 0x04000650, 0x04000654, 0x04000658, 0x0400065C,
0x04000660, 0x04000664, 0x04000668, 0x0400066C, 0x04000670, 0x04000674, 0x04000678, 0x0400067C

属性 R 初期値 0x00000000

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_m[x]								DECIMAL_m[x]	
符	整数部								小数部	

符号付き固定小数点数 (符号+整数部 19 ビット+小数部 12 ビット)

- m[x] (x=0~15)：カレントクリップ座標行列の各要素

$$\text{カレントのクリップ座標行列} = \begin{bmatrix} m[0] & m[1] & m[2] & m[3] \\ m[4] & m[5] & m[6] & m[7] \\ m[8] & m[9] & m[10] & m[11] \\ m[12] & m[13] & m[14] & m[15] \end{bmatrix}$$

カレントのクリップ座標行列 ([位置座標行列]・[射影行列]) を読み出すことができます。

カレント射影行列を読み出したい場合はカレント位置座標行列を単位行列にしてこのレジスタを読み出してください。

カレント位置座標行列を読み出したい場合はカレント射影行列を単位行列にしてこのレジスタを読み出してください。

安全に読み出すためにはジオメトリエンジンの停止確認後に読み出す必要があります。

VectorMatrix：カレント方向ベクトル行列を読み出す

名称 VECMTX_RESULT_x (x=0~8)

アドレス 0x04000680, 0x04000684, 0x04000688, 0x0400068C, 0x04000690, 0x04000694, 0x04000698, 0x0400069C,
0x040006A0

属性 R 初期値 0x00000000

31	30	24	23	16	15	12	11	8	7	0
S	INTEGER_m[x]								DECIMAL_m[x]	
符	整数部								小数部	

符号付き固定小数点数 (符号+整数部 19 ビット+小数部 12 ビット)

- m[x] (x=0~8)：カレント方向ベクトル行列の各要素

$$\text{カレント方向ベクトル行列} = \begin{bmatrix} m[0] & m[1] & m[2] \\ m[3] & m[4] & m[5] \\ m[6] & m[7] & m[8] \end{bmatrix}$$

安全に読み出すためにはジオメトリエンジンの停止確認後に読み出す必要があります。

7.2.10 ライト

TWL では平行光源のみサポートしています。

LightVector：ライトの方向ベクトルを設定する

名称 LIGHT_VECTOR アドレス 0x040004C8 属性 W コマンドコード 0x32

31	30	29	28	24	23	20	19	18	16	15	10	9	8	7	0	
LNUM	SZ	DECIMAL_Z					SY	DECIMAL_Y					SX	DECIMAL_X		
ライト	方向ベクトル Z 成分					方向ベクトル Y 成分					方向ベクトル X 成分					

符号付き固定小数点数（符号+小数部 9 ビット）

- LNUM[d31～d30]：ライト番号
0～3
- X, Y, Z[d29～d20], [d19～d10], [d09～d00]：方向ベクトル
設定後に方向ベクトル行列による座標変換が行われます。
ハードウェアによるベクトルの正規化は行われないため、単位ベクトルを設定してください。

LightColor：ライトカラーを設定する

名称 LIGHT_COLOR アドレス 0x040004CC 属性 W コマンドコード 0x33

31	30	24	23	16	15	14	10	9	8	7	5	4	0
LNUM											BLUE	GREEN	RED
ライト											ライトカラー		

- LNUM[d31～d30]：ライト番号
0～3
- [d14～d00]：ライトカラー
OpenGL のライトカラーには拡散光、鏡面光、環境光のパラメータがありますが、TWL では 1 つのパラメータへ簡略化されています。

7.2.11 マテリアル

物体は、物体表面の材質や置かれた環境によって見た目の質感が異なります。ライティング（照光処理）では、図 7-15 のように、鏡面反射色、拡散反射色、環境反射色、放射光色の 4 つのマテリアルカラーを使ってモデルの質感を表現します。

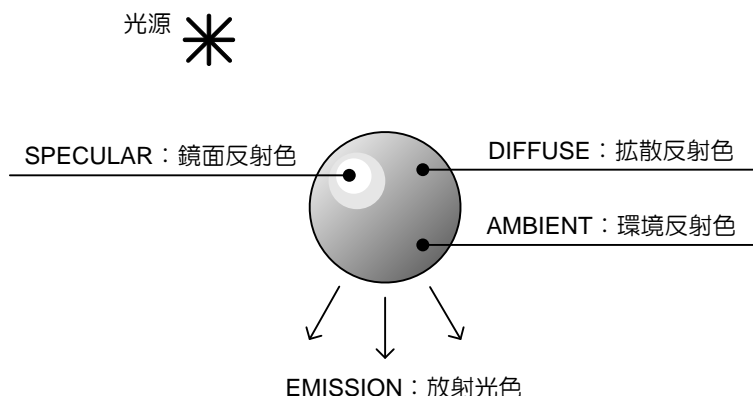


図 7-15 マテリアルカラー概念図

● 拡散反射色（ディフューズ）

オブジェクトがライトによって照らされたときのカラーです。オブジェクトの基本的なカラーと考えてください。拡散反射色は全方向に均等に反射されるものとして定義されるため視点の位置には影響されません。

図 7-16 のように、ライトのカラーと方向、およびポリゴンの法線に影響されます。

オブジェクトがライトによって直接照らされている部分のカラーにのみ影響を与えます。

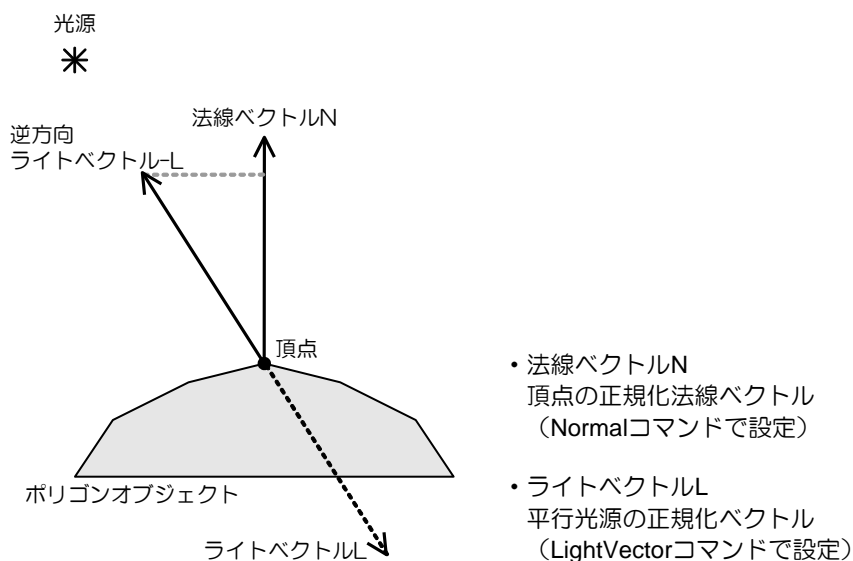


図 7-16 方向ベクトル関係図（拡散反射色）

● 環境反射色（アンビエント）

オブジェクトが環境光によって照らされたときのカラーです。

オブジェクトはライトの直接光だけでなく、他のオブジェクトからの反射光を受けています。この反射光を、シーン全体に一様に存在していると定義したとき、これを環境光と呼びます。環境光はシーンに一様に存在するため、オブジェクト全体のカラーに影響を与えます。

オブジェクトのカラーに与える影響について、ライトの直接光が当たる部分では拡散反射色が強く影響し、陰になる部分では環境反射色が支配的に影響します。

● 鏡面反射色（スペキュラ）

オブジェクトがライトによって照らされたときの光沢のカラーです。この光沢をスペキュラハイライトと呼びます。スペキュラハイライトは光学的には光源の映り込みによる反射を意味します。このため、光がオブジェクトに当たってまっすぐ視点に反射してくる部分が最も明るくなります。つまりライトのカラーと方向、ポリゴンの法線、視点の位置に影響されます。（図 7-17 参照）視点をずらすと、スペキュラハイライトは移動します。オブジェクトがライトによって直接照らされている部分のカラーにのみ視点に応じて影響を与えます。

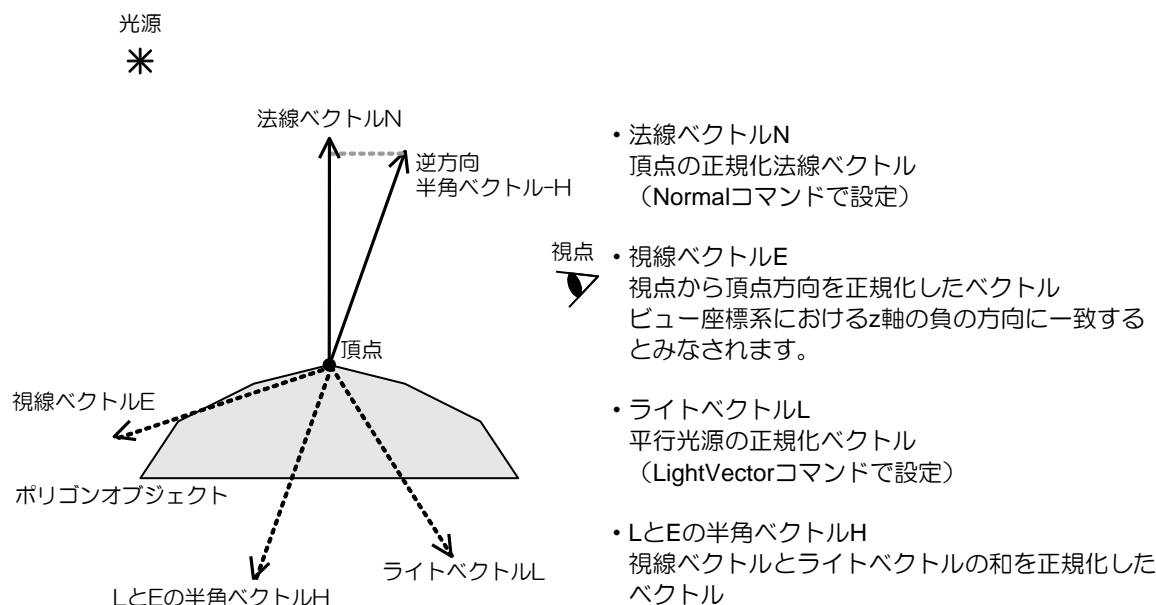


図 7-17 方向ベクトル関係図(鏡面反射色)

TWL の鏡面反射輝度の計算では、視線ベクトルをビュー座標系における z 軸の負の方向に一致するとみなし、ライトベクトルと法線ベクトルもビュー座標系に変換されることが前提になっています。このためビュー行列（LookAt 行列）を射影行列へ適用すると、ライトベクトルと法線ベクトルの変換後の座標系が視線ベクトルの座標系と異なってしまい、鏡面反射の結果が正常ではなくなってしまいます。よって、鏡面反射色へ黒（0）以外を設定する場合には、行列モードを Position・Vector 同時設定モードにして、ビュー行列（LookAt 行列）の回転成分を方向ベクトル行列に反映させる必要があります。鏡面反射色へ黒を設定すれば、拡散反射は視点に依存しないため、ビュー行列を射影行列へ適用することも可能です。つまり、位置座標行列にはモデル行列を設定し、射影行列にはビュー行列と射影行列を合成したものを設定できるということです。

● 放射光色（エミッション）

物体自身から発する光のカラーです。ただしライトのように扱われないので、他の物体を照らすことはありません（他の物体のカラーに影響を与えません）。その効果を得るには、放射光色と同じ色の光源を作成し、発光させたい物体と同じ位置に置く必要があります。

MaterialColor0：材質の拡散反射色と環境反射色を設定する

名称 DIF_AMB アドレス 0x040004C0 属性 W コマンドコード 0x30

31	30	26	25	24	23	21	20	16	15	14	10	9	8	7	5	4	0
	AMBIENT_BLUE		AMBIENT_GREEN		AMBIENT_RED		C		DIFFUSE_BLUE		DIFFUSE_GREEN		DIFFUSE_RED				
	環境反射色							頂	拡散反射色								

● C[d15]：頂点カラーセットフラグ

0	頂点カラーをセットしない
1	拡散反射色を頂点カラーとしてセットする

拡散反射色を頂点カラーにセットした場合、次の Color、Normal、MaterialColor0（頂点カラーセットフラグ）のいずれかのコマンドによってカレント頂点カラーが更新されるまで有効です。

また、レンダリングエンジンでは頂点カラーが（R:G:B = 6:6:6）ビットで扱われるため、拡散反射色が上位 5 ビットに適用されます。

このとき、拡散反射色が 0 のときは下位 1 ビットが 0 となり、拡散反射色が 0 以外のときは下位 1 ビットが 1 となります。

MaterialColor1：材質の鏡面反射色と放射光色を設定する

名称 SPE_EMI アドレス 0x040004C4 属性 W コマンドコード 0x31

31	30	26	25	24	23	21	20	16	15	14	10	9	8	7	5	4	0
	EMISSION_BLUE		EMISSION_GREEN		EMISSION_RED		S		SPECULAR_BLUE		SPECLAR_GREEN		SPECULAR_RED				
	放射光色							輝	鏡面反射色								

● S[d15]：鏡面反射輝度テーブル・イネーブルフラグ

0	ディセーブル
1	イネーブル

Shininess : 鏡面反射輝度テーブルを設定する

名称 SHININESS アドレス 0x0400004D0 属性 W コマンドコード 0x34

31	24	23	16	15	8	7	0
SHININESS_n (n=4x+3)				SHININESS_n (n=4x+2)			
Is=n 時の輝度				Is=n 時の輝度			

符号なし固定小数点数 (小数部 8 ビット)

鏡面反射の輝度を変換する 8 ビット×128 のテーブルを設定します。

直前に発行された MaterialColor1 コマンドで鏡面反射輝度テーブル・イネーブルフラグが 1 に設定された場合、ジオメトリエンジンはスペキュラ計算結果 Is の上位 7 ビットを元にテーブルを引き、鏡面反射輝度の変換を行います。このテーブルによって、鏡面反射の光り具合を調節することができます (計算式は「ライティング (照光処理)」参照)。また、鏡面反射輝度テーブルを書き換えることで、1 シーン中に複数の異なる鏡面反射効果を持ったポリゴンを表示できます。

鏡面反射輝度の計算結果 Is は、ベクトルの内積から求められるため、図 7-18 のように、光沢の中心に近いほど精度が粗く、離れるほど精度が良くなります ($A \cdot B = |A| |B| \cos \theta$)。Is の正確な計算式は「ライティング (照光処理)」を参照してください。

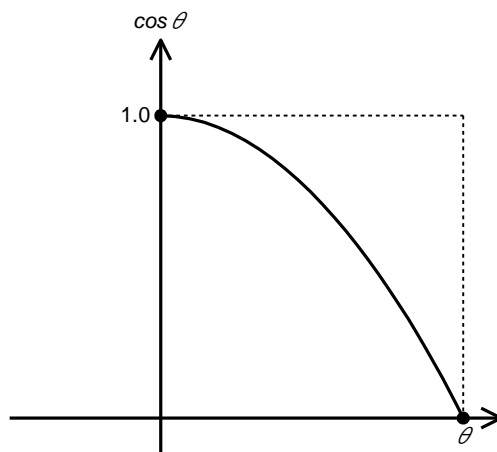


図 7-18 鏡面反射輝度

● テクニック

鏡面反射輝度テーブルを不連続な値に設定して、特殊なライティング効果を出すこともできます。

7.2.11.1 ライティング（照光処理）

ライティングは Normal コマンド発行時に行われ、その計算結果が頂点カラーとなります。
下記の演算が R,G,B 成分それぞれに対して行われます。

● 各マテリアルカラーのライティング演算式

マテリアルカラー	ライティング演算式
拡散反射色	$I_d = \max[0, -L \cdot N]$ $D = I_d * \text{light} * \text{diffuse_material}$
環境反射色	$A = \text{light} * \text{ambient_material}$
鏡面反射色	$I_s = \max[0, \cos 2\theta]$ （鏡面反射輝度テーブル・ディセーブル時） $S = I_s * \text{light} * \text{specular_material}$ （鏡面反射輝度テーブル・イネーブル時） $S = \text{shininess_table}[I_s] * \text{light} * \text{specular_material}$
放射光色	$E = \text{emission_material}$

L：ライトの方向ベクトル

N：法線ベクトル

H：L（ライトの方向ベクトル）と、視線ベクトル（Z 軸の負の方向に沿ったベクトル）の和を 2 で割ったベクトル
（両者を 2 等分する方向を指すため、「半角ベクトル」と呼びます）

θ ：ベクトル（-H）とベクトル（N）間の角度

I_d ：拡散反射輝度

I_s ：鏡面反射輝度

light：ライトカラー

diffuse_material：マテリアル拡散反射色

ambient_material：マテリアル環境反射色

specular_material：マテリアル鏡面反射色

emission_material：マテリアル放射光色

● 頂点カラー計算式

最終的な頂点カラーは、各マテリアルカラーのライティング結果を用いて下記式に従って算出されます。

$$C = \sum_{i=0}^3 [D_i + A_i + S_i] + E$$

C：頂点カラー

D_i ：ライト i に対する拡散反射色

A_i ：ライト i に対する環境反射色

S_i ：ライト i に対する鏡面反射色

E：自身の放射光色

ライト i がディセーブルのとき、対応したカラー成分（ D_i , A_i , S_i ）は計算されません。

イネーブルになっているライトの数が多いほど頂点カラー計算の負荷（=Normal コマンドの負荷）が増えますので、必要以上の数のライトをイネーブルにしないように注意してください。

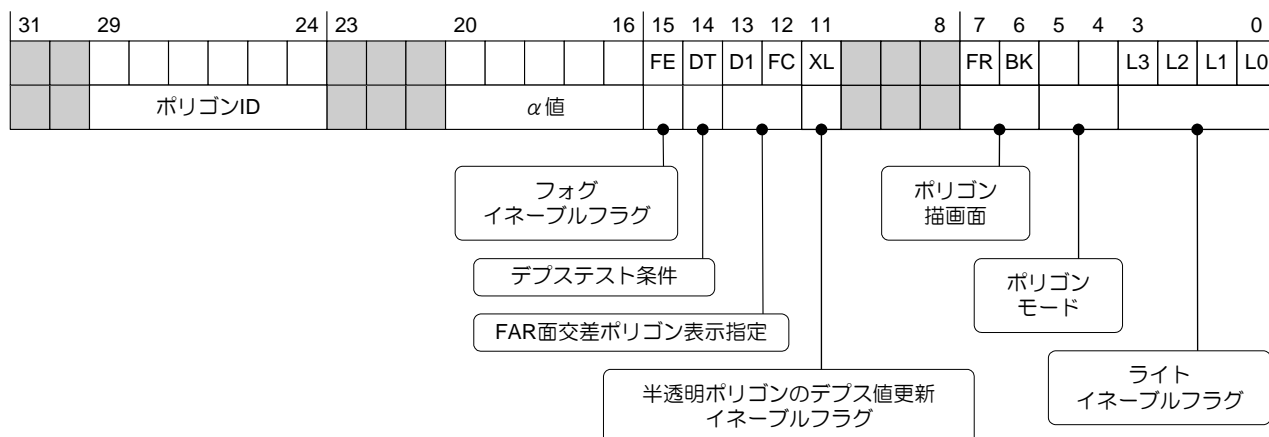
● ライティング OFF 時の頂点カラーについて

ライティングが OFF の場合でも、Normal コマンドが発行されると上式により頂点カラーが計算されるため、結果としてエミッションカラー（放射光色）が頂点カラーにセットされることになります。

7.2.12 ポリゴン属性

PolygonAttr：ポリゴン関連属性値を設定する

名称 POLYGON_ATTR アドレス 0x040004A4 属性 W コマンドコード 0x29



- [d29～d24]：ポリゴン ID

ポリゴン ID はレンダリングエンジンにおいて不透明ポリゴンと半透明ポリゴンの描画時に、それぞれ別々のアトリビュートバッファへ格納されます。

格納されたポリゴン ID は、半透明ポリゴン、シャドウポリゴンおよびエッジマーキング描画時に使用されます。

詳細は「レンダリングエンジン」の「ラスタライズ」を参照してください。

- [d20～d16]： α 値

1～31	ポリゴンの不透明度
0	ワイヤーフレーム表示

1 ≤ α ≤ 30 の場合は「半透明ポリゴン」、 α = 31 の場合は「不透明ポリゴン」と呼びます。

α = 0 にした時点でワイヤーフレーム表示となり、 α の本来の意味ではなくなります。

- FE[d15]：フォグイネーブルフラグ

イネーブルの場合、レンダリングエンジンでフォグブレンディングが行われます。

フォグブレンディングについては「レンダリングエンジン」の「フォグブレンディング」を参照してください。

0	ディセーブル
1	イネーブル

- DT[d14]：デプステスト条件

1 設定時、既に描画されているポリゴンに他のポリゴンを貼り付けることができます（デカルポリゴン）。

0	フラグメントのデプス値がデプスバッファのデプス値よりも小さい場合に描画
1	フラグメントのデプス値がデプスバッファのデプス値と等しい場合に描画

- [d13～d12]：ポリゴン表示指定

- D1[d13]：1 ドットポリゴンのレンダリング指定

ジオメトリ演算の結果、ポリゴンを構成する全頂点の座標 (x, y) が同一座標に集約されたポリゴン (= 1 ドットポリゴン) について、レンダリングエンジンに渡すかどうかを制御することができます。

0	1 ドットポリゴンになったらレンダリングしない
1	1 ドットポリゴンになってもレンダリングする

1 をセットした場合、1 ドットポリゴンは常にポリゴンリスト RAM および頂点 RAM へ書き込まれます。

0 をセットした場合、1 ドットポリゴンのデプス値によってポリゴンリスト RAM および頂点 RAM へ書き込まれるか捨てられるかが制御されます。

1 ドットポリゴンの表示境界デプス値は Disp1DotDepth レジスタで設定してください。

- FC[d12]：FAR 面交差ポリゴン表示指定

0	FAR 面と交差したら消去する
1	FAR 面と交差したらクリッピングする

FAR 面でクリッピングを行う場合、ジオメトリエンジンの負荷が大きくなります。

●XL[d11]: 半透明ポリゴンのデプス値更新イネーブルフラグ

α 値が 1~30 のポリゴンを描画する際、デプスバッファを更新するかどうか選択します。

1 を設定することにより半透明ポリゴンの描画領域にフォグが掛かり過ぎる状況を改善できる場合があります。

ただし背後のエッジマーキングが正しく描画されないことがありますので注意してください。

0	半透明ポリゴン描画時にデプスバッファを更新しない
1	半透明ポリゴン描画時にデプスバッファを更新する

●[d07~d06]: ポリゴン描画面指定

頂点を反時計回りに辿った面が表面になります。

●FR[d07]: 表（おもて）面の描画指定

0	ディセーブル
1	イネーブル

●BK[d06]: 裏（うら）面の描画指定

0	ディセーブル
1	イネーブル

描画指定をディセーブルに指定している場合に、指定している面が表示される面にある場合、このポリゴンはリスト RAM に書き込まれません。

四角形ポリゴンにおいて、最初の 3 頂点のいずれかに同一座標がある場合は重複がハードウェアで検出され、表裏関係なく常に表示されますので注意してください。また、最初の 3 頂点が重複せずに一直線上にある場合は内部計算の精度上の問題により一直線上でなくなることがあります。この場合カメラ等の状況によっては表面に判定されたり、裏面に判定されたりします。この問題を回避するには、下記対策が考えられます。

- ・ 頂点を送る順番を変える
- ・ 2 番目の座標値を 1 番目もしくは 3 番目の座標値と同じ値にする（最初の 3 頂点（1~3）のうち 1 と 2 もしくは 2 と 3 を同じ値にする）
- ・ 三角形に分割する

また、ポリゴンの頂点座標を重複させて線分を描画した場合、うらおもての判定が不可能となり、このフラグの設定に関わらず常に表示されます。

●PM[d05~d04]: ポリゴンモード

モジュレーションおよびデカルモードはテクスチャカラーとフラグメントカラーとのブレンディング方法です。

トゥーン／ハイライトシェーディングはフラグメントカラーのテーブルによる変換方法です。

シャドウポリゴンはステンシルバッファを使用した影付け機能です。

詳細については「レンダリングエンジン」のそれぞれの章を参照してください。

00	モジュレーションモード
01	デカルモード
10	トゥーン／ハイライトシェーディング
11	シャドウポリゴン

●トゥーン／ハイライトシェーディングについて

トゥーンおよびハイライトシェーディングに使用するテーブルは共通です。

DISP3DCNT レジスタによってトゥーン／ハイライトのどちらか一方を選択してください。

DISP3DCNT レジスタへ書き込んだ設定はフレームの切り替わり時に有効になるため、同一の描画フレーム内において両者を混在させることはできません。

●L3~L0[d03~d00]: ライトイネーブルフラグ

ライト 0~3 について個別に設定するフラグです。

0	ディセーブル（ライト OFF）
1	イネーブル（ライト ON）

PolygonAttr コマンド発行位置

PolygonAttr コマンドの設定値は Begin コマンド発行時に有効となり、その後続く頂点の属性として使用されます。PolygonAttr コマンドは Begin-End 間では発行しないでください。
また、ライトイネーブルフラグについては Begin コマンドで設定値を有効にただけでは頂点カラーに反映されません。Begin コマンドで設定値を有効にした後、Normal コマンドでライティング（照光処理）をさせて初めて頂点カラーに反映されます。

Disp1DotDepth：1 ドットポリゴン表示境界デプス値レジスタ

名称 DISP_1DOT_DEPTH アドレス 0x04000610 属性 W 初期値 0x7FFF

15	14	8	7	0
	INTEGER_W			DECIMAL_W
	W 座標			

固定小数点数（整数部 12 ビット+小数部 3 ビット）

●W 座標[d14～d00]：デプス値

PolygonAttr コマンドの「1 ドットポリゴンのレンダリング指定フラグ」が 0 の場合、ジオメトリエンジンは次のようにこのレジスタを参照します。
ポリゴンの全頂点の X および Y 座標が BG スクリーン座標の 1 ドット以下の範囲内へ座標変換される場合、最も小さい W 値（デプス値）がこのレジスタ設定値より大きければ、ポリゴンリスト RAM および頂点 RAM にポリゴンデータを書き込みません（結果として表示されません）。
Z バッファリング時であっても W 値が参照されます。

7.2.13 ポリゴン

Begin：頂点リストの開始を宣言する

名称 BEGIN_VTXS アドレス 0x04000500 属性 W コマンドコード 0x40

31	24	23	16	15	8	7	1	0
								TYPE
								タイプ

● TYPE[d01～d00]：プリミティブタイプ

00	三角形ポリゴン
01	四角形ポリゴン
10	連結三角形ポリゴン
11	連結四角形ポリゴン

連結ポリゴンでは頂点が共有されるため、同じ図形であれば単独ポリゴンよりも頂点 RAM の消費が抑えられます。プリミティブタイプによって、Vertex コマンドによる頂点発行順序と描画の関係は下表のようになります。

● Vertex コマンドによる頂点発行順序と描画の関係

三角形ポリゴン	まず頂点 "v0, v1, v2"で、次に"v3, v4, v5"と一連の三角形を描きます。
四角形ポリゴン	まず頂点 "v0, v1, v2, v3"で、次に"v4, v5, v6, v7"と一連の四角形を描きます。
連結三角形ポリゴン	一連の三角形を、まず頂点 "v0, v1, v2"、それから "v2, v1, v3"、次に "v2, v3, v4"と描いていきます。これは、面の表裏を同じ方向で三角形を描くための順序になっています（図 7-19 参照）。
連結四角形ポリゴン	一連の四角形を、まず "v0, v1, v3, v2"、それから "v2, v3, v5, v4"、次に "v4, v5, v7, v6"と描いていきます。これは、面の表裏を同じ方向で四角形を描くための順序になっています（図 7-19 参照）。

● プリミティブの表（おもて）面の定義

反時計回りが表面になります（v0, v1, v2, …：Vertex 系コマンド発行順）

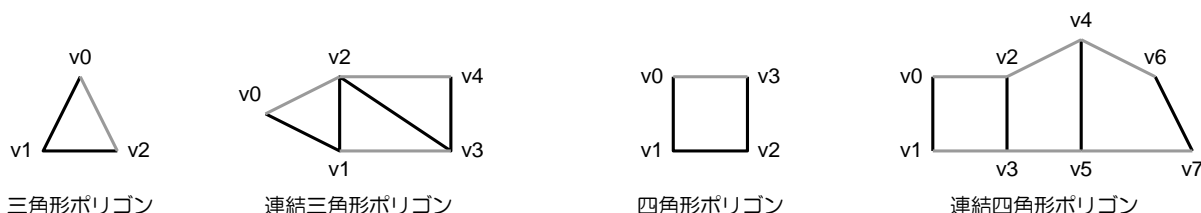


図 7-19 Vertex コマンドによる頂点発行順序

線分を描画したい場合は、上記プリミティブの隣接する頂点を同じ値に設定してください。ただし、線分はうらおもて判定が不可能なため、ポリゴン属性で表示面をディセーブルに指定していても常に描画されます。なお、アンチエイリアシングおよびエッジマーキング機能（「レンダリングエンジン」参照）は図 7-20 のような線分に対しても有効です。

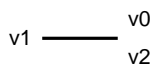


図 7-20 三角形ポリゴンの辺を利用した線分

図 7-21 のような形状の四角形ポリゴンを描画しようとした場合、意図しない結果になることがあります。四角形ポリゴンを描画する際は、図 7-21 のような形状にならない頂点の設定を行ってください。



図 7-21 意図しない結果となる形状の四角形ポリゴン

End : 頂点リストの終了を宣言する

名称 END VTXS アドレス 0x04000504 属性 W コマンドコード 0x41

[illegible]

Begin コマンドと End コマンドは、必ず対になるように発行してください。

Color：頂点カラーを直接設定する

名称 COLOR アドレス 0x04000480 属性 W コマンドコード 0x20

31	24	23	16	15	14	10	9	8	7	5	4	0
						BLUE		GREEN		RED		
						カラー						

頂点カラーは、次の Color、Normal、MaterialColor0（頂点カラーセットフラグ）のいずれかのコマンドによってカレント頂点カラーが更新されるまで有効です。

このため、複数の頂点で頂点カラーを共有することができます。

レンダリングエンジンでは頂点カラーが（R:G:B = 6:6:6）ビットで扱われるため、設定したカラー値が上位 5 ビットに適用されます。

このとき、設定したカラー値が 0 のときは下位 1 ビットが 0 となり、0 以外のときは下位 1 ビットが 1 となります。

また、通常は Begin—End 間で発行しますが、Begin より前で発行することも可能です。

コマンド列の例 1	Begin ⇒ Color ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End
コマンド列の例 2	Color ⇒ Begin ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End ⇒ Begin ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End

Normal : 法線ベクトルを設定する

名称 NORMAL アドレス 0x04000484 属性 W コマンドコード 0x21

31	30	29	28	24	23	20	19	18	16	15	10	9	8	7	0	
		S	NZ				S	NY				S	NX			
法線ベクトル Z 成分				法線ベクトル Y 成分				法線ベクトル X 成分								

符号付き固定小数点数 (符号+小数部 9 ビット)

ライティング（照光処理）は Normal コマンド実行時のみ行われます。

したがって、ライトの ON/OFF 切り替えや、ライトもしくはマテリアルのパラメータ変更後、その効果を頂点カラーへ反映させるためには **Normal** コマンドを再発行する必要があります※。

また、ハードウェアによるベクトルの正規化は行われませんので、単位ベクトルを設定してください。

ライティングによって求められた頂点カラーは、次の Color、Normal、MaterialColor0（頂点カラーセットフラグ）のいずれかのコマンドによってカレント頂点カラーが更新されるまで有効です。

このため、実用上は複数の頂点で法線ベクトルを共有することができます。

※ライトの ON/OFF に関しては、PolygonAttr コマンドで設定した後、Begin コマンドで設定値を有効にしてから Normal コマンドを発行してください。

●Normal コマンド発行位置

通常は Begin-End 間で発行しますが、Begin より前で発行することも可能です。

コマンド列の例 1	Begin ⇒ Normal ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End
コマンド列の例 2	Normal ⇒ Begin ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End ⇒ Begin ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End

名称 VTX 16 アドレス 0x0400048C 属性 W コマンドコード 0x23

符号付き固定小数点数 (符号+整数部 3 ビット+小数部 12 ビット)

名称 VTX 10 アドレス 0x04000490 属性 W コマンドコード 0x24

符号付き固定小数点数 (符号+整数部 3 ビット+小数部 6 ビット)

名称 VTX XY アドレス 0x04000494 属性 W コマンドコード 0x25

符号付き固定小数点数 (符号+整数部 3 ビット+小数部 12 ビット)

名称 VTX XZ アドレス 0x04000498 属性 W コマンドコード 0x26

符号付き固定小数点数 (符号+整数部 3 ビット+小数部 12 ビット)

名称 VTX YZ アドレス 0x0400049C 属性 W コマンドコード 0x27

符号付き固定小数点数（符号+整数部 3 ビット+小数部 12 ビット）

VertexDiff：頂点座標を最後に設定したデータの差分値で指定する

名称 VTX_DIFF アドレス 0x040004A0 属性 W コマンドコード 0x28

31	29	28	24	23	20	19	18	16	15	10	9	8	7	0	
		SZ	DECIMAL_Z				SY	DECIMAL_Y				SX	DECIMAL_X		
		Z 座標				Y 座標				X 座標					

符号付き固定小数点数（符号+小数部 9 ビット）

最後に設定した頂点データの値と加算した値を、新たな 16 ビットの頂点座標として設定します。

VertexDiff コマンドのデータは 16 ビットに符号拡張された上で、直前に設定した頂点座標に加算されます。
頂点座標（Vertex 系コマンドのデータ）は 16 ビット（符号+整数部 3 ビット+小数部 12 ビット）のため、VertexDiff コマンドのデータは頂点座標における小数点第 4 位～第 12 位にあたります（図 7-22 参照）。
なお、加算時にオーバーフローが発生する可能性がありますので注意してください。

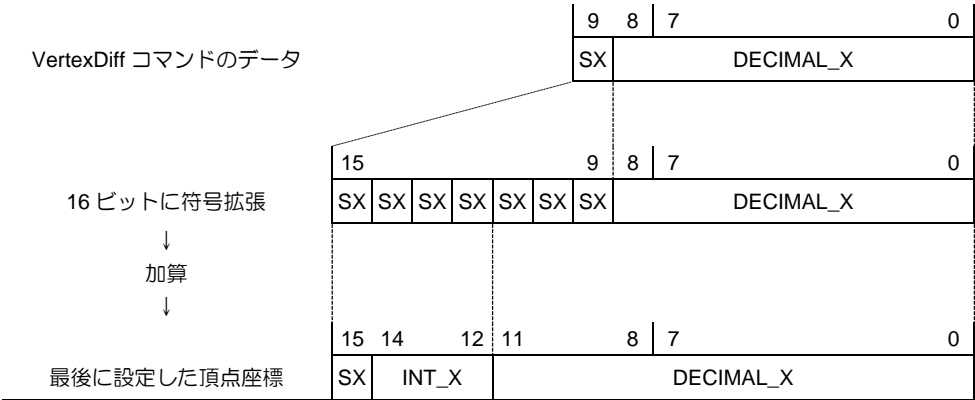


図 7-22 X 座標の加算処理

● **Vertex 系コマンド共通事項**

Vertex 系のコマンド発行時には、頂点 RAM へ BG スクリーン座標系に変換された頂点データが格納されます。
また、ポリゴンを構成する数の頂点データを処理した時点でポリゴンリスト RAM にポリゴンデータも格納されます。
Vertex 系のコマンドは必ず Begin-End 間で発行し、指定したプリミティブに対して頂点数の過不足がないようにしてください。

●クリッピング時のポリゴンにおける注意点

ポリゴンをクリッピングしたときにクリッピング面のポリゴンの頂点カラーの G 値、B 値のいずれか、または共に 0 となり、色化けを起こすことがあります。図 7-23 はその現象が発生したときの状態を表しています（G 値および B 値が共に 0 となった場合の図です）。これは、クリッピングにより新たに生成された頂点のカラーを算出する際、カラー値が 31 より高い値になることがあり、この場合 32 に丸められてしまった値の下位 5 ビットを最終カラーとしてしまうため、誤って 0 になってしまうためです。R 値だけは計算精度が高いため、この問題は発生しないことから、赤色がかった表示になっています。

※TWL モードでジオメトリ回路の回路修正を有効にした場合は、この現象は発生しません（SCFG_EXT レジスタで設定します）。

この現象は、以下の方法で回避できます。

- ポリゴンのスケーリングを小さくしてジオメトリ計算精度を確保する。
- ポリゴンの頂点間隔を短くしたり、頂点をクリッピング面から遠ざけることで低い計算精度による誤差の影響を低減させる。
- モデリングソフト等で頂点カラーを直接設定している場合、頂点カラーを $(R,G,B) = (31,30,30)$ 以下になるように設定する。
- 頂点カラーを直接設定していない場合、マテリアルやライトのカラーを調整し、算出される頂点カラーが $(R,G,B) = (31,30,30)$ 以下になるように設定する。

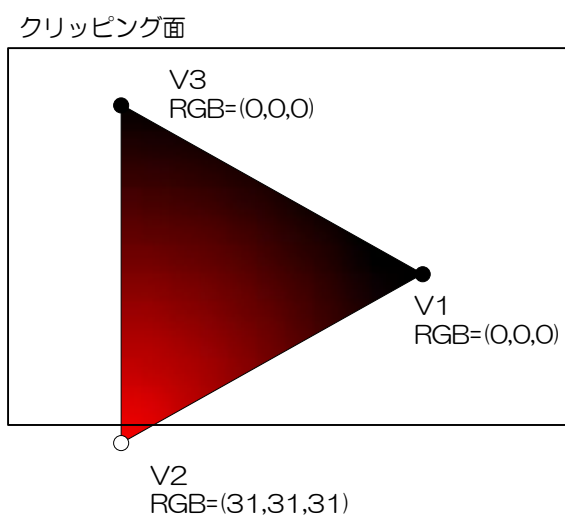


図 7-23 ポリゴンクリッピング時の色化け

7.2.14 テクスチャマッピング

TexCoord：テクスチャ座標を設定する

名称 TEXCOORD アドレス 0x04000488 属性 W コマンドコード 0x22

31	30	24	23	16	15	14	8	7	0
ST	INTEGER_T			DECIMAL_T	SS	INTEGER_S			DECIMAL_S
T 座標					S 座標				

符号付き固定小数点数（符号+整数部 11 ビット+小数部 4 ビット）

●TEX_T, TEX_S[d31～d16], [d15～d00]：テクスチャ座標

テクスチャ座標は、テクセルサイズを 1.0（小数部 4 ビット）とみなし、図 7-24 のようにテクスチャイメージ空間の座標を設定します。

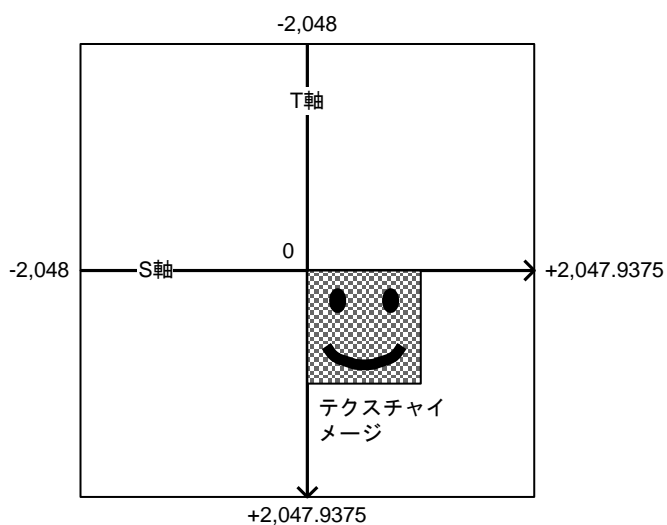


図 7-24 テクスチャイメージ空間図（テクスチャイメージが 1,024×1,024 テクセルの場合）

テクスチャ座標は、次の TexCoord コマンドでカレントテクスチャ座標が再設定されるまで有効です。このため、複数の頂点でテクスチャ座標を共有することができます。

●TexCoord コマンド発行位置

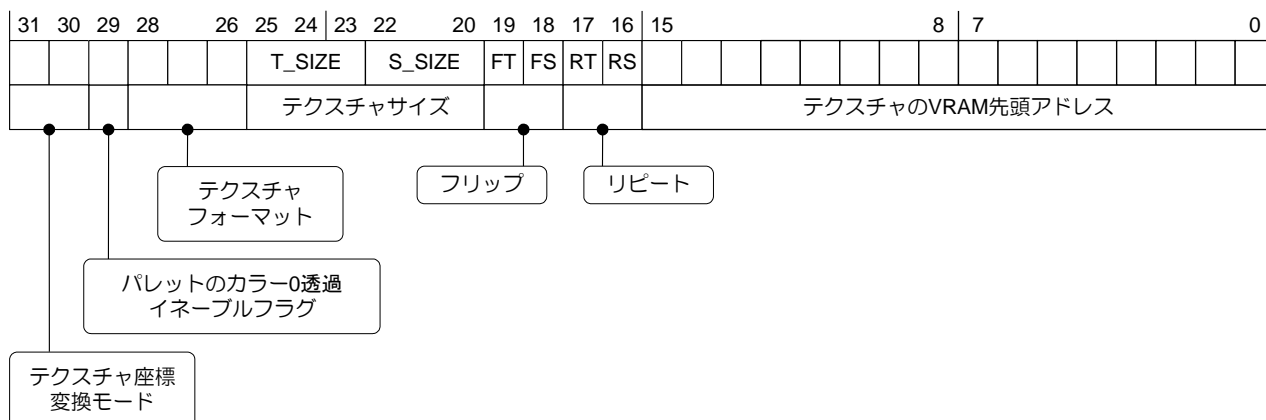
通常は Begin—End 間で発行しますが、Begin より前に発行することもできます。

コマンド列の例 1	Begin ⇒ TexCoord ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End
コマンド列の例 2	TexCoord ⇒ Begin ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End ⇒ Begin ⇒ Vertex ⇒ Vertex ⇒ Vertex ⇒ End

テクスチャマッピングを行う場合、TexCoord ⇒ Normal ⇒ Vertex の順でコマンドを発行すれば、ジオメトリエンジンが高速に処理できます。

TexImageParam : テクスチャのパラメータを設定する

名称 TEXIMAGE_PARAM アドレス 0x040004A8 属性 W コマンドコード 0x2A



● TGEN[d31~d30] : テクスチャ座標変換モード

00	テクスチャ座標変換なし
01	TexCoord ソース
10	Normal ソース
11	Vertex ソース

● TR[d29] : パレットのカラー0 透過イネーブルフラグ
4, 16, 256 色パレットテクスチャで透明テクセルを使用する場合は、1 を設定しパレットのカラー0 を透明色として参照できるようにしてください。

0	パレットのカラー0 設定値が有効
1	パレットのカラー0 を設定値に関わらず、透明とみなす

● TEXFMT[d28~d26] : テクスチャフォーマット

0	テクスチャなし
1	A3I5 半透明テクスチャ
2	4 色パレットテクスチャ
3	16 色パレットテクスチャ
4	256 色パレットテクスチャ
5	4×4 テクセル圧縮テクスチャ
6	A5I3 半透明テクスチャ
7	ダイレクトテクスチャ

● [d25~d20] : テクスチャサイズ

● T_SIZE, S_SIZE

8×8~1,024×1,024 のテクスチャサイズを選択できます。

0	8 テクセル
1	16 テクセル
2	32 テクセル
3	64 テクセル
4	128 テクセル
5	256 テクセル
6	512 テクセル
7	1,024 テクセル

● [d19~d18] : フリップ

テクスチャ座標がテクスチャサイズの範囲外に該当したとき、テクスチャイメージを上下／左右に反転してマッピングするかどうかを指定します(図 7-25、図 7-26 参照)。

リピートが設定されているときにのみフリップ指定が有効になります。

● FT[d19] : T 座標方向のフリップ

0	フリップしない
1	フリップする

● FS[d18] : S 座標方向のフリップ

0	フリップしない
1	フリップする

● [d17~d16] : リピート

テクスチャ座標がテクスチャサイズの範囲外に該当したとき、テクスチャイメージを繰り返してマッピングするかどうかを指定します。

● RT[d17] : T 座標方向のリピート

0	リピートしない
1	リピートする

● RS[d16] : S 座標方向のリピート

0	リピートしない
1	リピートする

● TEX_ADDR[d15~d00] : テクスチャの VRAM 先頭アドレス
3 ビット左シフトしてシステムが参照します

● TexImageParam コマンド発行位置について

通常は Begin より前で発行しますが、Begin—End 間で発行することもできます。

Begin—End 間で発行すれば、Begin—End 内のポリゴン毎に異なるテクスチャパラメータを設定することができます。

注意事項

ジオメトリエンジン内部の処理状況によっては TexImageParam コマンドで与えたパラメータが前のポリゴンのテクスチャ属性を上書きしてしまう問題があります。

※TWL モードでジオメトリ回路の回路修正を有効にした場合は、この現象は発生しません（SCFG_EXT レジスタで設定します）。

TexImageParam コマンドの発行位置を以下に示します。

● Begin—End 外の場合

```
TexImageParam;
Begin;
    TexCoord;
    Vertex;
    ...
End;
```

● 四角形ポリゴンで Begin—End 内の場合

(1)2 個目以降でテクスチャを変更するポリゴンのコマンド構成が「TexCoord→Vertex」の場合

```
Begin;
    TexImageParam;
    1 個目のポリゴン;
    TexImageParam;
    Normal;                      ダミーコマンドとして Normal コマンドを送ってください
    2 個目のポリゴン
    ...
End;
```

(2)2 個目以降でテクスチャを変更するポリゴンのコマンド構成が「TexCoord→Normal→Vertex」の場合

上記(1)の Normal コマンドの代わりにダミーコマンドとして 0xFF（未定義）を送ることでより効率的に問題を回避できます。ただし、TexImageParam だけでなく、TexPlttBase コマンドも送る場合は、不具合が発生しなくなりますのでダミーコマンドは必要なくなります。

● 三角形ポリゴンで Begin—End 内の場合

(1)2 個目以降でテクスチャを変更するポリゴンのコマンド構成が「TexCoord→Vertex」の場合

上記四角形ポリゴンの(1)の場合と同じです。

(2)2 個目以降でテクスチャを変更するポリゴンのコマンド構成が「TexCoord→Normal→Vertex」の場合

不具合は発生しません。

● 連結三角形ポリゴン、連結四角形ポリゴンで Begin—End 内の場合

この場合は、途中でテクスチャを変更するとき必ず一旦 End で区切るようにしてください。

つまり、TexImageParam コマンド発行位置を下記のように最初の Vertex コマンドの発行前に限定するようにしてください

```
Begin;
    TexImageParam;
    Vertex;
    ...
End;
```


● テクスチャのフリップ／リピート設定（テクスチャイメージが 1,024×1,024 テクセルの場合）

1) リピートなしの場合

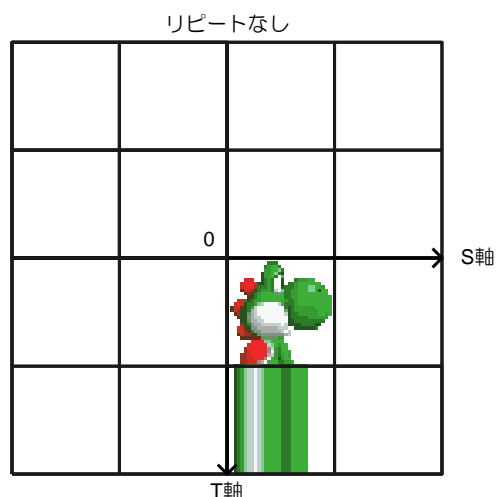


図 7-25 テクスチャイメージ空間図（リピートなし）

2) リピートありの場合のフリップ設定

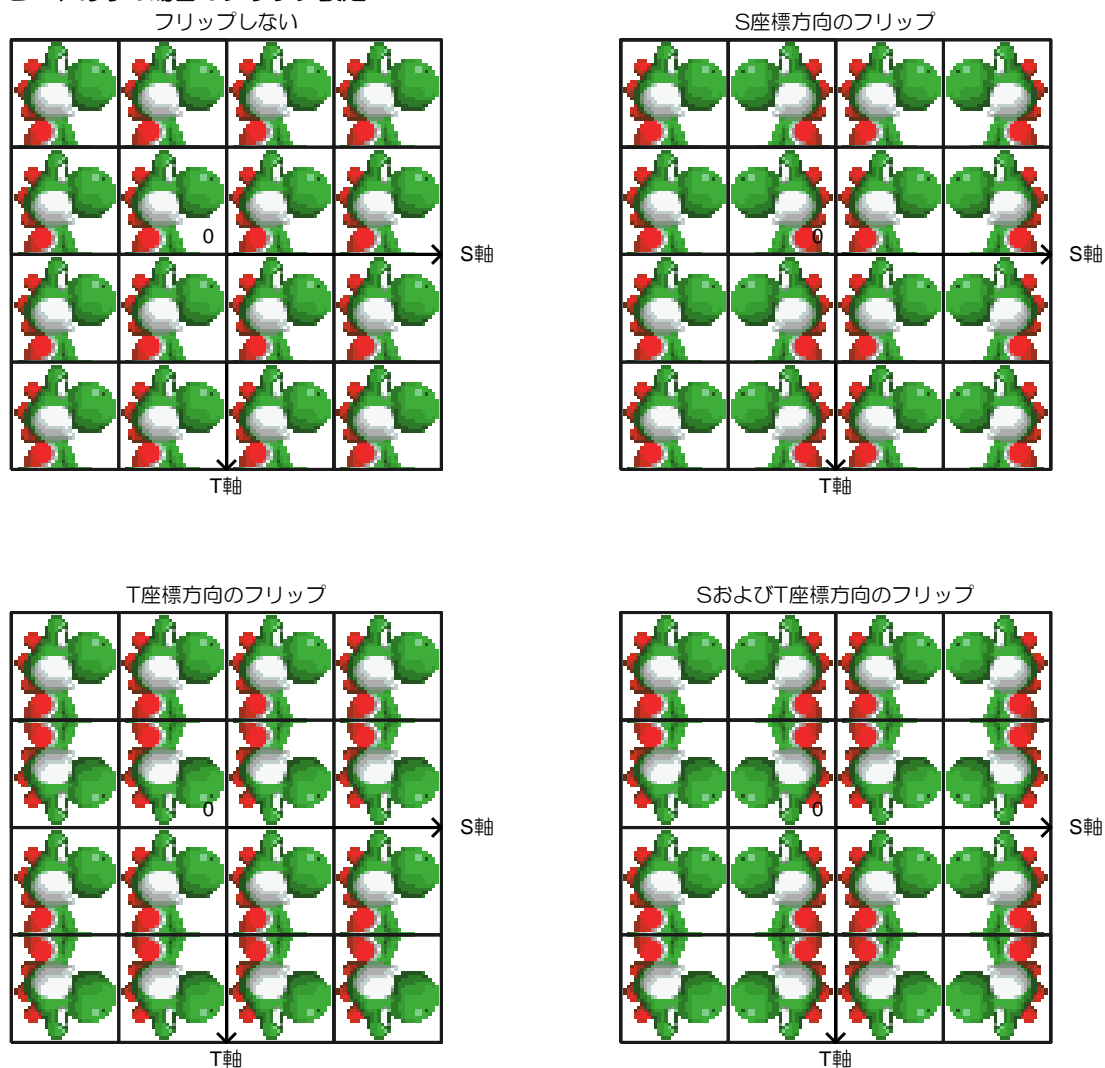


図 7-26 テクスチャイメージ空間図（リピートあり）

7.2.14.1 テクスチャ座標変換

TexImageParam コマンドによって、テクスチャの座標変換モードを切り換えることができます。
 下記に示す 3 つの各モードの入力座標は、該当コマンドで与えた値がそのまま計算に使用されます。
 座標変換後の値が使用されるわけではありませんので、前もってテクスチャ行列を適切に設定しておく必要があります。

1) TexCoord ソース

TexCoord コマンドの設定値を入力座標としてテクスチャ座標変換を行います。
 座標変換は TexCoord コマンド発行時に実行されます。

移動行列や回転行列などをテクスチャ行列に設定すれば、簡単なテクスチャスクロールが実現できます。

● 行列表現による演算式

$$[S' \quad T' \quad R' \quad Q'] = \begin{bmatrix} S & T & \frac{1}{16} & \frac{1}{16} \\ m[0] & m[1] & m[2] & m[3] \\ m[4] & m[5] & m[6] & m[7] \\ m[8] & m[9] & m[10] & m[11] \\ m[12] & m[13] & m[14] & m[15] \end{bmatrix}$$

● 小数点位置まで考慮した具体的な演算式

$$S' = \{m[0] \times (S \ll 12) + m[4] \times (T \ll 12) + m[8] \times (1 \ll 12) + m[12] \times (1 \ll 12)\} \gg 24$$

$$T' = \{m[1] \times (S \ll 12) + m[5] \times (T \ll 12) + m[9] \times (1 \ll 12) + m[13] \times (1 \ll 12)\} \gg 24$$

2) Normal ソース

Normal コマンドの設定値を入力座標としてテクスチャ座標変換を行います。
 座標変換は Normal コマンド発行時に実行されます。
 テクスチャ座標の移動成分には、直前の TexCoord コマンドで設定した S,T 値が適用されます。

カレント方向ベクトル行列を読み出して、-1.0~1.0 の方向ベクトル空間をテクスチャサイズの 1/2 まで拡大するスケール行列を乗算したものをテクスチャ行列に設定すれば、球状の反射マッピングが実現できます。
 このとき、テクスチャ座標の原点を球状テクスチャの中心へ平行移動させるには TexCoord コマンドを用いてください。

● 行列表現による演算式

$$[S' \quad T' \quad R' \quad Q'] = \begin{bmatrix} N_x & N_y & N_z & 1 \\ m[0] & m[1] & m[2] & m[3] \\ m[4] & m[5] & m[6] & m[7] \\ m[8] & m[9] & m[10] & m[11] \\ S & T & m[14] & m[15] \end{bmatrix}$$

● 小数点位置まで考慮した具体的な演算式

$$S' = \{m[0] \times (N_x \ll 3) + m[4] \times (N_y \ll 3) + m[8] \times (N_z \ll 3) + (S \ll 12) \times (1 \ll 12)\} \gg 24$$

$$T' = \{m[1] \times (N_x \ll 3) + m[5] \times (N_y \ll 3) + m[9] \times (N_z \ll 3) + (T \ll 12) \times (1 \ll 12)\} \gg 24$$

3) Vertex ソース

Vertex 系コマンドの設定値を入力座標としてテクスチャ座標変換を行います。

座標変換は Vertex 系コマンド発行時に実行されます。

テクスチャ座標の移動成分には、直前の TexCoord コマンドで設定した S,T 値が適用されます。

カレント位置座標行列を読み出してテクスチャ行列に設定すれば、ビュー座標に依存したテクスチャスクロール等が実現できます。

● 行列表現による演算式

$$[S' \ T' \ R' \ Q'] = [X \ Y \ Z \ 1] \begin{bmatrix} m[0] & m[1] & m[2] & m[3] \\ m[4] & m[5] & m[6] & m[7] \\ m[8] & m[9] & m[10] & m[11] \\ S & T & m[14] & m[15] \end{bmatrix}$$

● 小数点位置まで考慮した具体的な演算式

$$S' = \{m[0] \times X + m[4] \times Y + m[8] \times Z + (S \ll 12) \times (1 \ll 12)\} \gg 24$$

$$T' = \{m[1] \times X + m[5] \times Y + m[9] \times Z + (T \ll 12) \times (1 \ll 12)\} \gg 24$$

● テクスチャ座標変換演算式で用いるパラメータ間の小数点位置について

テクスチャ座標変換の演算式で使われる各パラメータは下表のフォーマットになっています。

このため、テクスチャ座標変換で小数部を 12 ビットに統一して演算するためには、Normal 座標を 3 ビット左シフト、テクスチャ座標を 8 ビット左シフトした上で演算式に適用すれば良いことが分かります。

しかし実際は「小数点位置まで考慮した具体的な演算式」のように、テクスチャ座標は 12 ビット左シフトされた上で演算式に適用され、演算結果を 24 ビット右シフトしたものを新たなテクスチャ座標としています。

このことは、テクスチャ座標のフォーマットを（符号 + 整数部 15 ビット + 小数部 0 ビット）と考えれば自然です。つまり、テクスチャ座標変換はテクセル単位で演算されるのではなく、1/16 テクセル単位で演算されるということになります。

パラメータ名	パラメータ	フォーマット
テクスチャ行列	m[num] (num = 0~15)	符号 + 整数部 19 ビット + 小数部 12 ビット
Normal 座標	Nx, Ny, Nz	符号 + 小数部 9 ビット
Vertex 座標	X, Y, Z	符号 + 整数部 3 ビット + 小数部 12 ビット
テクスチャ座標	S, T	符号 + 整数部 11 ビット + 小数部 4 ビット

● テクニク

ポリゴンで 2D 表現をする場合に、テクセルを LCD 上のピクセルへ 1 対 1 に対応させたい時、テクスチャのサンプリングがテクセルの左上の位置で行われるため、ポリゴンの回転等によってテクスチャが 1 テクセル分ずれてしまうことがあります。

この現象を回避するには、テクスチャ座標変換を適用してテクスチャのサンプリング位置を調整してください。

詳細は、「テクスチャイメージのサンプリング」を参照してください。

● ポリゴン処理サイクル数

ライトが 2 つ以下のときは Normal コマンドの有無によってジオメトリエンジンの実行サイクルは変わりません。ライトが 3 つ以上のときは Normal コマンドなしの方が高速になります。また、ライトの数に関わらず Normal コマンドを発行しない方が、コマンド FIFO への転送時間（バスの占有時間）は短くなります。

なお、下表のような原因は、ライトの数が少ない時には頂点カラーの計算サイクル数が短くても座標変換がボトルネックとなって、ポリゴン処理としては頂点座標の変換に必要な時間に固定となりますが、ライトの数が増えると頂点カラーの計算サイクル数の方が上回ってしまうためです。

コマンド構成	3 角形ポリゴンでのライト数					4 角形ポリゴンでのライト数				
	OFF	1ch	2ch	3ch	4ch	OFF	1ch	2ch	3ch	4ch
TexCoord→Normal→Vertex	28	28	28	30	33	37	37	37	40	44
Normal→Vertex	28	28	28	28	30	37	37	37	37	40
TexCoord→Vertex	28	28	28	28	28	37	37	37	37	37
Vertex	28	28	28	28	28	37	37	37	37	37

7.2.15 テスト

ジオメトリエンジンはテスト関連コマンドを実行すると、テストコマンドの結果（ステータスフラグや結果レジスタの値）を更新します。

BoxTest：直方体が視体積に入るかテストする

名称 BOX_TEST アドレス 0x040005C0 属性 W コマンドコード 0x70

31	30	28	27	24	23	16	15	14	12	11	8	7	0
SY	INT_Y	DECIMAL_Y					SX	INT_X	DECIMAL_X				
Y 座標						X 座標							

符号付き固定小数点数（符号+整数部 3 ビット+小数部 12 ビット）

31	30	28	27	24	23	16	15	14	12	11	8	7	0
SW	INT_W	DECIMAL_W					SZ	INT_Z	DECIMAL_Z				
幅							Z 座標						

31	30	28	27	24	23	16	15	14	12	11	8	7	0
SD	INT_D	DECIMAL_D					SH	INT_H	DECIMAL_H				
奥行き							高さ						

座標値は、図 7-27 に示されるボックスの基準頂点を指定してください。

ボックステストの結果はジオメトリエンジンステータスレジスタ（GXSTAT）に格納されます。

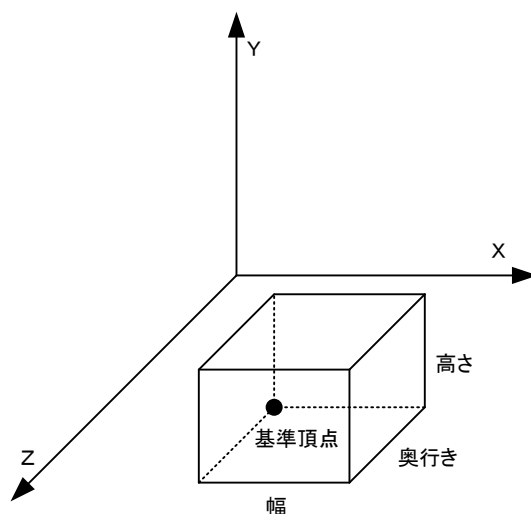


図 7-27 テストされるボックス

● ボックステストのテスト内容

ボックスを構成する 6 面のうち、いずれかの面がビューポリウムにかかるかどうかを判定します。

このため、ビューポリウムがボックスの内に完全に含まれる場合はビュー外の判定となることに注意してください。

※TWL モードでジオメトリ回路の回路修正を有効にした場合は、この現象は発生しません（SCFG_EXT レジスタで設定します）。

● ボックステストを正しく行うための条件

ボックステストは、ポリゴン属性の「FAR 面交差ポリゴン表示指定フラグ」および「1 ドットポリゴンのレンダリング指定フラグ」をどちらも 1 に設定した状態で行ってください（具体的な手順は下記参照）。

いずれかのフラグが 0 の場合、正しいテスト結果が得られない場合があります。

また、基準頂点に幅／高さ／奥行きを加算したときにオーバーフローが発生する可能性がありますので注意してください。（加算結果が-8.0 以上 8.0 未満に納まっている必要があります。）

<具体的な手順>

- 1) PolygonAttr コマンドで「FAR 面交差ポリゴン表示指定フラグ」および「1 ドットポリゴンのレンダリング指定フラグ」をどちらも 1 に設定
- 2) Begin コマンド
- 3) End コマンド
- 4) BoxTest コマンド

PositionTest : テスト用位置座標を設定する

名称 POS_TEST アドレス 0x040005C4 属性 W コマンドコード 0x71

31	30	28	27	24	23	16	15	14	12	11	8	7	0
SY	INT_Y	DECIMAL_Y					SX	INT_X	DECIMAL_X				
Y 座標							X 座標						

符号付き固定小数点数 (符号+整数部 3 ビット+小数部 12 ビット)

31	24	23	16	15	14	12	11	8	7	0
								SZ	INT_Z	DECIMAL_Z
Z 座標										

テスト用位置座標は、カレントクリップ座標行列により座標変換が行われます。
Position テストの結果 (クリップ座標) は PositionResult レジスタに格納されます。

VectorTest : テスト用方向ベクトルを設定する

名称 VEC_TEST アドレス 0x040005C8 属性 W コマンドコード 0x72

31		29	28	24	23	20	19	18	16	15	10	9	8	7	0	
		SZ	DECIMAL_Z					SY	DECIMAL_Y					SX	DECIMAL_X	
		Z 成分					Y 成分					X 成分				

符号付き固定小数点数 (符号+小数部 9 ビット)

テスト用方向ベクトルは、カレント方向ベクトル行列により座標変換が行われます。
Vector テストの結果 (ビュー座標空間での方向ベクトル) は VectorResult レジスタに格納されます。

PositionResult : PositionTest の計算結果を読み出す

名称 POS_RESULT_x (x=X,Y,Z,W)

アドレス 0x04000620, 0x04000624, 0x04000628, 0x040062C 属性 R 初期値 0x00000000

31	30	24	23	16	15	12	11	8	7	0
Sx	INTEGER_x							DECIMAL_x		
符	整数部							小数部		

符号付き固定小数点数 (符号+整数部 19 ビット+小数部 12 ビット)

クリップ座標 (X, Y, Z, W) の各値が入ります。

VectorResult : VectorTest の計算結果を読み出す

名称 VEC_RESULT_x (x=X,Y,Z) アドレス 0x04000630, 0x04000632, 0x04000634 属性 R 初期値 0x0000

15	8	7	0
Sx	INTEGER_x	DECIMAL_x	
符号	整数部	小数部	

符号付き固定小数点数 (符号+整数部 3 ビット+小数部 12 ビット)

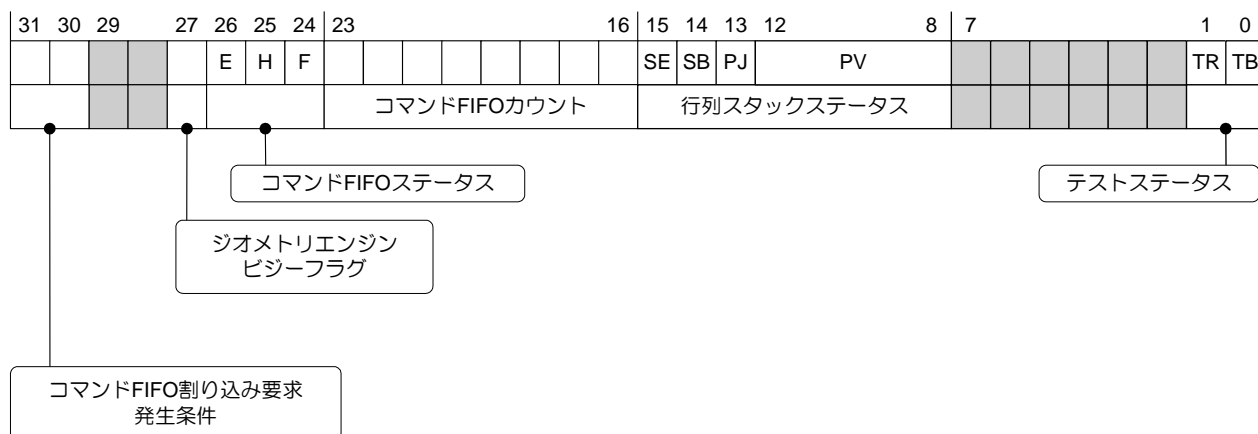
ビュー座標空間での方向ベクトル (X,Y,Z) の各値が入ります。
読み出される計算結果は±1 以下のため、整数部は符号拡張されます。

7.2.16 ステータス

ポリゴンリスト RAM および頂点 RAM の状態については、前述の DISP3DCNT レジスタで確認することができます。コマンド FIFO や行列スタック等については、下記の GXSTAT レジスタで確認することができます。

ジオメトリエンジンステータスレジスタ

名称 GXSTAT アドレス 0x04000600 属性 R/W 初期値 0x00000000



● FI[d31～d30] : コマンド FIFO 割り込み要求発生条件

00	コマンド FIFO 割り込み要求ディセーブル
01	コマンド FIFO が半分未満で割り込み要求
10	コマンド FIFO が空で割り込み要求
11	設定禁止

● B[d27] : ジオメトリエンジンビジーフラグ

0	ジオメトリエンジン停止中
1	ジオメトリエンジン動作中

コマンドやパラメータが送られてこない場合、コマンド待ち/パラメータ待ち状態でビジーフラグが0になります。コマンドやパラメータの供給が再開されると、ジオメトリ処理を再開します。

また、SwapBuffers コマンドを発行すると、その後コマンドを一切発行しなかった場合は V ブランクで開始される SwapBuffers 処理が完了したタイミングでジオメトリエンジンビジーフラグが0になります（それまでは1です）。このタイミングは、V ブランク後 33MHz 換算で 400 サイクル経過後です。

SwapBuffers コマンド発行後に次のコマンドを発行した場合は、V ブランクで開始される SwapBuffers 処理が完了したタイミングで次のコマンドが実行されます。

● [d26～d24] : コマンド FIFO ステータス

● E[d26] : コマンド FIFO エンプティフラグ

0	FIFO が空でない
1	FIFO が空

● H[d25] : コマンド FIFO アンダーハーフフラグ

0	FIFO が半分以上である
1	FIFO が半分未満

● F[d24] : コマンド FIFO フルフラグ

0	FIFO が一杯でない
1	FIFO が一杯

● [d23～d16] : コマンド FIFO カウント値

現在コマンド FIFO に格納されているコマンド/データの数参照できます

● [d15～d08]：行列スタックステータス

● SE[d15]：スタックエラーフラグ

行列スタックのオーバーフロー／アンダーフローが発生した場合に 1 がセットされます。

1 を書き込むことによってクリアできます。

0	スタックオーバーフロー／アンダーフローなし
1	スタックオーバーフロー／アンダーフローあり

行列スタックステータスのスタックエラーフラグを参照する際は、先に行列スタックビジューフラグを参照し発行済みの PushMatrix、PopMatrix コマンドの実行が終了していることを確認してください。

● SB[d14]：行列スタックビジューフラグ

PJ および PV の行列スタックレベルを参照するときは、このフラグを参照し発行済みの PushMatrix、PopMatrix コマンドの実行が終了していることを確認してください。

0	実行前の PushMatrix、PopMatrix コマンドがない
1	PushMatrix、PopMatrix コマンドが発行されており、実行終了していない

● PJ[d13]：Projection 行列スタックのレベル

現在のスタックレベル（0～1）が参照できます。

● PV[d12～d08]：Position・Vector 行列スタックのレベル

現在のスタックレベル（0～31）が参照できます。

● [d01～d00]：テストステータスフラグ

● TR[d01]：ボックステスト結果

0	ボックスを構成するすべての面が視体積の外にある
1	ボックスを構成する 6 面のうち、いずれかの面の一部またはすべてが視体積内にある

● TB[d00]：テストビジューフラグ

各テスト（BoxTest, PositionTest, VectorTest）のレディ／ビジューを参照できます

0	レディ
1	ビジュー

7.2.16.1 ポリゴンリスト RAM と頂点 RAM のデータ格納数

1) ポリゴンリスト RAM

ポリゴンリスト RAM の 52KB の内訳は、前半の ORDER エリア 12KB と後半の POLYGON エリア 40KB に分かれています。

ポリゴンリスト RAM へのポリゴン格納数は、ORDER エリア、POLYGON エリアともにポリゴンが連結されているかどうかには影響されません。

しかし、ポリゴンを連結することは、ポリゴン間の頂点が共有されるため頂点 RAM の消費量を抑えることに関しては有効です。

ポリゴンがどのように描画される状況においても、ポリゴンリスト RAM の ORDER エリアは 2048 ポリゴンが格納可能ですが、POLYGON エリアは状況によって格納可能なポリゴン数に変動します。

POLYGON エリアのポリゴンデータは 12 バイトのヘッダ領域と後続の 8 バイト以上の頂点インデックス領域から成り立っています。

通常、頂点インデックス領域は三角形ポリゴンで 8 バイト、四角形ポリゴンで 12 バイトを消費します。

よって各ポリゴンの最大格納数は下記のようになります。

(連結) 三角形ポリゴン：40KB / (ヘッダ 12 バイト + 8 バイト) = 2048 ポリゴン

(連結) 四角形ポリゴン：40KB / (ヘッダ 12 バイト + 12 バイト) = 1706 ポリゴン

ただし、クリッピングが発生し頂点が増えるたびに頂点インデックス領域の消費量は 4 バイトずつ増えます。

つまり、三角形ポリゴンでは 5 回、四角形ポリゴンでは 6 回クリッピングが発生すると、POLYGON エリアへ登録できるポリゴンが 1 つ減ることになります。

2) 頂点 RAM

頂点 RAM は 72KB の容量を持っており、全領域に頂点データが格納できます。

表 7-6 にプリミティブタイプ別の頂点 RAM 消費量と最大格納可能ポリゴン数を示します。

1 頂点につき 12 バイト消費します。

プリミティブタイプ	頂点 RAM 消費量	ポリゴン数に換算した最大格納可能数
三角形ポリゴン	ポリゴン毎に 3 頂点	2048
四角形ポリゴン	ポリゴン毎に 4 頂点	1536
連結三角形ポリゴン	先頭ポリゴン：3 頂点 後続ポリゴン：1 頂点	6142 ただし、2050 頂点でポリゴンリスト RAM の上限 2048 ポリゴンに達します
連結四角形ポリゴン	先頭ポリゴン：4 頂点 後続ポリゴン：2 頂点	3070 ただし、3414 頂点でポリゴンリスト RAM の上限 1706 ポリゴンに達します

表 7-6 プリミティブタイプ別 頂点 RAM 消費量と最大格納可能ポリゴン数

クリッピングによっても頂点 RAM は消費されるため、ポリゴンはできるだけ連結した方が、頂点 RAM のオーバーフローによって後続のポリゴンが登録できなくなる現象を回避するのに有効です。

2-1) 頂点共有の解除要因

● Z バッファリング時、W バッファリング時に共通した頂点共有の解除要因

連結ポリゴンでクリッピングが発生すると、隣接しているポリゴン間の頂点の共有は解除されます。

● Z バッファリング時にのみ発生する頂点共有の解除要因

また、Z バッファリング時には、連結ポリゴン内の隣接するポリゴン間で頂点 RAM へ格納されるべき W 値（クリップ座標の W 値の 2 倍）が 1 頂点でも 16 ビット（クリップ座標での 15 ビット）の範囲を超えているポリゴンと全く超えていないポリゴンが存在していた場合にも、これらの間での頂点の共有は解除されます。（図 7-28 参照）

これは、Z バッファリング時の頂点 RAM には Z 値と W 値が両方とも格納されることによって W 値の精度が 16 ビットしか確保できなくなるため、W 値 24 ビットの上位 16 ビットか下位 16 ビットのどちらを格納するかがこれらのポリゴン間で異なってしまうためです。

W バッファリング時には、頂点 RAM へ Z 値を格納する必要がないためクリップ座標の W 値 24 ビットがそのまま格納され、このような制限による頂点共有の解除は発生しません。

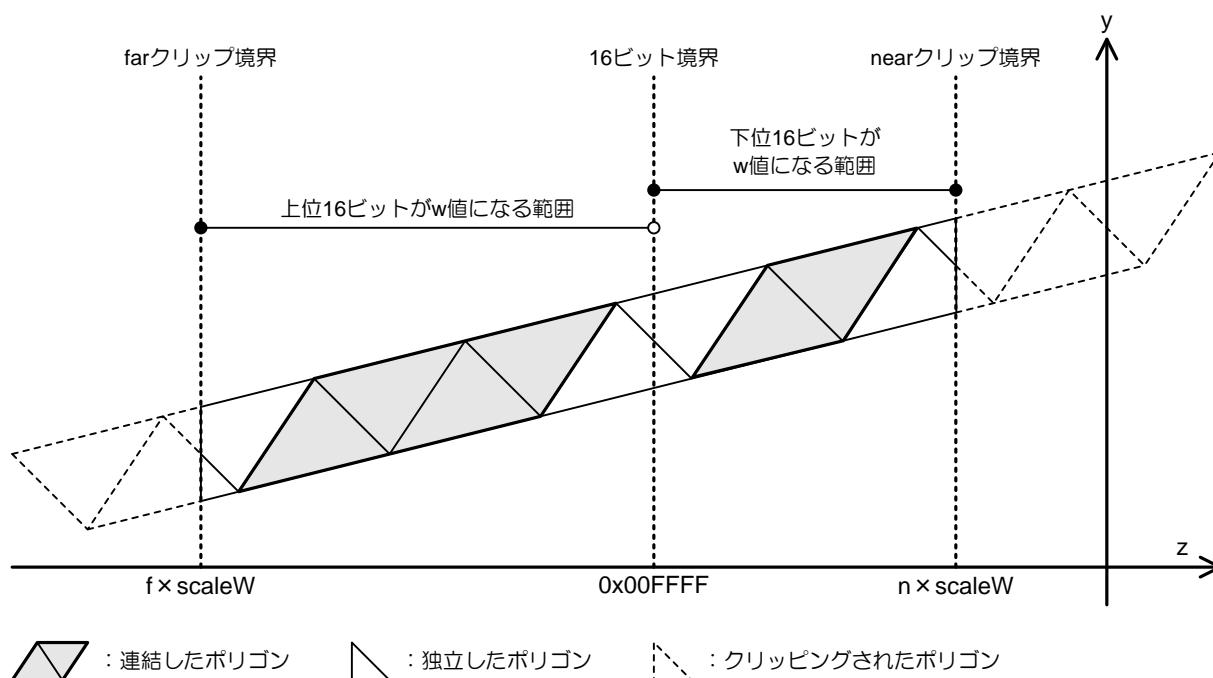


図 7-28 連結ポリゴンの頂点共有解除概念図（クリップ座標系）

● ポリゴン属性の 1 ドットポリゴン描画に関する設定における解除要因

ポリゴン属性で「1 ドットポリゴンを描画しない」に設定し、かつ、ポリゴンの W 値が 1 ドットポリゴン表示境界デプス値レジスタの値を超える場合、連結ポリゴンの頂点共有が解除されます。この場合のポリゴン共有が解除される条件の判定はポリゴン毎に行われるため、1 ドットポリゴン表示境界デプス値レジスタの値を超えないポリゴン群については頂点共有されたままとなります。これを回避するには、連結ポリゴンのポリゴン属性で「1 ドットポリゴンを表示する」に設定してください。

※TWL モードでジオメトリ回路の回路修正を有効にした場合は、頂点共有は解除されません（SCFG_EXT レジスタで設定します）。

3) 各メモリへ格納可能なポリゴン数とメモリ容量

各メモリへ格納可能なポリゴン数とメモリ容量の関係は下式のようになります。

メモリ領域	格納可能なポリゴンとメモリ容量の関係式
ポリゴンリスト RAM の POLYGON 領域	$(12+8) \times (F3+MF3) + (12+12) \times (F4+MF4) + 4 \times CLIP \leq 40K \text{ バイト}$
頂点 RAM	$(12 \times 3) \times (F3+MP3+DMF3) + (12 \times 1) \times (MF3-MP3-DMF3) +$ $(12 \times 4) \times (F4+MP4+DMF4) + (12 \times 2) \times (MF4-MP4-DMF4) +$ $12 \times CLIP \leq 72K \text{ バイト}$

F3：三角形ポリゴン数、MF3：連結三角形ポリゴン数、MP3：連結三角形ポリゴンのプリミティブ数

F4：四角形ポリゴン数、MF4：連結四角形ポリゴン数、MP4：連結四角形ポリゴンのプリミティブ数

CLIP：クリッピング回数

DMF3：連結三角形ポリゴンが頂点解除された回数

DMF4：連結四角形ポリゴンが頂点解除された回数

4) まとめ

ポリゴンリスト RAM と頂点 RAM へ登録可能なポリゴン数のうち、少ない方の制限によって実際に描画されるポリゴン数が決まります。

連結ポリゴンを多用した場合には頂点 RAM にかなり余裕ができますので、ポリゴンリスト RAM の容量制限を中心に気を付ければ良い状況が多いものと推定されます。

連結ポリゴンを多用する場合においては、全ポリゴンで 1 回ずつクリッピングが発生するという通常発生しない状況においても、下記のポリゴン数が確保できることになります。

(連結) 三角形ポリゴン：40K バイト / $(12+8+4) = 1706$

(連結) 四角形ポリゴン：40K バイト / $(12+12+4) = 1462$

7.2.17 計算精度の注意点

TWL ではワールド座標系として 32 ビット空間（符号+整数部 19 ビット+小数部 12 ビット）を指定可能ですが、ハードウェア内部での計算精度も 32 ビットであるため、32 ビット空間の端では物体の歪みや回り込みといった現象が起こります。

問題なく使用するためには、29 ビット（符号+整数部 16 ビット+小数部 12 ビット）の範囲内で使用してください。

また、TWL は 24 ビットの視界空間（符号 1 ビット+整数部 11 ビット+小数部 12 ビット）を持ちます。それ以上の空間を視体積として設定しますと物体の歪みや回り込みが起こりますのでご注意ください。

7.3 レンダリングエンジン

TWL モードではシステム設定によって、NITRO で問題の発生していた以下の機能について修正された回路を使用することができます。ただし、NITRO 互換モードでは、注意事項に従って問題を回避する必要があります。

- シャドウ：「シャドウポリゴン」を参照してください。
- 4×4 テクスセル圧縮テクスチャ：「4×4 テクスセル圧縮テクスチャ」を参照してください。

7.3.1 概要

レンダリングエンジンの仕様一覧を表 7-7 に示します。

動作周波数	33.514MHz
描画データ	三角形および四角形
描画能力	最大 12 万ポリゴン／秒（60 fps） 最大 3000 万ピクセル／秒（60 fps）
陰面処理	Z 値バッファリング、W 値バッファリング方式切り換え
シェーディング	グーローシェーディング
テクスチャマッピング	透視補正、モジュレーション／デカル 4×4 テクスセル圧縮方式サポート、半透明テクスチャサポート フリップ、リピート イメージサイズ 8×8 テクスセル～1,024×1,024 テクスセル
その他の機能 （表 7-8 参照）	α ブレンディング、 α テスト、アンチエイリアシング、 エッジマーキング、フォグ、トゥーンシェーディング、 ハイライトシェーディング、シャドウ、ワイヤーフレーム、 クリアイメージ

表 7-7 レンダリングエンジンの仕様一覧

レンダリングエンジンの各機能の概要を表 7-8 に示します。

α ブレンディング	カラーバッファに格納されるカラー値と、入力されたフラグメント※のカラー値をフラグメント※の α 値を元にしてブレンディングします。 (※ここではテクスチャブレンディング後のフラグメントを指します)
α テスト	フラグメント※の α 値とレジスタに設定された参照値を比較し、フラグメント※の α 値の方が大きい時のみ描画します。 (※ここではテクスチャブレンディング後のフラグメントを指します)
アンチエイリアシング	ポリゴン境界に対して、本来の表示位置からのずれを元に算出される係数(5bit 幅)で後ろにあるポリゴンのカラー値と混合します。
エッジマーキング	ポリゴン ID (6bit) が異なるポリゴン境界に対してポリゴンエッジを指定色(8 色)でマーキングします。アンチエイリアシングが有効の場合はエッジマーキング処理した後にアンチエイリアシングが処理されます。
フォグ	フォグ濃度テーブルを使用して、設定されたフォグのカラー値とカラーバッファのカラー値をブレンディングします。フォグ濃度は 32 段階設定することができ、対象となるピクセルのデプス値によって線形補間した値が適用されます。 2D の前面に 3D を表示させる場合、カラーバッファを利用することで 2D 面にもフォグをかけることができます。
トゥーンシェーディング	輝度計算の結果を急峻にすることでアニメ調の絵を表現することが可能です。
ハイライトシェーディング	テクスチャカラー以上の輝度を表現することが可能です。
シャドウ	シャドウボリュームを定義することで凹凸面にも簡単に影付けすることが可能です。
ワイヤーフレーム	ポリゴンの面を描画せず、辺のみ描画することができます。
クリアイメージ	カラーバッファ、デプスバッファ、アトリビュートバッファのフォグイネーブルフラグの初期値として、VRAM 上のクリアイメージを適用することができます。

表 7-8 レンダリングエンジンの各機能の概要

注意事項

レンダリング関連の各レジスタはダブルバッファ構成となっており、SwapBuffers コマンドを発行した直後の V ブランク期間開始時に各レジスタの内容がレンダリングエンジンへ送られます。

そのため、フレームの途中であっても、これらのレジスタにデータを書き込むことができますが変更した時点のフレームにおける描画イメージには反映されませんので注意してください。

7.3.2 レンダリング方式

7.3.2.1 ラインバッファレンダリング

TWL ではフレームバッファ方式ではなくラインバッファ方式を採用しているため、1 ライン上に数多くのポリゴンが重なると、水平期間内にレンダリングを完了できない（ラインズオーバー）可能性があります。

このため、カラーバッファを 48 ライン分持ち FIFO 動作をさせ、V ブランク期間の途中からレンダリングを開始し表示前のラインデータを描き貯めておくことでラインズオーバーが容易には発生しないようになっています（図 7-29）。

RDLINES_COUNT レジスタを読み出すことで、描画し終わったフレーム表示中にカラーバッファが最低何ラインまでになったかを確認することができます。

つまり、ラインズオーバー発生の有無を確認することはできませんがラインズオーバーの発生リスクがどのくらいであったかを知ることができます。

RDLINES_COUNT レジスタについては「ステータス」を参照してください。

実際には、レンダリングエンジンはカラーバッファに対して読み書きを行います。ここでは FIFO 動作のみ図 7-29 に示します。

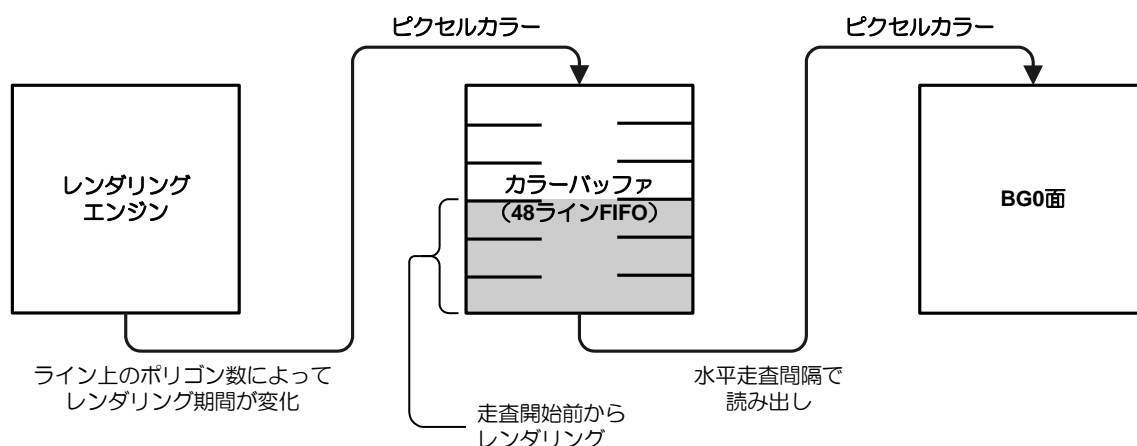


図 7-29 カラーバッファの FIFO 動作

7.3.2.2 レンダリングエンジン内のバッファ

表 7-9 の通り、レンダリングエンジンにはピクセル毎の情報を格納するバッファがあります。

「3D グラフィックス」のハードウェア・ブロック図（図 7-1）も参照してください。

ステンシルバッファ	1 ビット／ピクセルのバッファが 1 ライン分あります。 シャドウポリゴンの描画時に使用されます。
アトリビュートバッファ	23 ビット／ピクセルのバッファが 2 ライン分あります。 ピクセル毎に、ポリゴン ID とフォグイネーブルフラグを格納するバッファです。 ポリゴン ID については、不透明ポリゴンと半透明ポリゴンで別々に管理されます。
デプスバッファ	24 ビット／ピクセルのバッファが 2 ライン分あります。 デプステストやフォグブレンディングの計算に使用されます。
カラーバッファ	23 ビット／ピクセル（R:G:B:A = 6:6:6:5）のバッファが 48 ライン分あります。

表 7-9 レンダリングエンジン内のバッファ

7.3.2.3 ブランク期間

レンダリングエンジンは、LCD の 214 ライン走査開始時から描画を開始します。
LCD が最終 191 ライン目の表示を始めるまではレンダリングエンジンが描画を行っているため、レンダリングエンジンのブランク期間は、LCD191～213 の計 23 ラインとなっています（図 7-30 参照）。
レンダリングエンジンが描画中に参照する VRAM 領域（テクスチャイメージ、テクスチャパレット）を安全に書き換えるには、この 23 ライン内で行ってください。

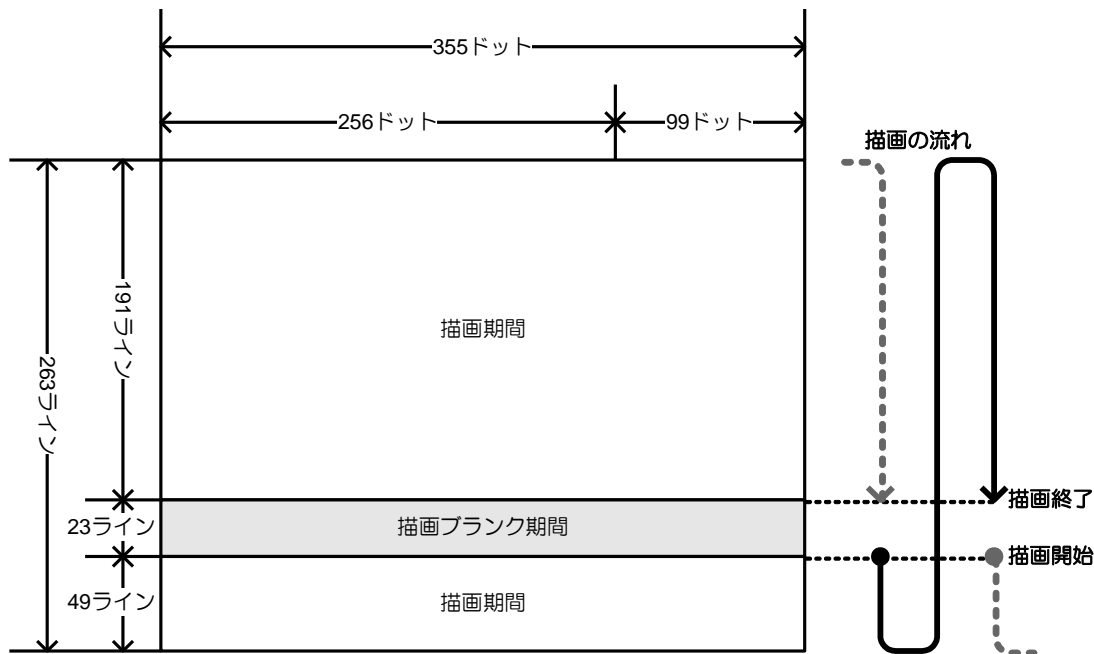


図 7-30 レンダリングエンジンのブランク期間

レンダリングエンジンのタイミング仕様を表 7-10 に示します。

項目		スペック	期間
描画期間	水平ドット数	256 ドット	45.8316us
	垂直ライン数	49+191 ライン	15.2533ms
総期間	水平ドット数	355 ドット	63.5556us
	垂直ライン数	263 ライン	16.7151ms
ブランク期間	ブランクライン数	23 ライン	1.4618ms
走査周期	V 周期	59.8261Hz	16.7151ms

表 7-10 レンダリングエンジン タイミング仕様

7.3.2.4 1ラインで描画可能なポリゴン数

レンダリングエンジンが1ラインで描画できる期間は355ドットです（表 7-10 参照）。

1ドットは6サイクルに相当し、ライン毎に4サイクルのオーバーヘッドがかかるため、レンダリングに使用できるサイクル数は $355 \times 6 - 4 = 2126$ サイクルとなります。

TWL は、テクスチャマッピングされた半透明ポリゴンでも1ピクセル/サイクルのフィルレートがありますが、ポリゴンのレンダリング開始毎に8サイクルのオーバーヘッドがかかります。

よって、1ラインでの描画を最低限保証できるポリゴン数は下記ようになります。

$$(\text{1ライン描画保証ポリゴン数}) = (2126 \text{ サイクル}) / (8 \text{ サイクル} + \text{ポリゴンの水平方向のピクセル数})$$

また、この場合のオーバーヘッド分を除いたライン単位フィルレートは次のようになります。

$$(\text{ライン単位フィルレート}) = (2126 \text{ サイクル}) - (8 \text{ サイクル} \times \text{1ライン描画保証ポリゴン数})$$

参考）上式の結果を表にすると表 7-11 のようになります

水平方向の ピクセル数	8	16	32	64	128	256
1ライン描画保証 ポリゴン数	132	88	53	29	15	8
ライン単位の フィルレート	1070	1422	1702	1894	2006	2062

表 7-11 1ラインで描画可能なポリゴン数とフィルレート（計算値）

ラインバッファはFIFOになっているため、実効値としてはもっと多くのポリゴンが描画可能です。

7.3.3 レンダリングバッファ初期化

7.3.3.1 クリアレジスタによる初期化

ClearColorAttr：クリアカラーアトリビュートレジスタ

名称 CLEAR_COLOR アドレス 0x04000350 属性 W 初期値 0x00000000

31	29	24	23	20	16	15	14	10	9	8	7	5	4	0
						F		BLUE		GREEN			RED	
		クリアポリゴン ID						α 値	霧	カラー				

- クリアポリゴン ID [d29～d24]：ポリゴン ID 初期値
アトリビュートバッファにおける不透明ポリゴン ID の初期値を設定できます。
エッジマーキングにおいて、画面端とクリッピングでできた辺に関しては、ポリゴン ID とこのクリアポリゴン ID とを比較しエッジマーキングが適用されるかどうかが決まります。
- α 値[d20～d16]： α 初期値
カラーバッファの α 初期値を設定できます。
通常、2D との合成時には 0 を指定してください。
- F[d15]：フォグイネーブルフラグ
アトリビュートバッファのフォグイネーブルフラグ初期値を設定できます。
2D 面との合成時に、背後のプレーンに対してフォグを適用するかどうかをコントロールできます。
2D の背景を鮮明に表示したい場合に有効です。
- カラー[d14～d00]：クリアカラーRGB 値
カラーバッファの RGB 初期値を設定できます。
レンダリングエンジン内のカラーバッファは (R:G:B = 6:6:6) ビットのため、クリアカラー値が 0 のときは下位ビットが 0 として扱われ、クリアカラー値が 0 以外のときは下位 1 ビットが 1 として扱われます。

ClearDepth：クリアデプスレジスタ

名称 CLEAR_DEPTH アドレス 0x04000354 属性 W 初期値 0x7FFF

15	14	8	7	0
	CLEARDEPTH			
	クリアデプス値			

- CLEARDEPTH[d14～d00]：クリアデプス値
レンダリングエンジン内のデプスバッファは 24 ビット/ピクセルのため、クリアデプス値は 9 ビット左シフトして扱われます。このとき、下位 9 ビットは 0 として扱われますが、クリアデプス値が 0x7FFF のときのみ下位 9 ビットが 1 として扱われます。

デプスバッファリング方式によるデプス値の違いに関しては、「ジオメトリエンジン」の「デプスバッファリング」を参照してください。

7.3.3.2 クリアイメージによる初期化

3D 表示コントロールレジスタ (DISP3DCNT) の「クリアイメージイネーブルフラグ」をセットすると、カラーバッファ、デプスバッファ、アトリビュートバッファのフォグイネーブルフラグの初期値として VRAM 上のクリアイメージが使用されるようになります。

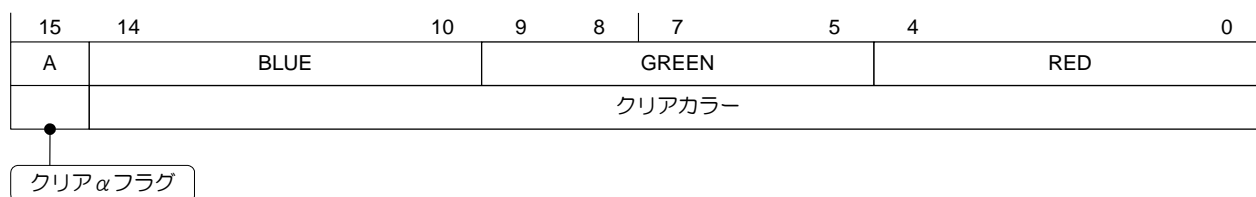
アトリビュートバッファのポリゴン ID については、この機能を使用しても CLEAR_COLOR レジスタの値が使用されません。

クリアイメージを格納した VRAM は、RAM バンクコントロールレジスタによってクリアイメージバッファに割り当ててください。

各クリアイメージのフォーマットは、下記の通りです。

また、クリアイメージの VRAM マッピングを図 7-31 に示します。

クリアカラーイメージフォーマット



● [d15] : クリアαフラグ

実際のクリアα値は次のようになります。

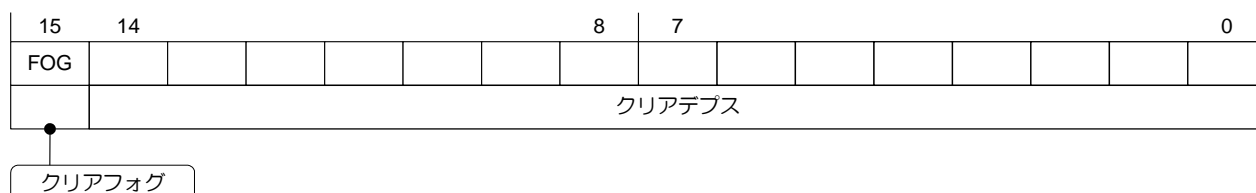
0	0x00
1	0x1F

● [d14~d00] : クリアカラーRGB 値

カラーバッファの RGB 初期値を設定できます。

レンダリングエンジン内のカラーバッファは (R:G:B = 6:6:6) ビットのため、クリアカラー値が 0 のときは下位ビットが 0 として扱われ、クリアカラー値が 0 以外のときは下位 1 ビットが 1 として扱われます。

クリアデプスイメージフォーマット



● FOG[d15] : クリアフォグ

アトリビュートバッファのフォグイネーブルフラグ初期値を設定できます。

2D 面との合成時に、背後のプレーンに対してフォグを適用するかどうかをコントロールできます。

2D の背景を鮮明に表示したい場合に有効です。

● [d14~d00] : クリアデプス

デプスバッファリング方式によるデプス値の違いに関しては、「ジオメトリエンジン」章の「デプスバッファリング」を参照してください。

	ドット 0	1	2	3		253	254	255
ライン 0	0h	2h	4h	6h		1FAh	1FCh	1FEh
1	200h	202h	204h	206h		3FAh	3FCh	3FEh
2	400h	402h					5FCh	5FEh
3	600h	602h					7FCh	7FEh
4	800h							9FEh
251	1F600h							1F7FEh
252	1F800h	1F802h					1F9FCh	1F9FEh
253	1FA00h	1FA02h					1FBFCh	1FBFEh
254	1FC00h	1FC02h	1FC04h	1FC06h		1FDFAh	1FDFCh	1FDFEh
255	1FE00h	1FE02h	1FE04h	1FE06h		1FFFAh	1FFFCCh	1FFFEh

図 7-31 クリアイメージの VRAM マッピング (テクスチャイメージスロット 2、3 共通)

ClearImageOffset : クリアイメージのオフセット設定レジスタ

名称 CLRIMAGE_OFFSET アドレス 0x04000356 属性 W 初期値 0x0000

15							8	7							0
Y オフセット								X オフセット							

レンダリングバッファ初期化時に読み込むクリアイメージに対して、オフセットを与えることができます。

画面の途中で 256×256 のデータ領域を超えると、回り込んで読み込まれます。

クリアイメージのオフセット概念図を図 7-32 に示します。

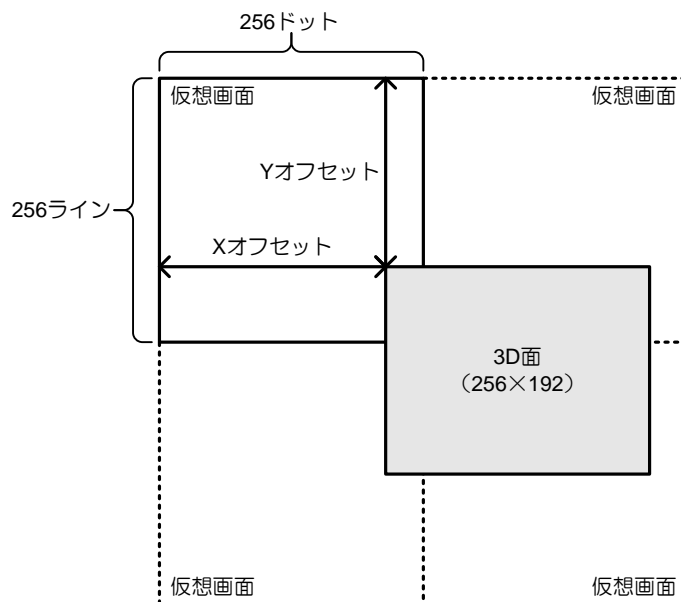


図 7-32 クリアイメージのオフセット

7.3.4 ラスタライズ

ポリゴンの面をピクセル単位に分割し、バッファに書き込むことをラスタライズと呼びます。

この際、レンダリングエンジンはジオメトリエンジンから渡された頂点カラーを元に、ポリゴン内のピクセルカラーを補間します。

レンダリングエンジンは、ピクセルカラーをカラーバッファに格納するとともに、各ピクセルのポリゴン ID とフォグイネーブルフラグをアトリビュートバッファに格納します。

レンダリングエンジンは、ポリゴンリスト RAM から不透明ポリゴンを描画し終えた後、半透明ポリゴンを描画します。

7.3.4.1 不透明ポリゴン

不透明ポリゴンは ($\alpha = 31$) のポリゴンです。

- ポリゴン ID について
ポリゴン描画時に、アトリビュートバッファの不透明ポリゴン ID の領域へ格納されます。
- フォグイネーブルフラグについて
不透明ポリゴン描画時には、フラグメントのフォグイネーブルフラグがアトリビュートバッファのフォグイネーブルフラグに上書きされます。

7.3.4.2 半透明ポリゴン

半透明ポリゴンは ($1 \leq \alpha \leq 30$) または半透明テクスチャを貼ったポリゴンです。

このためシャドウポリゴンも含まれます。

7.3.4.2.1 $1 \leq \alpha \leq 30$ のポリゴン

- ポリゴン ID について
ポリゴン描画時に、アトリビュートバッファの半透明ポリゴン ID の領域へ格納されます。
異なる半透明ポリゴン同士が画面上で重なり合う場合、同じポリゴン ID を使用した半透明ポリゴンは上書きされません。
- フォグイネーブルフラグについて
半透明ポリゴンが描画される際、フラグメントのフォグイネーブルフラグとアトリビュートバッファのフォグイネーブルフラグとの AND 演算結果がアトリビュートバッファに書き込まれます。
この機能により、特定の半透明ポリゴンにだけフォグをかけないようにすることが可能です。

7.3.4.2.2 半透明テクスチャを貼ったポリゴン

半透明テクスチャ（A3I5 および A5I3 テクスチャ）を貼ったポリゴンは、たとえすべてのテクセルが不透明になっていたとしてもポリゴンリスト RAM では半透明ポリゴンの領域に格納されます。

このため、不透明ポリゴンの描画後に描画されることになります。

ポリゴン ID やフォグイネーブルフラグの処理は、ピクセルが不透明（ $\alpha=31$ ）か半透明（ $1 \leq \alpha \leq 30$ ）かで異なるため、不透明なピクセルと半透明なピクセルがポリゴン内に混在する場合には注意してください。

- ポリゴン ID について

- 不透明なピクセルの場合

ポリゴン描画時に、アトリビュートバッファの不透明ポリゴン ID の領域へ格納されます。

このため、エッジマーキングの対象になります。

- 半透明なピクセルの場合

ポリゴン描画時に、アトリビュートバッファの半透明ポリゴン ID の領域へ格納されます。

異なる半透明ポリゴン同士が画面上で重なり合う場合、同じポリゴン ID を使用した半透明ポリゴンは上書きされません。

- フォグイネーブルフラグについて

- 不透明なピクセルの場合

フラグメントのフォグイネーブルフラグがアトリビュートバッファのフォグイネーブルフラグに上書きされます。

- 半透明なピクセルの場合

半透明ポリゴンが描画される際、フラグメントのフォグイネーブルフラグとアトリビュートバッファのフォグイネーブルフラグとの AND 演算結果がアトリビュートバッファに書き込まれます。

この機能により、特定の半透明ポリゴンにだけfogをかけないようにすることが可能です。

7.3.4.3 ワイヤースケルトン

ワイヤースケルトンは（ $\alpha = 0$ ）に設定されたポリゴンです。

この場合、 α は本来の「不透明度」の意味ではなくなり、ポリゴンの輪郭だけ（ワイヤースケルトン）を描画します。

クリッピングされた場合、クリッピング境界（クリッピングによって新しくできた辺）もワイヤースケルトン描画されます。

注意事項

回路の特性上ワイヤースケルトンを半透明で描画することは不可となっております。

- ポリゴン ID について

ワイヤースケルトン描画時に、アトリビュートバッファの不透明ポリゴン ID の領域へ格納されます。（ワイヤーの部分のみポリゴン ID が更新されます）

7.3.4.4 シャドウポリゴン

オブジェクトがライトの照射を遮っている空間を「シャドウボリューム」と定義します。

すると、影はシャドウボリュームが他のオブジェクトと交差した領域に発生していると考えられます。

このシャドウボリュームを実現するためのポリゴンが「シャドウポリゴン」です。

まず、シャドウボリュームの内側描画時にステンシルバッファへのマスクイメージのみ作成し、次にシャドウボリュームの外側描画時にそのマスク領域を除いて描画することによって、視点から見える影のイメージが生成されます。

マスク用と描画用のシャドウポリゴンは、ポリゴン ID により識別されます。

ポリゴン ID	種別
0	マスク用シャドウポリゴン
1～63	描画用シャドウポリゴン

シャドウボリュームの概念図を図 7-33 に、マスク用シャドウポリゴンの描画説明図を図 7-34、描画用シャドウポリゴンの描画説明図を図 7-35 にそれぞれ示します。

シャドウポリゴンで影付けを行う手順は下記の通りです。

1. 描画順序の設定

シャドウポリゴンはマスク用・描画用ともに半透明ポリゴンでなければならず、かつ描画順序に依存する（後述の注意事項参照）ため、SwapBuffers で半透明ポリゴンをマニュアルソートモードにし、ジオメトリエンジンへ転送した順にレンダリングされるように設定してください。



2. マスク用シャドウポリゴンの描画

ポリゴン属性を“裏面のみ描画”、“ID = 0”、“ $\alpha = 1 \sim 30$ ”、“シャドウポリゴン”へ設定し、マスク用シャドウポリゴンを描画してください。レンダリングエンジンはカラーバッファを更新せず、ステンシルバッファに 1 を立てたマスクイメージだけを作成します。



3. 描画用シャドウポリゴンの描画

次に、ポリゴン属性を“両面描画”、“ID = 1～63”、“ $\alpha = 1 \sim 30$ ”、“シャドウポリゴン”へ設定し、描画用シャドウポリゴンを描画してください。レンダリングエンジンはまずステンシルバッファを読み込み、それが 1 の場合には 0 にリセットし、0 の場合にはカラーバッファへ描画しようと試みます。

● ポリゴン ID について

このとき、アトリビュートバッファ上の不透明ポリゴンの ID と半透明ポリゴンの ID のどちらとも異なっている場合のみ描画されます。これは、影を落とすオブジェクトのポリゴン ID と描画用シャドウポリゴンの ID を同じ番号にすることによって、自分自身へ影が落ちないようにするための仕様です。

複数の影が重なる場合にはそれらのポリゴン ID を同じにするかどうかで、重ね書きするかしらないかを制御することができます。

● フォグイネーブルフラグについて

また、シャドウポリゴンが描画される際、フラグメントのフォグイネーブルフラグとアトリビュートバッファのフォグイネーブルフラグとの AND 演算結果がアトリビュートバッファに書き込まれます。

この機能により、影部分にだけフォグをかけないようにすることが可能です。

（応用例：黒いフォグの抜き領域にて表現したスポットライト）。

■シャドウボリューム

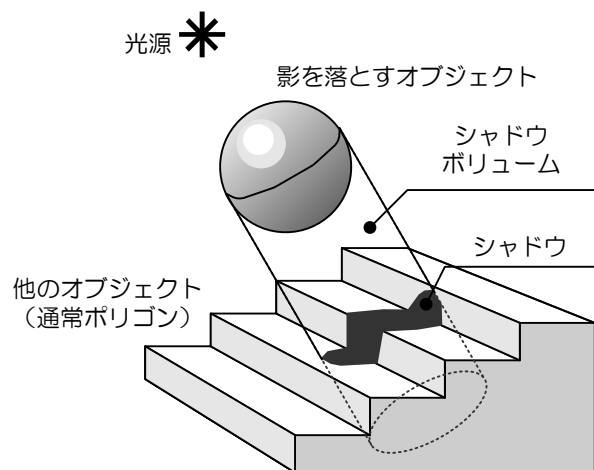


図 7-33 シャドウボリューム概念図

●シャドウボリュームの形状

原則としてシャドウボリュームは閉じた立体の形状とします。ポリゴン数削減等のため開いた形状とする場合は、後述の注意事項を参照してください。

●シャドウボリュームの向き

球体オブジェクトの影をつくるには、光源と球体オブジェクトの一直線上に円筒形のシャドウボリュームをつくります。

●シャドウボリュームの位置

円筒形のシャドウボリュームは、光源から見えない位置（球体の光が当たらない位置）に配置します。

●シャドウボリュームの長さ

影はシャドウボリュームの中に入ったオブジェクトの表面に描画されます。このためシャドウボリュームは影を描画したいオブジェクトの表面を貫通する長さになります。

■ステンシルバッファの演算

1) マスク用シャドウポリゴン描画時

カラーバッファには影響を与えず、デプステストがフェイルした場合にステンシルバッファへ1がセットされます。

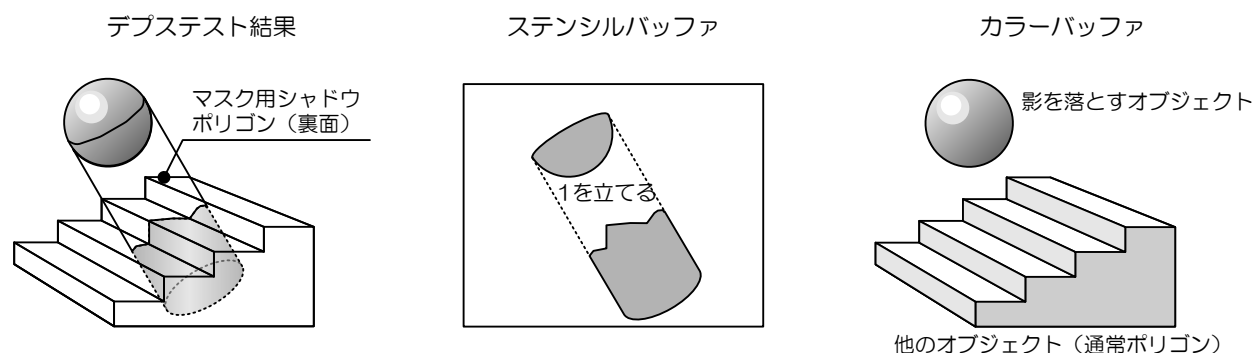


図 7-34 マスク用シャドウポリゴン描画時

2) 描画用シャドウポリゴン描画時（マスク用シャドウポリゴンを描画した後に描画してください）

ステンシルバッファが既に1の場合は0をセットしますが、デプステストをパスした場合に、アトリビュートバッファのポリゴンIDと描画用シャドウポリゴンのポリゴンIDが等しくなければカラーバッファに描画されます。

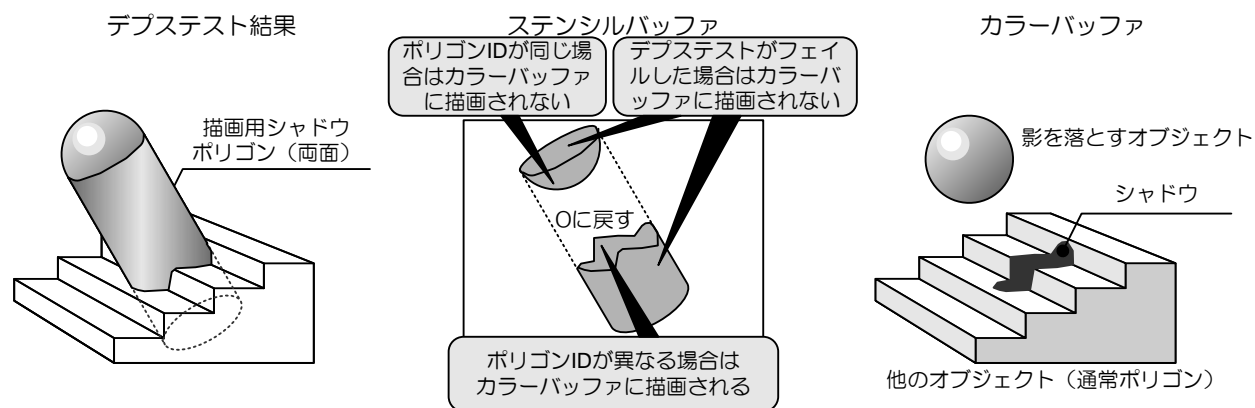


図 7-35 描画用シャドウポリゴン描画時

図 7-35 のように、シャドウは描画用シャドウポリゴンにおける側面の一部分が描画されたものであるため、テクスチャマッピングを行った場合には、シャドウに対して直接貼り付けたような効果にはなりません。

●注意事項

1) シャドウポリゴンの描画順序

1 つの影を生成するために必要なマスク用シャドウポリゴンと描画用シャドウポリゴンは必ず続けて描画してください。

複数のマスク用シャドウポリゴンをまとめて描画した後に描画用シャドウポリゴンも同様に描画した場合、意図した領域へ影が生成されないことがあります。

(正しい描画例)

マスク用シャドウポリューム1 → 描画用シャドウポリューム1 →

マスク用シャドウポリューム2 → 描画用シャドウポリューム2

2) シャドウポリュームの形状

シャドウポリュームを開いた形状にすると、マスク用シャドウポリゴンが存在せず描画用シャドウポリゴンのみ存在する領域が発生し、正しく影が生成されない場合があるため注意してください。

また、シャドウポリュームはクリッピングされることによっても開いた形状になってしまいます。

●テクニック

下記の手順で描画すると、半透明ポリゴンにも影が落ちているシーンを表現できます（ただし、より多くのシャドウポリゴンが必要となります）。

1. 不透明ポリゴン →
2. シャドウポリゴン（不透明ポリゴン上の影） →
3. 影が落とされる半透明ポリゴン（デプスバッファ更新） →
4. シャドウポリゴン（半透明ポリゴン上の影／ポリゴン ID は 2 と共通） →
5. 影が落とされない半透明ポリゴン

図 7-36 に概要図を示します。

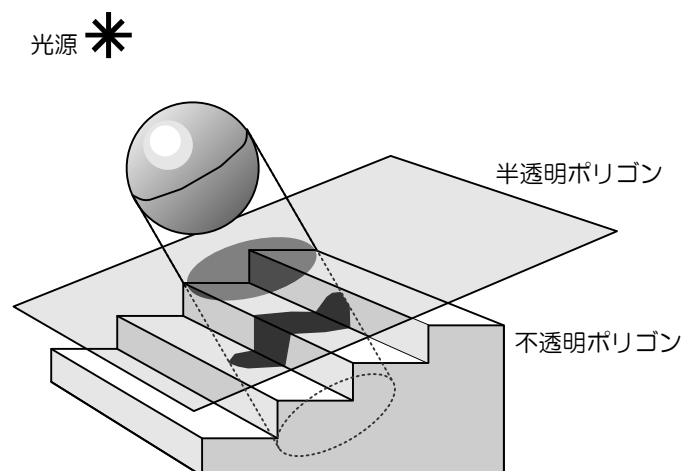


図 7-36 半透明ポリゴンへの影の描画テクニック

7.3.4.5 トゥーン/ハイライト シェーディング

ToonTable : トゥーンテーブルレジスタ

名称 TOON_TABLE_x (x=0~15)

アドレス 0x04000380, 0x04000384, 0x04000388, 0x0400038C, 0x04000390, 0x04000394, 0x04000398, 0x0400039C,
0x040003A0, 0x040003A4, 0x040003A8, 0x040003AC, 0x040003B0, 0x040003B4, 0x040003B8, 0x040003BC

属性 W 初期値 0x00000000

31	30	26	25	24	23	21	20	16	15	14	10	9	8	7	5	4	0

トゥーンシェーディング、ハイライトシェーディングで使⽤します。
トゥーンシェーディングおよびハイライトシェーディングでは、フラグメントカラーの R 値を輝度とみなし、この R 値（上位 5 ビット）をインデックスにしてトゥーンテーブルから RGB 値を参照します。（図 7-37 参照）

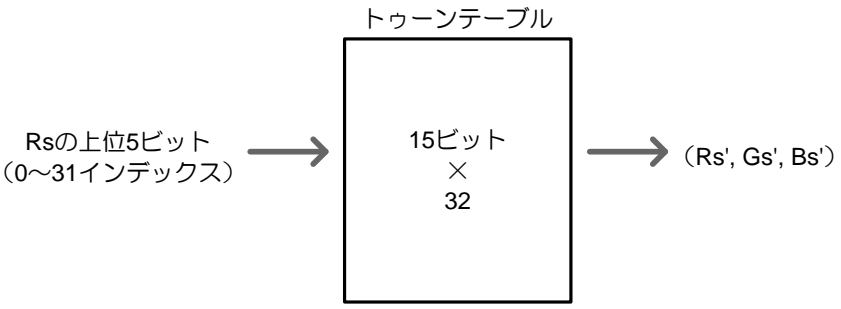


図 7-37 トゥーンテーブルによる変換

7.3.4.5.1 トゥーンシェーディング

フラグメントカラーの R 値を輝度とみなし、この R 値(上位 5 ビット)をインデックスにしてトゥーンテーブルから RGB 値を参照し、新たなフラグメントカラーとします。

テクスチャカラーおよびテーブル参照後のフラグメントカラーのビット数は ($R_5:G_5:B_5 = 5:5:5$) であるため、テクスチャブレンディング演算前に次の式に従って ($R_6:G_6:B_6 = 6:6:6$) へ拡張されます。

- $R_6 = R_5 \ll 1$ (R_5 が 0 の場合)
- $R_6 = (R_5 \ll 1) + 1$ (R_5 が 0 以外の場合)

テクスチャマッピングを行う場合は、フラグメントカラーとして変換後の値を使うこと以外、モジュレーションモードと同じ計算式を用います(表 7-12)。

テクスチャの種類	半透明テクスチャの場合	半透明テクスチャでない場合
テクスチャブレンディング演算式	$R = \{(R_t+1) \times (R_s'+1)-1\} / 64$ $G = \{(G_t+1) \times (G_s'+1)-1\} / 64$ $B = \{(B_t+1) \times (B_s'+1)-1\} / 64$ $A = \{(A_t+1) \times (A_s+1)-1\} / 64$	$R = \{(R_t+1) \times (R_s'+1)-1\} / 64$ $G = \{(G_t+1) \times (G_s'+1)-1\} / 64$ $B = \{(B_t+1) \times (B_s'+1)-1\} / 64$ $A = A_t \times A_s$

(R, G, B, A) : 新しく書き込まれるフラグメントカラー (演算結果の小数点以下は切り捨てられます)

(R_t, G_t, B_t, A_t) : (R:G:B = 6:6:6) ビットへ拡張されたテクスチャカラー

(R_s, G_s, B_s, A_s) : フラグメントカラー

(R_{s'}, G_{s'}, B_{s'}) : テーブル変換後に (R:G:B = 6:6:6) ビットへ拡張されたフラグメントカラー

表 7-12 テクスチャブレンディング演算式 (トゥーンシェーディング)

注意事項

トゥーンテーブルは、すべてのトゥーンシェーディングポリゴン共通のフラグメントカラーとして参照されます。
 トゥーンシェーディングポリゴン毎に別々のカラーにしたい場合は、単色ポリゴンであってもテクスチャを使用して着色してください。

環境反射色や放射光色を使用してフラグメントカラー R の最小値が底上げされると、R の有効範囲が狭くなるためにトゥーンシェーディングの階調が粗くなってしまいます。

トゥーンシェーディング時のマテリアルカラーは本来の意味を失っているため、拡散反射色のみ使用して階調を確保することを推奨します。

7.3.4.5.2 ハイライトシェーディング

フラグメントカラーの R 値を輝度とみなし、この R 値(上位 5 ビット)をインデックスにしてトゥーンテーブルから RGB 値を参照した値(カラーオフセット値)をフラグメントカラーに対して加算します。

テクスチャカラーおよびテーブル参照後のフラグメントカラーのビット数は ($R_5:G_5:B_5 = 5:5:5$) であるため、テクスチャブレンディング演算前に次の式に従って ($R_6:G_6:B_6 = 6:6:6$) へ拡張されます。

- $R_6 = R_5 \ll 1$ (R_5 が 0 の場合)
- $R_6 = (R_5 \ll 1) + 1$ (R_5 が 0 以外の場合)

テクスチャマッピングを行う場合は、テクスチャカラーの RGB それぞれにフラグメントカラーの R でモジュレーションを行ったカラーに対してカラーオフセット値を加算します。(表 7-13 参照)

テクスチャにハイライトが当たったような効果(テクスチャ自身の色よりも明るい色を出す)ことができます。

テクスチャの種類	半透明テクスチャの場合	半透明テクスチャでない場合
テクスチャブレンディング演算式	$R = \min[63, \{(R_t+1) \times (R_s+1) - 1\} / 64 + R_{s'}]$ $G = \min[63, \{(G_t+1) \times (R_s+1) - 1\} / 64 + G_{s'}]$ $B = \min[63, \{(B_t+1) \times (R_s+1) - 1\} / 64 + B_{s'}]$ $A = \{(A_t+1) \times (A_s+1) - 1\} / 32$	$R = \min[63, \{(R_t+1) \times (R_s+1) - 1\} / 64 + R_{s'}]$ $G = \min[63, \{(G_t+1) \times (R_s+1) - 1\} / 64 + G_{s'}]$ $B = \min[63, \{(B_t+1) \times (R_s+1) - 1\} / 64 + B_{s'}]$ $A = A_t \times A_s$

(R, G, B, A) : 新しく書き込まれるフラグメントカラー(演算結果の小数点以下は切り捨てられます)

(R_t, G_t, B_t, A_t) : (R:G:B = 6:6:6) ビットへ拡張されたテクスチャカラー

(R_s, G_s, B_s, A_s) : フラグメントカラー

(R_{s'}, G_{s'}, B_{s'}) : テーブル変換後に (R:G:B = 6:6:6) ビットへ拡張されたフラグメントカラー

表 7-13 テクスチャブレンディング演算式(ハイライトシェーディング)

注意事項

カラーオフセット値を加算するため、フラグメントカラーの色相が変化する場合があります。

7.3.5 テクスチャ

7.3.5.1 テクスチャブレンディング

レンダリングエンジンは、ジオメトリエンジンから渡されたテクスチャの設定（フリップ／リピート）や頂点のテクスチャ座標および w 値から、ポリゴン内の各ピクセルに対応するテクスチャ座標を補間します。

ポリゴンの各ピクセルカラーに対してテクセルカラーをブレンディングすることをテクスチャブレンディングと呼びます。

PolygonAttr レジスタにより、テクスチャブレンディングのモードをモジュレーションかデカルに設定できます。

7.3.5.1.1 テクスチャイメージのサンプリング

TWL では、各頂点のテクスチャ座標から、ポリゴン内にある各ピクセルの左上位置に対応するテクスチャ座標を補間し、テクスチャイメージをサンプリングします。（図 7-38 参照）

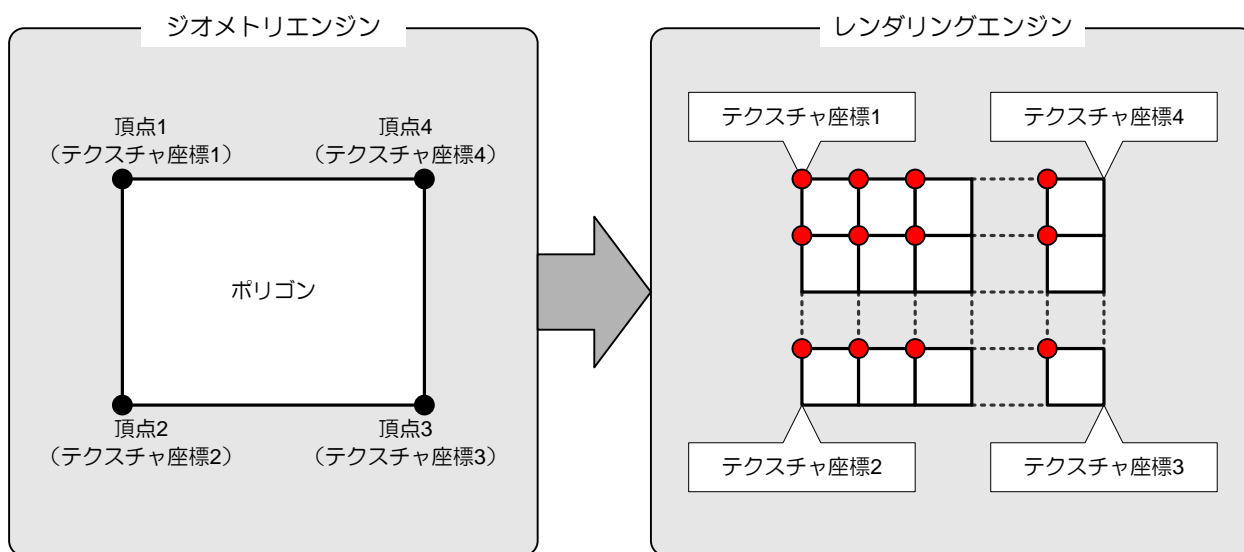


図 7-38 テクスチャイメージのサンプリング

例えば、8×8のテクスチャを貼ったポリゴンの表示が横 14 ドットであった場合、ピクセルとテクセルの対応は図 7-39 の左図のようになります。

このままポリゴンを裏返した場合、テクセルとピクセルの対応がずれることになります。

図 7-39 の右図では、左端にテクセル 0 が現れ（テクスチャの H フリップなしの場合は抜きになります）、右端にテクセル 0 が 1 回しかサンプリングされていません。

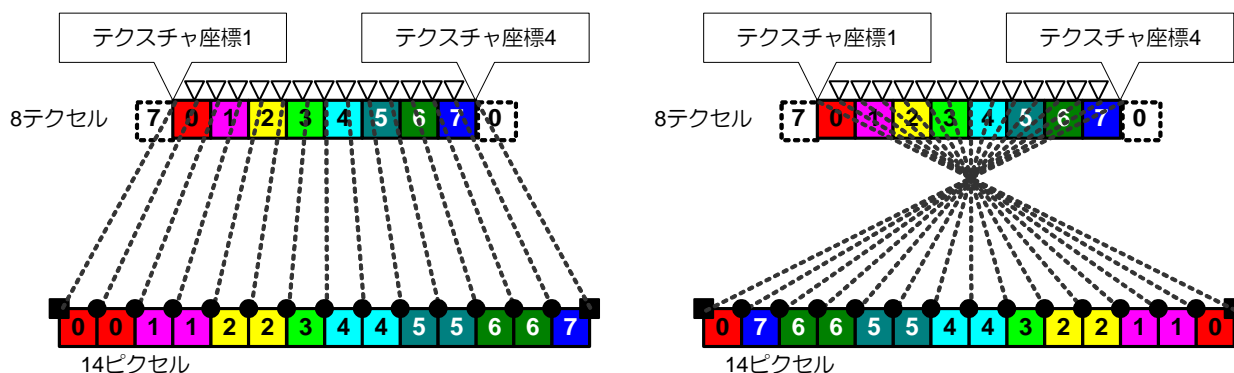


図 7-39 8×8 テクセルのテクスチャを横 14 ドットのポリゴンに貼り付けた場合

同様に、8×8 のテクスチャを貼ったポリゴンの表示が横 8 ドットであった場合は図 7-40 のようになります。

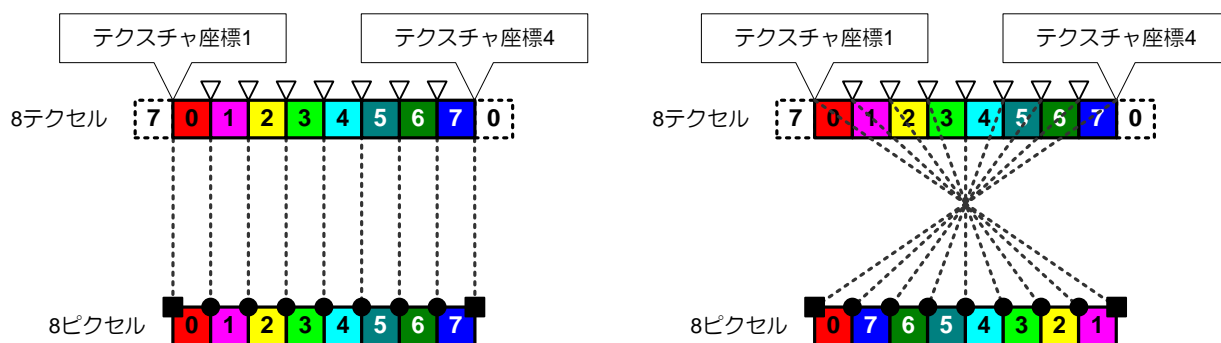


図 7-40 8×8 テクセルのテクスチャを横 8 ドットのポリゴンに貼り付けた場合

ポリゴンを利用して OBJ や BG のような 2D 表現を行う場合は、テクセルとピクセルの対応を 1 対 1 にすることになりますが、このような場合では 8×8 テクセルのテクスチャを貼ったポリゴンが LCD 上の 8×8 ピクセルにレンダリングされる際、表（おもて）面および裏面の表示は図 7-41 のようになります。

表（おもて）面								ポリゴンを横方向に裏返し								ポリゴンを縦方向に裏返し							
00	01	02	03	04	05	06	07	00	07	06	05	04	03	02	01	00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17	10	17	16	15	14	13	12	11	70	71	72	73	74	75	76	77
20	21	22	23	24	25	26	27	20	27	26	25	24	23	22	21	60	61	62	63	64	65	66	67
30	31	32	33	34	35	36	37	30	37	36	35	34	33	32	31	50	51	52	53	54	55	56	57
40	41	42	43	44	45	46	47	40	47	46	45	44	43	42	41	40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57	50	57	56	55	54	53	52	51	30	31	32	33	34	35	36	37
60	61	62	63	64	65	66	67	60	67	66	65	64	63	62	61	20	21	22	23	24	25	26	27
70	71	72	73	74	75	76	77	70	77	76	75	74	73	72	71	10	11	12	13	14	15	16	17

右90°（左270°）回転								180° 回転								右270°（左90°）回転							
00	70	60	50	40	30	20	10	00	07	06	05	04	03	02	01	00	10	20	30	40	50	60	70
01	71	61	51	41	31	21	11	70	77	76	75	74	73	72	71	07	17	27	37	47	57	67	77
02	72	62	52	42	32	22	12	60	67	66	65	64	63	62	61	06	16	26	36	46	56	66	76
03	73	63	53	43	33	23	13	50	57	56	55	54	53	52	51	05	15	25	35	45	55	65	75
04	74	64	54	44	34	24	14	40	47	46	45	44	43	42	41	04	14	24	34	44	54	64	74
05	75	65	55	45	35	25	15	30	37	36	35	34	33	32	31	03	13	23	33	43	53	63	73
06	76	66	56	46	36	26	16	20	27	26	25	24	23	22	21	02	12	22	32	42	52	62	72
07	77	67	57	47	37	27	17	10	17	16	15	14	13	12	11	01	11	21	31	41	51	61	71

図 7-41 LCD 上での表面および裏面の表示

7.3.5.1.2 デカルモード

テクスチャの α 値により、ライティング（照光処理）で生成された頂点カラーをグーローシェーディングした結果（フラグメントカラー）かテクスチャのカラー値のどちらか一方を表示します。

半透明テクスチャの場合は、テクスチャの α 値でブレンディングされます。

表 7-14 にデカルモードにおけるテクスチャブレンディング演算式を示します。

テクスチャカラーのビット数は（ $R_5:G_5:B_5 = 5:5:5$ ）であるため、テクスチャブレンディング演算前に次の式に従って（ $R_6:G_6:B_6 = 6:6:6$ ）へ拡張されます。

- $R_6 = R_5 << 1$ (R_5 が 0 の場合)
- $R_6 = (R_5 << 1) + 1$ (R_5 が 0 以外の場合)

テクスチャの種類	半透明テクスチャの場合	半透明テクスチャでない場合
テクスチャブレンディング演算式	$R = \{At \times Rt + (31 - At)Rs\} / 32$ $G = \{At \times Gt + (31 - At)Gs\} / 32$ $B = \{At \times Bt + (31 - At)Bs\} / 32$ $A = As$ <p>ただし、例外処理として $At = 0$ のときは $(R, G, B, A) = (Rs, Gs, Bs, As)$ $At = 31$ のときは $(R, G, B, A) = (Rt, Gt, Bt, As)$ が適用されます。</p>	$R = At \times Rt + (1 - At)Rs$ $G = At \times Gt + (1 - At)Gs$ $B = At \times Bt + (1 - At)Bs$ $A = As$

(R, G, B, A) : 新しく書き込まれるフラグメントカラー（演算結果の小数点以下は切り捨てられます）

(Rt, Gt, Bt, At) : ($R:G:B = 6:6:6$) ビットへ拡張されたテクスチャカラー

(Rs, Gs, Bs, As) : フラグメントカラー

表 7-14 テクスチャブレンディング演算式（デカルモード）

7.3.5.1.3 モジュレーションモード

ライティング（照光処理）で生成された頂点カラーをグーローシェーディングした結果（フラグメントカラー）をテクスチャのカラー値で変調して表示します。

表 7-15 にモジュレーションモードにおけるテクスチャブレンド演算式を示します。

テクスチャカラーのフラグメントカラーのビット数は（R₅:G₅:B₅ = 5:5:5）であるため、テクスチャブレンド演算前に次の式に従って（R₆:G₆:B₆ = 6:6:6）へ拡張されます。

- R₆ = R₅ << 1 （R₅ が 0 の場合）
- R₆ = （R₅ << 1）+ 1 （R₅ が 0 以外の場合）

テクスチャの種類	半透明テクスチャの場合	半透明テクスチャでない場合
テクスチャブレンド演算式	R = {(Rt+1)×(Rs+1)-1}／64 G = {(Gt+1)×(Gs+1)-1}／64 B = {(Bt+1)×(Bs+1)-1}／64 A = {(At+1)×(As+1)-1}／32	R = {(Rt+1)×(Rs+1)-1}／64 G = {(Gt+1)×(Gs+1)-1}／64 B = {(Bt+1)×(Bs+1)-1}／64 A = At×As

（R, G, B, A）：新しく書き込まれるフラグメントカラー（演算結果の小数点以下は切り捨てられます）

（Rt, Gt, Bt, At）：（R:G:B = 6:6:6）ビットへ拡張されたテクスチャカラー

（Rs, Gs, Bs, As）：フラグメントカラー

表 7-15 テクスチャブレンド演算式（モジュレーションモード）

7.3.5.2 テクスチャフォーマット

TWL では 7 種類のテクスチャフォーマットを扱うことができます。
テクスチャフォーマットの一覧を表 7-16 に示します。

フォーマット	1 テクセルにつき 何色から選択可能か	パレット ベース境界※	α 値 ビット数	1 テクセル当たりの ビット数
4 色パレットテクスチャ	4	0x08	0	2
16 色パレットテクスチャ	16	0x10	0	4
256 色パレットテクスチャ	256	0x10	0	8
4×4 テクセル圧縮テクスチャ	4 (4×4 テクセル毎)	0x10	0	3 (パレットインデックス データ含む)
A3I5 半透明テクスチャ	32	0x10	3	8
A5I3 半透明テクスチャ	8	0x10	5	8
ダイレクトカラーテクスチャ	32,768	パレット 不使用	1	16

表 7-16 テクスチャフォーマット一覧

※「パレットベース境界」は、TexPlttBase コマンドでパレットベースを 1 増やしたときのアドレス増加分です。

7.3.5.2.1 テクスチャイメージ

レンダリングエンジンは TexImageParam コマンドによって指定したフォーマットでテクスチャイメージスロットのテクスチャイメージを参照します。
テクスチャイメージはテクセルデータによって構成されます。

7.3.5.2.1.1 4 色パレットテクスチャ

テクセルデータフォーマット

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T7		T6		T5		T4		T3		T2		T1		T0	
8 テクセル分のデータ (2bit / texel)															

● T7～T0：テクセルデータ

テクスチャカラーパレットのカラーNo. (0～3) を指定します。

表示テクスチャ

T0	T1	T2	T3	T4	T5	T6	T7

7.3.5.2.1.2 16 色パレットテクスチャ

テクセルデータフォーマット

15	12	11	8	7	4	3	0
T3		T2		T1		T0	
4 テクセル分のデータ (4bit / texel)							

- T3～T0：テクセルデータ
テクスチャカラーパレットのカラーNo. (0～15) を指定します。

表示テクスチャ

T0	T1	T2	T3				

7.3.5.2.1.3 256 色パレットテクスチャ

テクセルデータフォーマット

15	8	7	0
T1		T0	
2 テクセル分のデータ (8bit / texel)			

- T1～T0：テクセルデータ
テクスチャカラーパレットのカラーNo. (0～255) を指定します。

表示テクスチャ

T0	T1						

7.3.5.2.1.4 4×4 テクセル圧縮テクスチャ

画像を 4×4 画素のブロックに分割して、2 ビットのインデックス化されたパレット付き画像に変換することで圧縮効果を得るフォーマットです。

テクセルデータフォーマット

31				24	23	16				15	8				7	0					
T33	T32	T31	T30	T23	T22	T21	T20	T13	T12	T11	T10	T03	T02	T01	T00						
4×4 テクセル分のデータ (2bit / texel)																					

- T33～T00：テクセルデータ
カラーNo. (0～3) を指定します。

表示テクスチャ

T00	T01	T02	T03				
T10	T11	T12	T13				
T20	T21	T22	T23				
T30	T31	T32	T33				

パレットインデックスデータフォーマット

15	14	13					8	7							0
A	PTY														
パレット設定		パレットアドレス													

●パレット設定

●A：3色／4色設定フラグ

0	3色+透明色モード (カラー3を透明色とする)
1	4色モード

●PTY：パレットタイプ選択フラグ

0	4色パレット (4×4 テクセル毎に 4 パレット)
1	線形補間 4 色パレット (4×4 テクセル毎に 2 パレット)

●パレットアドレス

カラーデータが格納されているテクスチャパレットスロットのアドレスを、2色単位（4バイト単位）で指定します。つまりテクスチャパレットスロット上において、（TexPlttBase コマンドで設定した値×0x10）+（パレットアドレス設定値×4）番地にあるカラーをカラー0とみなします。

テクセルのカラー値は RGB 各々について表 7-17 のように計算されます。

PTY=0		
PTY=0	A=0	カラー0[5:0]：(パレット 0[4:0]==0) ? (パレット 0[4:0] × 2) : (パレット 0[4:0] × 2 + 1) カラー1[5:0]：(パレット 1[4:0]==0) ? (パレット 1[4:0] × 2) : (パレット 1[4:0] × 2 + 1) カラー2[5:0]：(パレット 2[4:0]==0) ? (パレット 2[4:0] × 2) : (パレット 2[4:0] × 2 + 1) カラー3[5:0]：透明色
	A=1	カラー0[5:0]：(パレット 0[4:0]==0) ? (パレット 0[4:0] × 2) : (パレット 0[4:0] × 2 + 1) カラー1[5:0]：(パレット 1[4:0]==0) ? (パレット 1[4:0] × 2) : (パレット 1[4:0] × 2 + 1) カラー2[5:0]：(パレット 2[4:0]==0) ? (パレット 2[4:0] × 2) : (パレット 2[4:0] × 2 + 1) カラー3[5:0]：(パレット 3[4:0]==0) ? (パレット 3[4:0] × 2) : (パレット 3[4:0] × 2 + 1)
PTY=1	A=0	カラー0[5:0]：パレット 0[4:0] × 2 カラー1[5:0]：パレット 1[4:0] × 2 カラー2[5:0]：パレット 0[4:0] + パレット 1[4:0] カラー3[5:0]：透明色
	A=1	カラー0[5:0]：パレット 0[4:0] × 2 カラー1[5:0]：パレット 1[4:0] × 2 カラー2[5:0]：(パレット 0[4:0] × 5 + パレット 1[4:0] × 3) / 4 カラー3[5:0]：(パレット 0[4:0] × 3 + パレット 1[4:0] × 5) / 4

表 7-17 テクセルのカラー値

4×4 テクセルにつき、1つのパレットインデックスデータが対応付けられます（図 7-42 参照）。

テクスチャイメージスロット

4×4 テクセル圧縮モードにおいては、テクスチャイメージデータおよびテクスチャパレットインデックスデータを以下のようにマッピングしてください。

- テクセルデータ

テクスチャイメージスロット 0 および 2 に置いてください。

- テクスチャパレットインデックスデータ

テクスチャイメージスロット 1 に置いてください。

- テクセルデータとテクスチャパレットインデックスデータの対応

テクスチャイメージスロット 0 の TIAa 番地に置かれた 4×4 テクセルデータに対応するテクスチャパレットインデックスデータは、テクスチャイメージスロット 1 の (TIAa/2) 番地に置いてください。

テクスチャイメージスロット 2 の TIAb 番地に置かれた 4×4 テクセルデータに対応するテクスチャパレットインデックスデータは、テクスチャイメージスロット 1 の (0x10000 + (TIAb/2)) 番地に置いてください。(図 7-42 参照)

4×4 テクセルにつき、1 つのテクスチャパレットインデックスデータが対応付けられることになります。

個々のテクセル色は、テクスチャパレットインデックスデータで指定したアドレス(テクスチャカラーパレットスロット)のカラーを 0 番として、テクセルデータで指定したカラーNo.の色が採用されます。

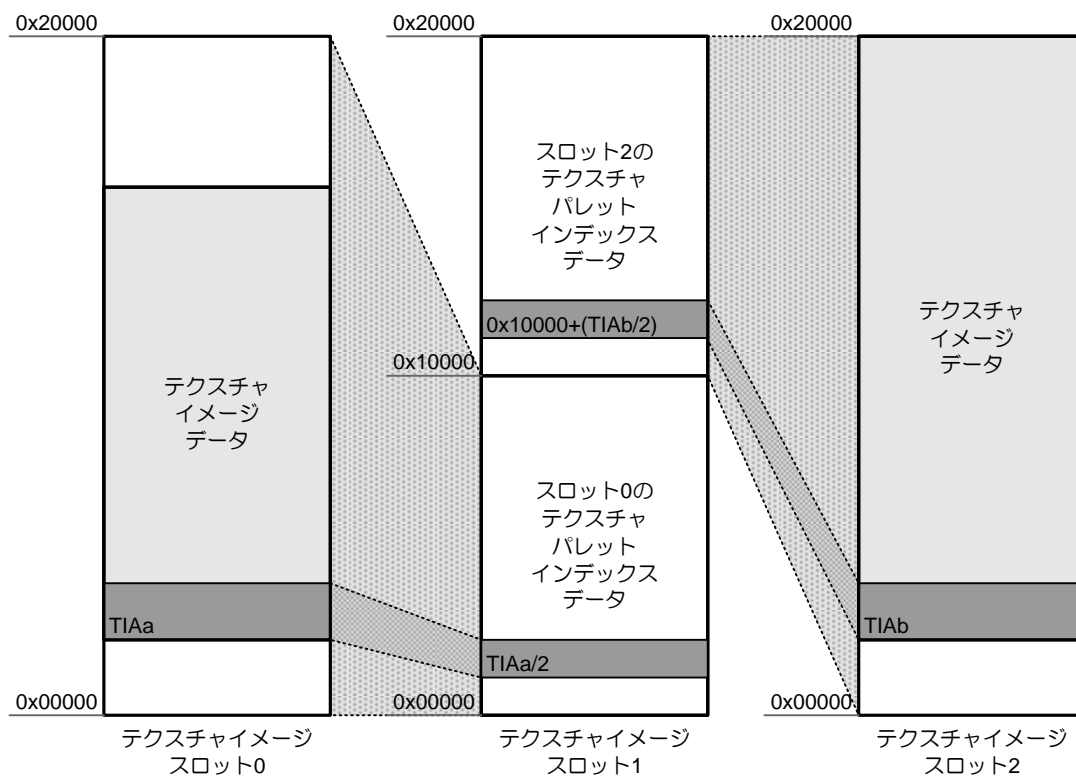


図 7-42 テクスチャイメージスロット

注意事項

テクスチャデータはスロット 0 およびスロット 2 に置かれており、アドレスが飛んでしまうため 1024×1024 は参照不可能となっています。4x4 テクセル圧縮テクスチャモードでは 1024×512 以下のテクスチャサイズで使用してください。

7.3.5.2.1.5 A3I5 半透明テクスチャ

テクセルデータフォーマット

15	13	12	8	7	5	4	0
ALPHA	INDEX	ALPHA	INDEX				
T1	T0						

● T1～T0：テクセルデータ

● ALPHA： α 値

テクセルの透明度を指定します。0 が透明となります。

α 値は、レンダリングエンジンにおいて下表の通り 5 ビットに拡張された上で使用されます。

(5 ビット $\alpha = \{(3 \text{ ビット } \alpha \ll 2) + (3 \text{ ビット } \alpha \gg 1)\}$)

3 ビット α	0	1	2	3	4	5	6	7
5 ビット α	0	4	9	13	18	22	27	31

● INDEX：カラーNo.

テクスチャカラーパレットのカラーNo. (0～31) を指定します。

表示テクスチャ

T0	T1						

7.3.5.2.1.6 A5I3 半透明テクスチャ

テクセルデータフォーマット

15	11	10	8	7	3	2	0
ALPHA	INDEX	ALPHA	INDEX				
T1	T0						

● T1～T0：テクセルデータ

● ALPHA： α 値

テクセルの透明度を指定します。0 が透明となります。

● INDEX：カラーNo.

テクスチャカラーパレットのカラーNo. (0～7) を指定します。

表示テクスチャ

T0	T1						

7.3.5.2.1.7 ダイレクトカラーテクスチャ

テクセルデータフォーマット

15	14	10	9	8	7	5	4	0
ALPHA	BLUE			GREEN			RED	
T0								

- T0：テクセルデータ
テクセルのカラーを直接設定します。テクスチャカラーパレットは使用しません。

表示テクスチャ

T0							

7.3.5.2.2 テクスチャパレット

テクスチャパレットスロットにはテクスチャカラーデータを格納します。

テクスチャカラーデータフォーマット

15	14	10	9	8	7	5	4	0
		TEX_COLOR_BLUE		TEX_COLOR_GREEN			TEX_COLOR_RED	
		カラー						

テクスチャカラーパレットは、テクスチャのフォーマット（4色／16色／256色／4×4圧縮テクセル／A3I5／A5I3）によって参照のされ方が決まります。

基本的には、テクスチャパレットベースで指定したパレットからテクセルデータで特定したカラーNo.のカラーが参照されます。

ただし4×4テクセル圧縮テクスチャの場合は、テクスチャパレットベースに加えパレットアドレスを設定することでパレットを指定します。

各テクスチャカラーパレットのパレットベースとパレットアドレスマップについては、図 7-43 から図 7-46 を参照してください。

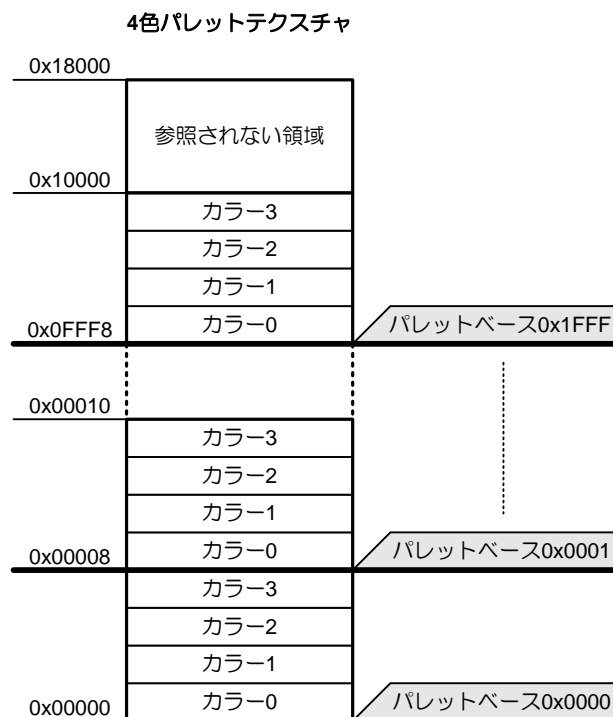


図 7-43 パレットベースとパレットアドレス（4色パレット）

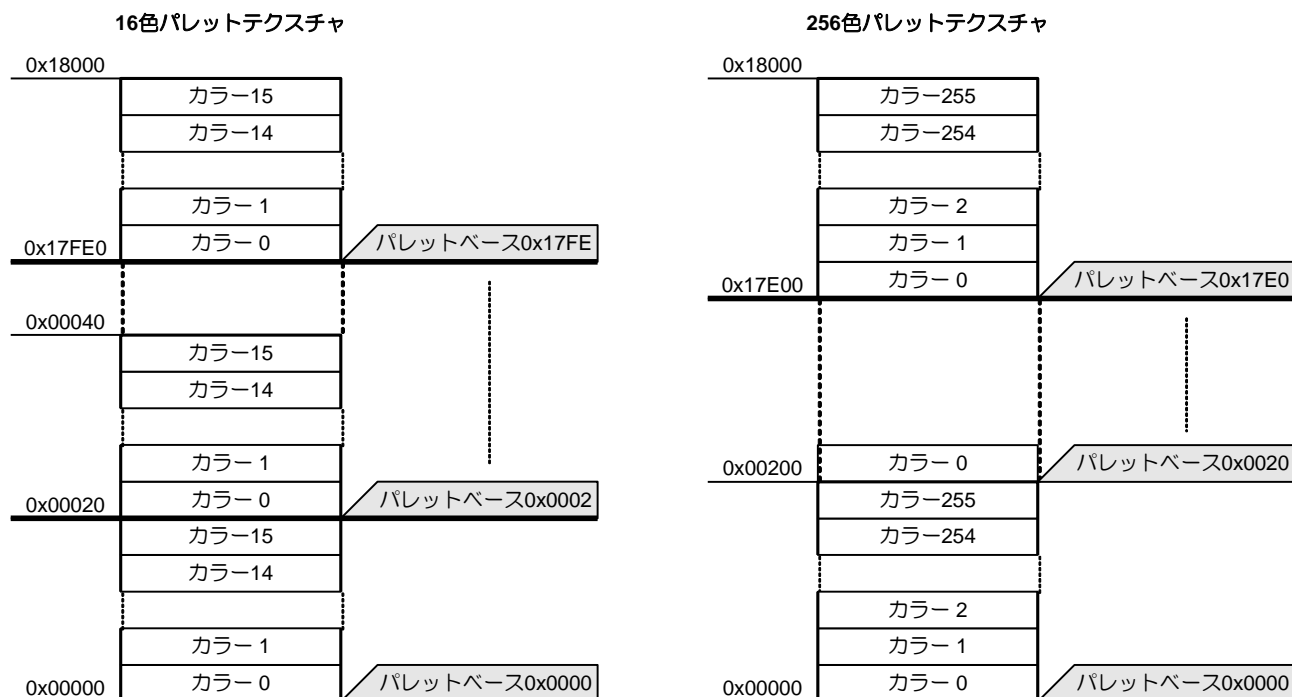


図 7-44 パレットベースとパレットアドレス (16 色パレット、256 色パレット)

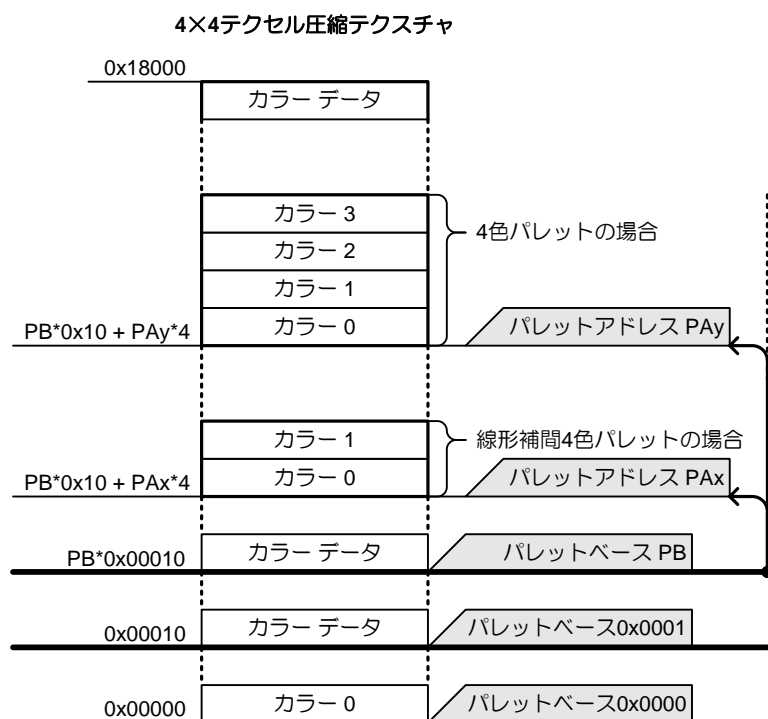


図 7-45 パレットベースとパレットアドレス (4×4 テクセル圧縮)

4×4 テクセル圧縮テクスチャの場合は、パレットのアドレスをパレットベースとパレットアドレスで設定します。また、4 色パレットの場合はカラー0～カラー3 まで参照されますが、線形補間 4 色パレットの場合はカラー0 と 1 しか参照されません。

残りの 2 色はカラー0 と 1 から計算で求められ、テクセルに適用されます。

また、4 色パレットの場合においても「3 色+透明色モード」のときは、カラー3 は参照されず透明色となります。

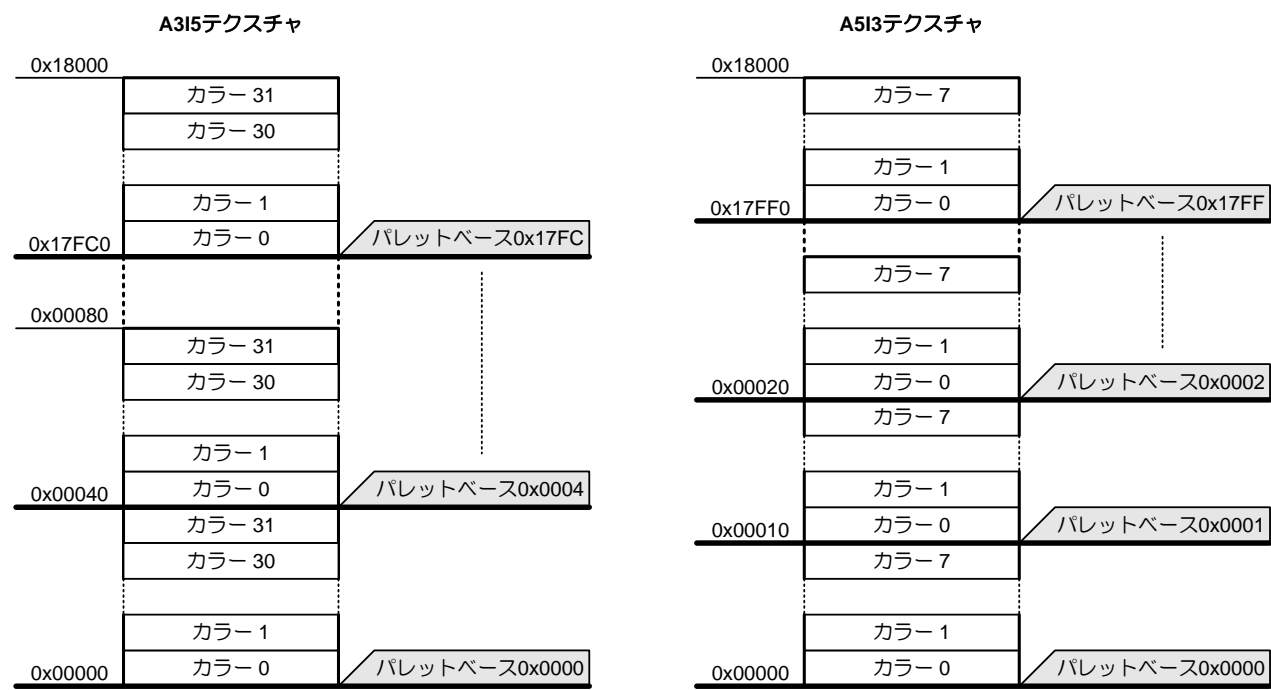


図 7-46 パレットベースとパレットアドレス (A3I5、A5I3)

7.3.6 α テスト

テクスチャレンディング後のフラグメントの α 値を下記レジスタ設定値と比較し、設定値以下の場合は描画しません。

AlphaReference： α テスト比較値レジスタ

名称 ALPHA_TEST_REF アドレス 0x04000340 属性 W 初期値 0x0000

15	8	7	4	0
				ALPHA_REFERENCE
				α テスト比較値

● ALPHA_REFERENCE[d04～d00]： α テスト比較値

α テスト ON のとき、 α 値がこの設定値以下のピクセルは描画されません

● ワイヤフレーム表示のポリゴンについて

ポリゴン属性において $\alpha=0$ にした時点でポリゴンはワイヤフレーム表示となり、 α の本来の意味ではなくなります。半透明テクスチャを貼らない場合、ワイヤフレーム部分の実際の α 値は 31 となります。

このため α テストを ON にして α テスト比較値を設定した場合、ワイヤフレームを表示すると、 α テスト比較値を 31 にしたとき以外は表示されてしまいます。

7.3.7 α ブレンディング

レンダリングエンジンの α ブレンディングは、3D 面生成後に 2D 面との α ブレンディングを自然に行えるような仕様になっています。2D 面との α ブレンディングは、2D グラフィックス機能のカラー特殊効果機能によって行われます。

また、レンダリングエンジンの α ブレンディングは DISP3DCNT レジスタの α ブレンディングイネーブルフラグによって ON/OFF を制御できます。

表 7-18 に α ブレンディング時の演算式を示します。

α ブレンディングイネーブルフラグ	カラーバッファに新しく格納されるデータの演算式
ON のとき	$R = \{(As+1) \times Rs + (31-As)Rb\} / 32$ $G = \{(As+1) \times Gs + (31-As)Gb\} / 32$ $B = \{(As+1) \times Bs + (31-As)Bb\} / 32$ $A = \max[As, Ab]$ <p>ただし例外処理として、 $As = 0$ のときは $(R, G, B, A) = (Rb, Gb, Bb, \max[As, Ab])$ $As = 31$ または $Ab = 0$ のときは $(R, G, B, A) = (Rs, Gs, Bs, \max[As, Ab])$ が適用されます。</p>
OFF のとき	$As = 0$ のとき $(R, G, B, A) = (Rb, Gb, Bb, Ab)$ As が 0 以外のとき $(R, G, B, A) = (Rs, Gs, Bs, As)$

(R, G, B, A)：カラーバッファに新しく格納されるカラー（演算結果の小数点以下は切り捨てられます）

(Rb, Gb, Bb, Ab)：カラーバッファのカラー

(Rs, Gs, Bs, As)：フラグメントカラー

表 7-18 α ブレンディング時の演算式

7.3.7.1 3D の α ブレンディング

カラーバッファのカラー値とフラグメントのカラー値を、フラグメントの α 値を元にしてブレンディングし、その結果をカラーバッファに書き戻す処理を行います。

7.3.7.2 2D と 3D の α ブレンディングの前処理

上記 3D の α ブレンディングでは、カラーバッファの α 値はフラグメントの α 値の方が大きい場合のみ更新されます。これは 2D と 3D の α ブレンディングをする場合に、半透明ポリゴン同士が重なった領域の α 値を近似するための仕様です。

また、カラーバッファの α 値が0のときに半透明ポリゴンを描画した場合は、 α ブレンディングは行われずフラグメントカラーがそのままカラーバッファへ書き込まれます。

つまり、ClearColor でカラーバッファの α 値を0でクリアしておくことによって、ClearColor のRGB値がブレンドされることがなくなり、2D面との合成がより自然になります。

2D と 3D の α ブレンディングについては、「2D と 3D の α ブレンディング」を参照してください。

7.3.8 エッジマーキング

エッジマーキングは、アトリビュートバッファ上でポリゴン ID が異なっている不透明ポリゴンのエッジを縁どりする機能です。

DISP3DCNT レジスタのエッジマーキングイネーブルフラグによって ON/OFF を制御できます。

エッジマーキングに使用されるカラーは、ポリゴン ID の上位 3 ビットをインデックスとして 8 通りのカラーが選択されます。

EdgeColor : エッジカラーレジスタ

名称 EDGE COLOR x (x=0~3)

アドレス 0x04000330, 0x04000334, 0x04000338, 0x0400033C 属性 W 初期値 0x00000000

ポリゴン ID (n1)=2x+1 のときのエッジマージング色																		ポリゴン ID (n0)=2x のときのエッジマージング色																	
BLUE n1										GREEN n1					RED n1			BLUE n0										GREEN n0					RED n0		
31	30	26			25	24	23	21	20	16	15	14	10			9	8	7	5	4	0														

- [d30～d16], [d14～d00]：エッジマーキング色
エッジマーキングに適用される8通りのカラーを設定します。

ポリゴンがクリッピングされた場合、クリッピング境界（クリッピングによって新しくできた辺）にもエッジマーキングが適用されます。

また、画面の端ではクリアポリゴンIDと比較する仕様になっています。

このため、クリッピングで新しくできた辺に関してクリアポリゴン ID とポリゴン ID が同じであれば、エッジマーキングが適用されません。

注意事項

PolygonAttr コマンドで「半透明ポリゴンのデプス値更新イネーブルフラグ」に 1 を設定した場合、半透明ポリゴンの背後に存在する不透明ポリゴンのエッジマーキングが正しく描画されないことがありますので注意してください。
(エッジマーキングの判定にデプスバッファの値を参照しているため)

FogTable : フォグ濃度テーブルレジスタ

名称 FOG_TABLE_x (x=0~7)

アドレス 0x04000360, 0x04000364, 0x04000368, 0x0400036C, 0x04000370, 0x04000374, 0x04000378, 0x0400037C

属性 W 初期値 0x00000000

31		30		24		23		22		16		15		14		8		7		6		0	
		DENSITY n3				DENSITY n2				DENSITY n1				DENSITY n0									
		フォグ濃度 n3=4x+3				フォグ濃度 n2=4x+2				フォグ濃度 n1=4x+1				フォグ濃度 n0=4x									

32 段階のフォグ濃度テーブルを設定します。

各ピクセルにおけるフォグ濃度は、対応するデプスパッファの値によって線形補間された値が使用されるため、任意のフォグ濃度曲線を近似することができます（図 7-47 参照）。

7.3.9.1.1 フォグ濃度計算式

F_IVL = (0x400 >> FOG_SHIFT) とし、フォグテーブルの i 番目のパラメータを f(i) とすれば、フォグ濃度はデプス値の上位 15 ビット (Zd) に応じて下記の通りになります。

1) 0x0000 ≤ Zd ≤ (FOG_OFFSET + F_IVL - 1) のとき

フォグ濃度 f = f(0)

2) (FOG_OFFSET + F_IVL × i) ≤ Zd ≤ (FOG_OFFSET + F_IVL × (i+1) - 1) (i = 1~31) のとき

$$\text{フォグ濃度 } f = \frac{\{f(i) - f(i-1)\} \times \{Zd - (FOG_OFFSET + F_IVL \times i)\}}{F_IVL} + f(i-1)$$

3) FOG_OFFSET + F_IVL × 32 ≤ Zd ≤ 0x7FFF のとき

フォグ濃度 f = f(31)

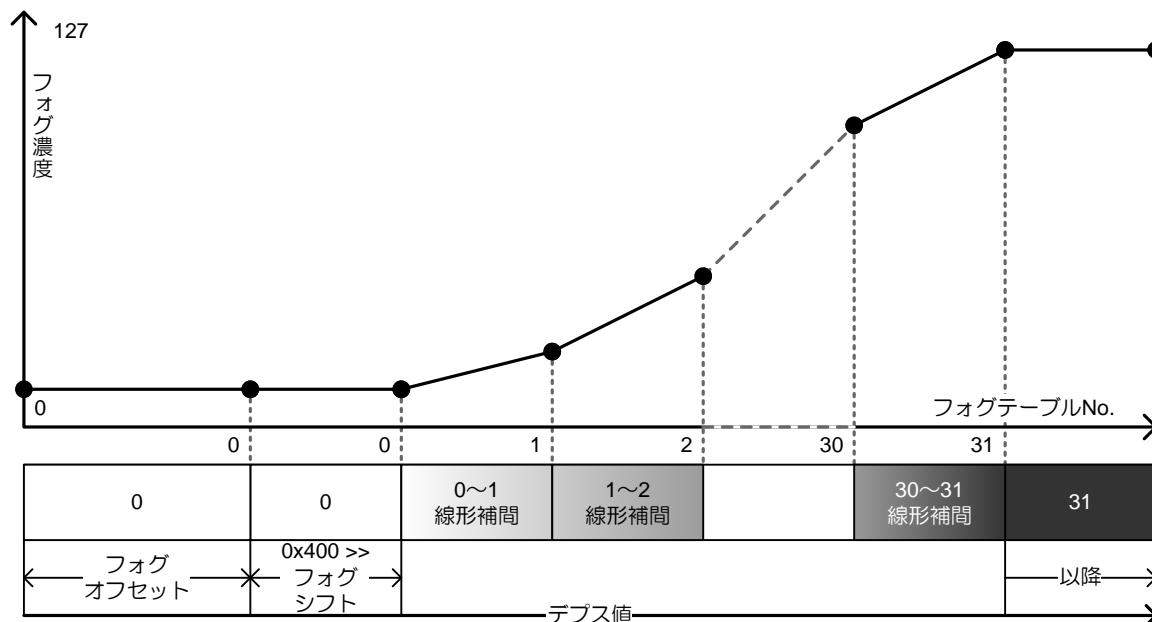


図 7-47 デプス値とフォグ濃度の関係

フォグシフトは DISP3DCNT レジスタ (3D 表示コントロールレジスタ) で設定します。

表 7-19 にフォグブレンディング演算式を示します。

フォグモード	「ピクセルのカラー値と α 値に フォグブレンディング」	「ピクセルの α 値のみに フォグブレンディング」
フォグ ブレンディング 演算式	$R = \{ f \times R_f + (128 - f) \times R_s \} / 128$ $G = \{ f \times G_f + (128 - f) \times G_s \} / 128$ $B = \{ f \times B_f + (128 - f) \times B_s \} / 128$ $A = \{ f \times A_f + (128 - f) \times A_s \} / 128$ <p>ただし、例外処理として $f = 127$ のときは $(R, G, B, A) = (R_f, G_f, B_f, A_f)$ が 適用されます。</p>	$R = R_s$ $G = G_s$ $B = B_s$ $A = \{ f \times A_f + (128 - f) \times A_s \} / 128$ <p>ただし、例外処理として $f = 127$ のときは $A = A_f$ が適用されます。</p>

(R, G, B, A) : カラーバッファに新しく格納されるカラー（演算結果の小数点以下は切り捨てられます）

(Rf, Gf, Bf, Af) : フォグカラー

(Rs, Gs, Bs, As) : カラーバッファのカラーに対してエッジマーキングを行った後のカラー

表 7-19 フォグブレンディング演算式

7.3.9.2 2D へのフォグの前処理

フォグがディセーブルであれば、カラーバッファの α 値が 0 でクリアされている領域は 2D 面との合成時に抜き領域として扱われ、抜き領域からは背後の 2D 面がそのままのカラーで見えることになります（「レンダリング後の 3D 面に適用できる 2D グラフィックス機能」参照）。

しかしフォグがイネーブルであれば、この抜き領域に対してもフォグブレンディングが行われ、カラーバッファの値が更新されます。

そしてこの後に 2D のカラー特殊効果（2D と 3D の α ブレンディング）機能によりカラーバッファと 2D 面の α ブレンディングを行うことによって、抜き領域から見えている背後の 2D 面に対してもフォグをかけることが可能になります。

このように、2D へのフォグブレンディングは 2D と 3D の α ブレンディングによって行います。

「2D 面との α ブレンディング」章を参照してください。

7.3.10 アンチエイリアシング

アンチエイリアシングは、前側カラーバッファにおけるエッジ部分のカラーを後側カラーバッファのカラーとブレンディングする機能です。
前側カラーバッファは一番手前のポリゴン、後側カラーバッファはそれより後ろに存在するすべてのポリゴン（クリアカラー含む）のレンダリング結果が格納されています。
アンチエイリアシングは不透明ポリゴンのエッジ部分にのみ適用されます。
アンチエイリアシングによってカラーバッファに新しく格納された α 値は、2D での α ブレンディング時にブレンディングの係数として使われるため、2D の背景であってもアンチエイリアシングがかかったような効果になります。

表 7-20 にアンチエイリアシングの演算式を示します。
アンチエイリアシングの概念図を図 7-48 に、LCD への出力イメージを図 7-49 に示します。

	アンチエイリアシング演算式
A2 が 1 以上のとき	$RA = \{ (AC+1) \times R1 + (31 - AC) \times R2 \} / 32$ $GA = \{ (AC+1) \times G1 + (31 - AC) \times G2 \} / 32$ $BA = \{ (AC+1) \times B1 + (31 - AC) \times B2 \} / 32$ $AA = \{ (AC+1) \times 31 + (31 - AC) \times A2 \} / 32$
A2 が 0 のとき	$RA = R1$ $GA = G1$ $BA = B1$ $AA = \{ (AC+1) \times 31 \} / 32$

(RA, GA, BA, AA) : カラーバッファに新しく格納されるカラー（演算結果の小数点以下は切り捨てられます）
(R1, G1, B1, 31) : 前側カラーバッファのカラー（アンチエイリアシングの対象は不透明ポリゴンのため α 値は 31）
(R2, G2, B2, A2) : 後側カラーバッファのカラー
AC : アンチエイリアシング係数（5 ビット）
アンチエイリアシング係数は、ほぼそのピクセルに対するポリゴンの占有率（面積）が適用されます。

表 7-20 アンチエイリアシングの演算式

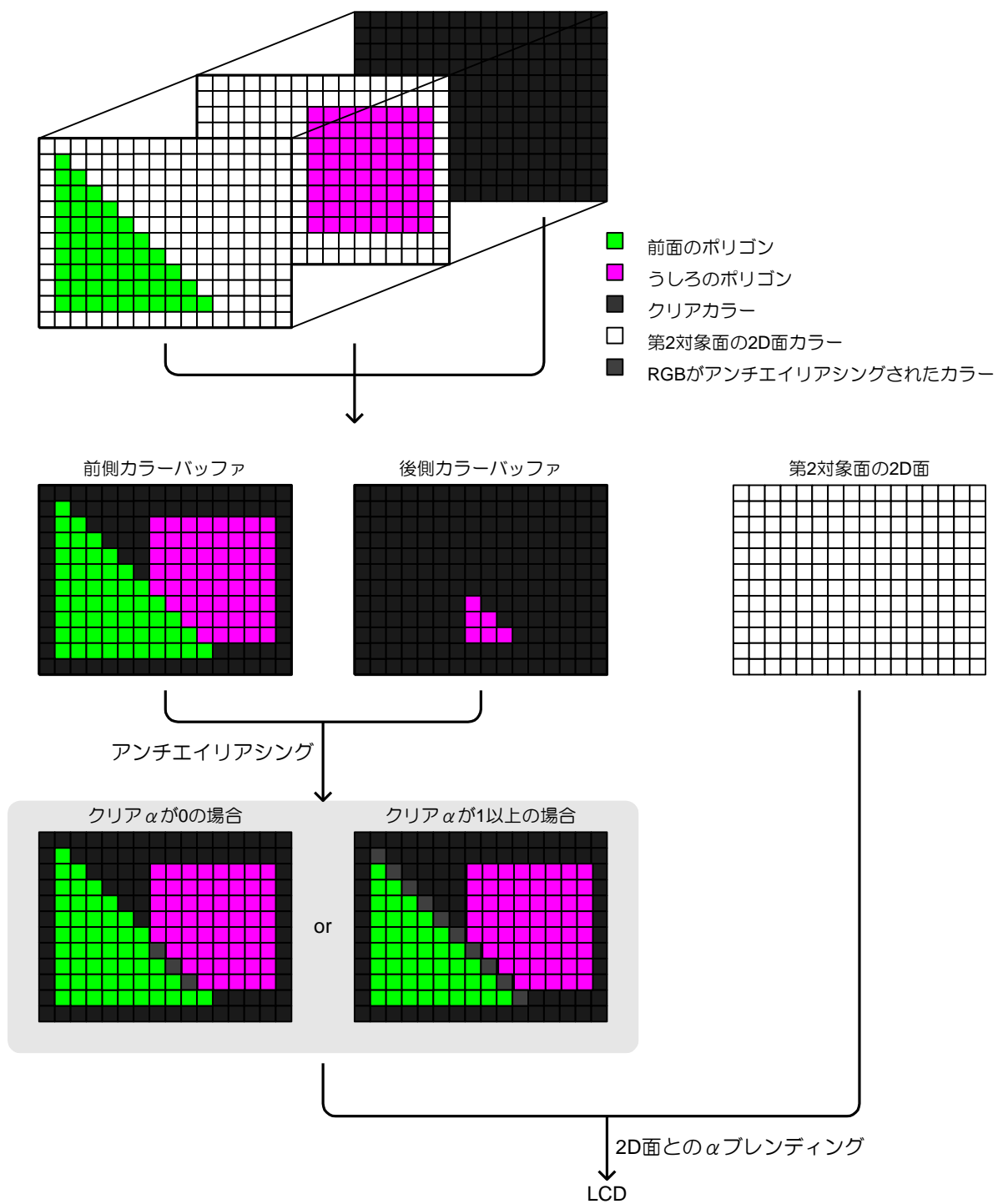


図 7-48 アンチエイリアシングの概念図

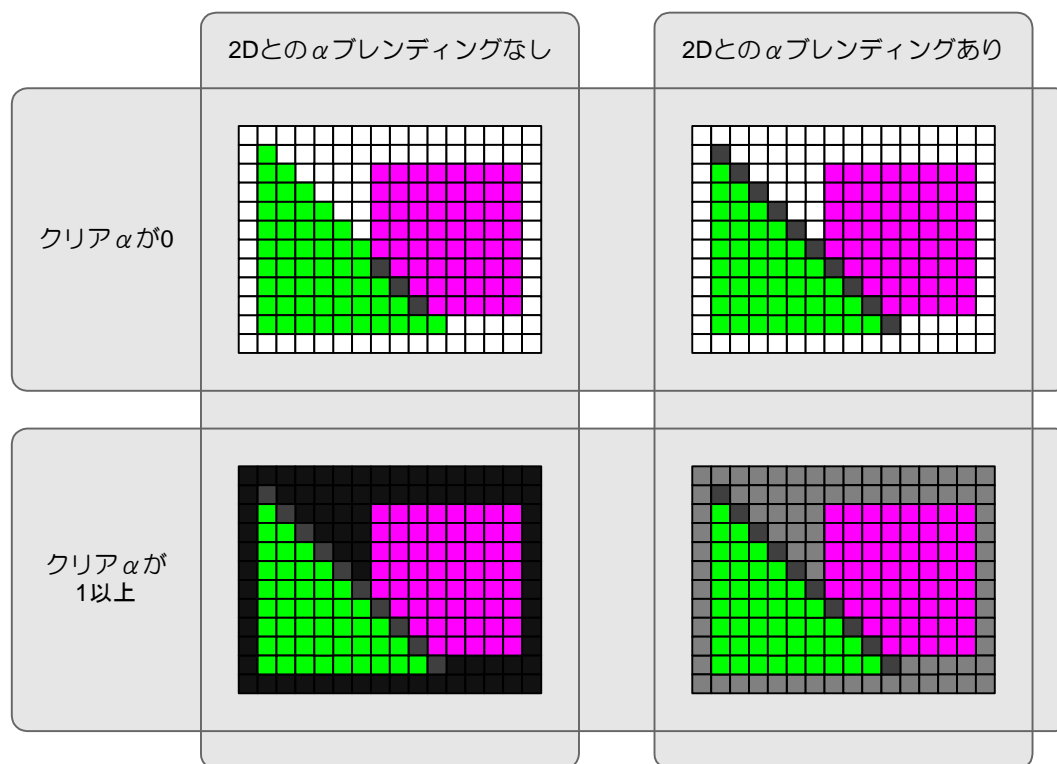


図 7-49 LCD への最終出力イメージ図（アンチエイリアシング）

図 7-49 は見た目のイメージ図です。実際の結果 RGB 値は表 7-21 を参照してください。

A2	2D面とのαブレンディング	アンチエイリアシング結果 (RGBA)		2D面とのαブレンディング結果 (RGB)	
		エッジ	背景	エッジ	背景
0	なし	(R1, G1, B1, AA)	(R2, G2, B2, 0)	(R1, G1, B1)	2D面のカラー
0	あり	(R1, G1, B1, AA)	(R2, G2, B2, 0)	(R1, G1, B1)と2D面のカラーとのαブレンディングカラー (αブレンディング係数AA)	2D面のカラー
1以上	なし	(RA, GA, BA, AA)	(R2, G2, B2, A2)	(RA, GA, BA)	(R2, G2, B2)
1以上	あり	(RA, GA, BA, AA)	(R2, G2, B2, A2)	(RA, GA, BA)と2D面のカラーとのαブレンディングカラー (αブレンディング係数AA)	(R2, G2, B2)と2D面のカラーとのαブレンディング結果 (αブレンディング係数A2)

ポリゴン内部は不透明（α値= 31）のため、2D と αブレンディングされませんので省略しています。

表 7-21 アンチエイリアシングと、2D 面とのαブレンディングとの関係

表 7-21 のように、A2 が 0 の場合はアンチエイリアシングによるブレンディングが α 値のみに適用され、RGB カラーはブレンディングされずそのままカラーバッファに書き込まれます。
 この場合は 2D 面との αブレンディングを行うことによって、RGB に反映させてください（アンチエイリアシングによる α 値が αブレンディングに適用されます）。

7.3.11 ステータス

レンダリング済みライン数カウントレジスタ

名称 RDLINES_COUNT アドレス 0x04000320 属性 R 初期値 0x0000

15	8	7	5	0
				RENDERED_LINES_MIN
				レンダリング済みライン数の最小値

● RENDERED_LINES_MIN[d05~d00]：レンダリング済みライン数の最小値（0～46）

「直前の描画フレーム中に、カラーバッファが最低何ラインまでになったか」を確認することができます。

このレジスタは V 周期毎に更新されます。

ラインバッファは 48 ライン分ありますが 2 ラインはカレントバッファになるため、レジスタの最大カウント値は 46 です。

この値によってラインズオーバー発生の有無を確認することはできませんが、ラインズオーバーの発生リスクがどのくらいであったかを知ることができます。

ラインズオーバー発生の有無を確認するには、3D 表示コントロールレジスタ（DISP3DCNT）の「カラーバッファ・アンダーフローフラグ」を参照してください。

注意事項

カウンタ値が 0 になるとレンダリングエンジンのライン描画が破綻（ラインズオーバー）する恐れがあります。

カウンタ値が 0 に近くなっている場合はジオメトリエンジンに送るポリゴン数を減らす等の対策を行いレンダリングエンジンにかかる負荷を減らしてください。

参考）描画バッファ方式の比較

● 一般的なフレームバッファ方式

描画用と表示用に、フレームバッファを複数（2～3）枚持ち、描画完了直後の V ブランク期間にバッファをスワップする方式です。

描画ポリゴンの数やピクセル数がレンダリングエンジンの処理能力よりも多いと、レンダリングが間に合わないためフレームレートが落ちてしまいます。

● TWL の FIFO ラインバッファ方式

描画と表示は同じ FIFO に対して行う方式です。FIFO は 48 ライン分のカラーデータを持つ容量があります。

表示時は LCD 表示タイミングに同期して FIFO からデータを読み出します。

水平方向にはドットクロックで読み出し、垂直方向には水平走査間隔（355 ドットクロック）で読み出すことになります。

1 ラインに描画させようとするポリゴンの数やピクセル数がレンダリングエンジンの処理能力よりも多いと、レンダリングが表示に間に合わないため表示が破綻してしまいます。

7.4 レンダリング後の 3D 面に適用できる 2D グラフィックス機能

TWL では、レンダリングが完了した 3D 面を直接 LCD へ表示せず、BG0 として表示します。

このため、2D グラフィックス機能の一部を適用した表現が可能です。

2D グラフィックス機能の基本的な仕様は「2D グラフィックス」を参照してください。

7.4.1 ラスタスクロール

2D 面とは違い、垂直方向のスクロールはできません。水平方向のみスクロールできます。

BG0 オフセット設定レジスタ

名称 BG0OFS アドレス 0x04000010 属性 W 初期値 0x0000

15								8	7	0	
								SH	INTEGER_H		
								H オフセット			

符号付き固定小数点数（符号+整数部 8 ビット）

● H[d08～d00] : H オフセット

水平方向の表示開始位置を変更できます。

2D 面とは違い d08 は符号ビットになり、-256～255 の範囲で指定可能です。

水平方向のスクロールによって表示画面がスクリーンを超えた部分は透明表示となります。（図 7-50 参照）

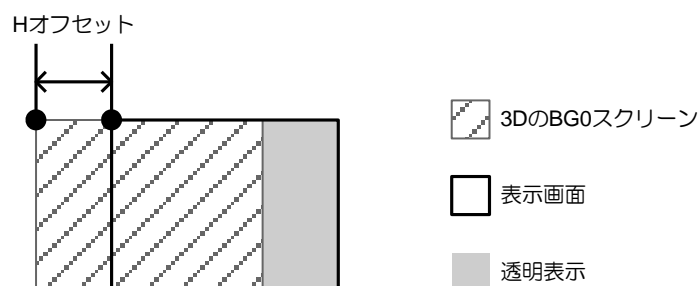


図 7-50 3D 面の H オフセット

7.4.2 2D 面との表示優先順位

BG0 コントロールレジスタ

名称 BG0CNT アドレス 0x04000008 属性 R/W 初期値 0x0000

15								8	7							1	0
																優先順位	

表示優先順位を調整することにより、3D 面の前後へ 2D 面を配置することができます。

「表示優先順位」の図を参照してください。

7.4.3 ウィンドウ

3D 面の BG0 に対してもウィンドウを適用できます。

ウィンドウ位置設定レジスタ

名称 WINxH (x=0, 1) アドレス 0x04000040, 0x04000042 属性 W 初期値 0x0000

15	8							7	0						
ウィンドウ左上 x 座標								ウィンドウ右下 x 座標							

名称 WINxV (x=0, 1) アドレス 0x04000044, 0x04000046 属性 W 初期値 0x0000

15	8							7	0						
ウィンドウ左上 y 座標								ウィンドウ右下 y 座標							

ウィンドウ内側コントロールレジスタ

名称 WININ アドレス 0x04000048 属性 R/W 初期値 0x0000

15	8							7	0						
		EFCT	OBJ	BG3	BG2	BG1	BG0			EFCT	OBJ	BG3	BG2	BG1	BG0
ウィンドウ 1 内側								ウィンドウ 0 内側							

ウィンドウ外側コントロールレジスタ

名称 WINOUT アドレス 0x0400004A 属性 R/W 初期値 0x0000

15	8							7	0						
		EFCT	OBJ	BG3	BG2	BG1	BG0			EFCT	OBJ	BG3	BG2	BG1	BG0
OBJ ウィンドウ内側								ウィンドウ (0, 1 および OBJ ウィンドウ) の外側							

ウィンドウコントロールレジスタのカラー特殊効果イネーブルフラグの設定に関わらず、3D 面が第一優先面になる場合の α ブレンディングは常に有効となります。

7.4.4 カラー特殊効果

各レジスタのパラメータに関する詳細は、2D グラフィックスの「カラー特殊効果」を参照してください。

7.4.4.1 2D 面との α ブレンディング

2D と 3D の α プレンディングや 2D へのフォグの後処理は、2D のカラー特殊効果機能における α プレンディングを使って行います。

前処理に関しては、「2D と 3D の α ブレンディングの前処理」、「2D へのフォグの前処理」を参照してください。

カラー特殊効果コントロールレジスタ

名称 BLDCNT アドレス 0x04000050 属性 R/W 初期値 0x0000

15		13			8			7		6		5		0		
		BD	OBJ	BG3	BG2	BG1	BG0	0	1	BD	OBJ	BG3	BG2	BG1	BG0	
		第 2 对象面					特殊効果選択		第 1 对象面							

- [d07～d06]：特殊効果選択

[d07]を0、[d06]を1にセットすることで α ブレンディングを行うことができます。

3D 面と 2D 面の α ブレンディングは、2 面の優先順位の関係によって処理が異なります。

第 1 対象面が 2D 面であった場合には基本仕様通り BLDALPHA レジスタの設定値が適用されます。

第1対象面が3D面であれば、カラーバッファへレンダリングされている α 値を用いて第2対象面との α ブレンディングを行います（ピクセル単位での α ブレンディング）。

注意事項

カラーバッファの α 値が0の部分は2Dの抜き領域と同等の扱いになるため α ブレンディングの対象から外され、カラーバッファの α 値が1以上の部分は α ブレンディングの対象に入ることにご注意ください。

カラー特殊効果・ α ブレンディング係数レジスタ

名称 BLDALPHA アドレス 0x04000052 属性 R/W 初期値 0x0000

15			12				8		7			4		0	
			EVB								EVA				

α ブレンディング処理に使用する係数を、BLDALPHA レジスタの EVA、EVB で設定します。

(EVA、EVB の値が 16 以上のときは 16 とみなされます)

7.4.4.2 輝度アップ/ダウン

カラー特殊効果コントロールレジスタ

名称 BLDCNT アドレス 0x04000050 属性 R/W 初期値 0x0000

15		13		8				7	6	5	0				
		BD	OBJ	BG3	BG2	BG1	BG0	1		BD	OBJ	BG3	BG2	BG1	BG0
第2対象面								特殊効果選択		第1対象面					

●[d07～d06]：特殊効果選択

[d07]に 1 をセットすることで、輝度変更処理を行うことができます。

[d06]が 0 の場合は輝度アップ、1 の場合は輝度ダウン処理が行われます。

第2対象面はすべて 0 をセットしておく必要があります。

カラー特殊効果・輝度変更係数レジスタ

名称 BLDY アドレス 0x04000054 属性 W 初期値 0x0000

15		8				7	4				0				
											EVY				

輝度変更処理に使用する係数を、BLDY レジスタの EVY で設定します。

(EVY の値が 16 以上のときは 16 とみなされます)

8. DMA

DMA は、CPU を介さず、DMA コントローラによりデータ転送を高速に行う機能です。

TWL の ARM9 バスには NITRO 互換の DMA チャンネルが 4 つ (DMA0~3)、DMA 間の調停方式の選択など新機能を搭載した新 DMA チャンネルが 4 つ (NDMA0~3) の合計 8 つの DMA チャンネルがあります。(ARM7 バスにも 8 つあります) 優先順位は DMA0 が最も高く、次いで DMA1、DMA2、DMA3 の順、NDMA0、NDMA1、NDMA2、NDMA3 となっていますが、ラウンドロビン方式ではバス所有権が循環的に移動するため、NDMA0~3 の間での優先順位がなくなります。現在動作中の DMA よりも優先順位の高い DMA の起動がかかると一時中断し、優先順位の高い DMA が実行されます。この後、優先順位の高い DMA が終了すると、中断していた DMA が続きから動作します。このため、限られた期間で確実に転送を行う必要があるものほど優先順位の高い DMA を使用してください。

なお、CPU は DMA 動作中に TCM およびキャッシュ以外の RAM へアクセスすることができません。
そのため、DMA が停止するまでの間、割り込み処理を TCM 以外で行っている場合は割り込みが遅延します。

8.1 NITRO 互換 DMA

NITRO 互換 DMA は、DMA 複数チャネル並列起動時の回路修正以外は NITRO から変更はありません。

DMAx ソースアドレスレジスタ (x=0~3)

名称 DMAxSAD (x=0~3)

アドレス 0x040000B0, 0x040000BC, 0x040000C8, 0x040000D4 属性 R/W 初期値 0x00000000

31	27	24	23	16	15	8	7	0
				DMA ソースアドレス				

DMAx デスティネーションアドレスレジスタ (x=0~3)

名称 DMAxDAD (x=0~3)

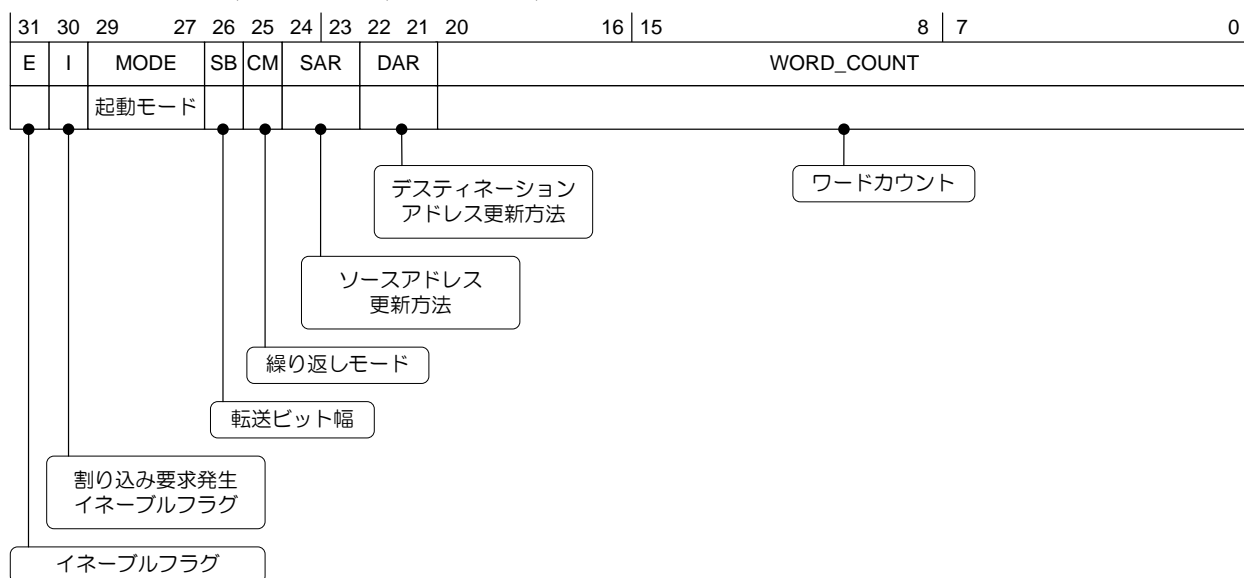
アドレス 0x040000B4, 0x040000C0, 0x040000CC, 0x040000D8 属性 R/W 初期値 0x00000000

31	27	24	23	16	15	8	7	0
				DMA デスティネーションアドレス				

DMAx コントロールレジスタ (x=0~3)

名称 DMAxCNT (x=0~3)

アドレス 0x040000B8, 0x040000C4, 0x040000D0, 0x040000DC 属性 R/W 初期値 0x00000000



● E[d31] : DMA イネーブルフラグ

0	ディセーブル
1	イネーブル

● SB[d26] : 転送ビット数選択

0	16 ビット
1	32 ビット

● I[d30] : 割り込み要求発生イネーブルフラグ

0	ディセーブル
1	イネーブル

● CM[d25] : 繰返しモード選択フラグ

0	繰返しモードでない
1	繰返しモード

● MODE[d29~d27] : DMA 起動モード選択

000	直ちに起動
001	V ブランクで起動
010	H ブランクで起動 (V ブランク期間中における H ブランクでは起動しません)
011	表示に同期して起動 (表示期間における各 H ライン描画開始に同期して起動)
100	メインメモリ表示
101	TWL / NITRO カード
110	カートリッジ
111	ジオメトリコマンド FIFO

● SAR[d24~d23] : ソースアドレス更新方法選択

00	インクリメント
01	デクリメント
10	固定
11	設定禁止

● DAR[d22~d21] : デスティネーションアドレス更新方法選択

00	インクリメント
01	デクリメント
10	固定
11	インクリメント/リロード

● WORD_COUNT[d20~d00] : ワードカウント
転送回数を指定します。

● 繰り返しモードについて

DMA を繰り返しモードに設定すると、起動モードで設定した条件が発生する度に自動的に DMA が起動します。
繰り返しモードでない場合は、ワードカウント分の転送が終了した時点で DMA が停止します。
繰り返しモードを解除するには、後述する注意事項の手順に従って DMA イネーブルフラグを 0 にセットしてください。

● アドレス更新方法に関して

アドレスの更新方法の処理内容は表 8-1 の通りです。

アドレス更新方法	処理の内容
インクリメント	1 回の転送毎に、アドレス値を 1 回分進めます
デクリメント	1 回の転送毎に、アドレス値を 1 回分戻します
固定	アドレス値は固定のままとします
インクリメント／リロード	1 回の転送毎にインクリメントを行い、ワードカウント分の転送終了時に、転送開始時のアドレス値に戻します

表 8-1 アドレス更新方法による処理の内容

注意事項

ソースあるいはデスティネーションにカートリッジ空間を設定した場合、アドレス更新方法を「固定」および「デクリメント」に設定することを禁止します（ハード未対応のため）。

● DMA 起動モードについて

「メインメモリ表示起動モード」

メインメモリ表示モードでは、DMA のソースアドレスはメインメモリのメモリ領域以外に設定しないでください。また転送ビット選択は必ず 32 ビットモードにし、ワードカウント値を必ず 4 に設定してください。

「ジオメトリコマンド FIFO 起動モード」

ジオメトリコマンド FIFO が半分未満になると DMA が起動し、112 ワード※分の転送を行い、この繰り返しによって転送量がワードカウントに達すると終了します。

（※ コマンドバック時は展開前のワード数です）

● DMA の起動・停止に関する注意事項

1) DMA を起動したとき

DMA イネーブルフラグをセットしてから DMA が起動するまで、システムクロック（33.514Mhz）換算で 2 サイクルの遅延が発生します。この間に DMA 関連レジスタにアクセスすることは DMA の誤動作につながる危険がありますので、他の処理を行うか、ダミーのロード命令を挿入する等の手段で避けるようにしてください（メインプロセッサはロード命令がシステムクロック換算で 1/2 サイクル^{※1}で実行されるため、同一レジスタへのロード命令が 2 つ^{※2}以上必要です）。

2) DMA を停止するとき

DMA は、DMA のスタートトリガとなる信号が発せられると起動します。ここで、スタートトリガが発生すると同時に CPU によって DMA をディセーブルにすると DMA が暴走してしまうことがあります。DMA をディセーブルにする際は、スタートトリガのタイミングから 4 サイクル^{※3}以上ずらすようにしてください。

2-1) DMA リピート機能を使用しない場合

DMA を 1 回実行した後に自動的に停止しますので、DMA イネーブルフラグを強制的にクリアせず 0 になるのを待つようにしてください。

2-2) DMA リピート機能を使用した場合

DMA のスタートトリガとなる信号と、CPU で DMA イネーブルフラグをクリアするタイミングを 4 サイクル^{※3}以上ずらすようにしてください。例えば、DMA 終了時に発生する割り込みを用い、次のスタートトリガ発生前に DMA イネーブルフラグをクリアする等の手段で DMA を安全に停止させることができます。

この方法を用いることができない場合、下記の方法で DMA を停止するようにしてください。

2-3) H ブランク、V ブランク自動起動モードにおいて DMA を停止する方法

V ブランク期間中では DMA が停止状態であり、スタートトリガが発生しませんので、安全に DMA イネーブルフラグをクリアできます。この方法を用いることができない場合、下記の手順 1～3 を用います。

手順 1 DMA コントロールレジスタに表 8-2 の設定を 16 ビット書き込みします

設定項目	設定内容
DMA イネーブルフラグ	1（イネーブル）
DMA 起動タイミング	00（「直ちに起動」モード）
DMA 繰り返しモード	0（繰り返しモードでない）
その他のビット	変更しません

表 8-2 レジスタ設定（手順 1）

手順 2 4 サイクル^{※3}以上の処理を行います。

例（NITRO 互換モード時）：

（NOP 命令 3 回 または LDR 命令）+ 手順 3 による STR 命令の 1 サイクル目により、計 4 サイクルとなります。（STR 命令によって実際に書き込まれるのは 2 サイクル目です）

手順 3 DMA コントロールレジスタに表 8-3 の設定を 16 ビット書き込みし、DMA を停止します。

設定する項目	設定内容
DMA イネーブルフラグ	0（ディセーブル）
DMA 起動タイミング	00（「直ちに起動」モード）
DMA 繰り返しモード	0（繰り返しモードでない）
その他のビット	変更しません

表 8-3 レジスタ設定（手順 3）

注意事項

上記の手順 1 によって DMA が 1 回余分に起動してしまう場合がありますので留意してください。

※1 TWL モードの倍速動作時は 1/4 サイクルです。

※2 TWL モードの倍速動作時は 4 つです。

※3 TWL モードの倍速動作時は 8 サイクルです。

TWL モードではシステム設定により NITRO 互換 DMA コントローラの回路修正の有効／無効を指定することができます。有効とした場合は、以下の注意事項を考慮する必要はありません（NITRO 互換モードではこの問題は回避されません）。

● **ARM9 システムバスにおける DMA 複数チャンネル並列起動時に関する注意事項**

ARM946E-S がメインメモリなど 1 システムサイクル(33.514MHz)にてアクセスできない領域へアクセスを開始すると同時に優先度が低い ARM9-DMA が（自動）起動し、更にその直後に優先順位が高い ARM9-DMA の自動起動が発生すると、優先順位が高い DMA が暴走します。なお、ARM7 側ではシステムバスの仕様の違いにより症状は発生しません。

対策方法

表 8-4 に示す DMA 並列起動カテゴリ表中の、カテゴリ 3 に分類されているものについて併用を禁止し、更に DMA の起動を TCM から行うようにしてください。なお、V ブランク起動と H ブランク起動の組み合わせについては不具合が起きませんので併用可能とします。

並列起動カテゴリ番号	DMA 内容
1	直ちに起動
2	ジオメトリコマンド FIFO（通常）
3	ジオメトリコマンド FIFO（自動起動） V ブランク起動（※H ブランク起動との併用可能） H ブランク起動（※V ブランク起動との併用および H ブランクの複数起動可能） 表示同期 メインメモリ表示 カード

表 8-4 ARM9-DMA 並列起動カテゴリ表

NDMAx ソースアドレスレジスタ (x=0~3)

名称 NDMAxSAD (x=0~3)

アドレス 0x04004104, 0x04004120, 0x0400413C, 0x04004158 属性 R/W 初期値 0x00000000

31	24	23	16	15	8	7	2	0
DMASRC								
DMA ソースアドレス								

NDMAx デスティネーションアドレスレジスタ (x=0~3)

名称 NDMAxDAD (x=0~3)

アドレス 0x04004108, 0x04004124, 0x04004140, 0x0400415C 属性 R/W 初期値 0x00000000

31	24	23	16	15	8	7	2	0
DMADEST								
DMA デスティネーションアドレス								

- DMASRC, DMADEST[d31~d02] : ソース (デスティネーション) アドレス
新 DMA の DMA ソースアドレスと DMA デスティネーションアドレスの下位 2 ビットは常に 0 が出力されます。

NDMAx 総転送ワードカウントレジスタ (x=0~3)

名称 NDMAxTCNT (x=0~3)

アドレス 0x0400410C, 0x04004128, 0x04004144, 0x04004160 属性 R/W 初期値 0x00000000

31	27	24	23	16	15	8	7	0
TOTALCNT								
総転送ワード数								

- TOTALCNT[d27~d00] : 総転送ワード数
総転送ワード数に 0x00000000 を設定した場合は、0x10000000 ワードの設定となります。総転送ワード数は、転送ワード数およびブロック転送ワード数の整数倍でも、それ以上の値でなくてもかまいません。

NDMAx ワードカウントレジスタ (x=0~3)

名称 NDMAxWCNT (x=0~3)

アドレス 0x04004110, 0x0400412C, 0x04004148, 0x04004164 属性 R/W 初期値 0x00000000

31	24	23	16	15	8	7	0
WORDCNT							
転送ワード数							

- WORDCNT[d23~d00] : 転送ワード数
転送ワード数に 0x00000000 を設定した場合は、0x10000000 ワードの設定となります。転送ワード数は、ブロック転送ワード数の整数倍でも、それ以上の値でなくてもかまいません。

NDMAx ブロック転送インターバルレジスタ (x=0~3)

名称 NDMAxBCNT (x=0~3)

アドレス 0x04004114, 0x04004130, 0x0400414C, 0x04004168 属性 R/W 初期値 0x00000000

31	24	23	17	16	15	8	7	0
PS					ICNT			
					インターバルタイマ			

- PS[d17~d16] : プリスケラ選択

00	システムクロック (33.514MHz)	10	システムクロックの 16 分周
01	システムクロックの 4 分周	11	システムクロックの 64 分周
- ICNT[d15~d00] : インターバルタイマ
インターバルタイマに 0x0000 を設定した場合は、指定された転送ワード数を最後まで連続して転送します。

NDMAx フィルデータレジスタ (x=0~3)

名称 NDMAxFDATA (x=0~3)

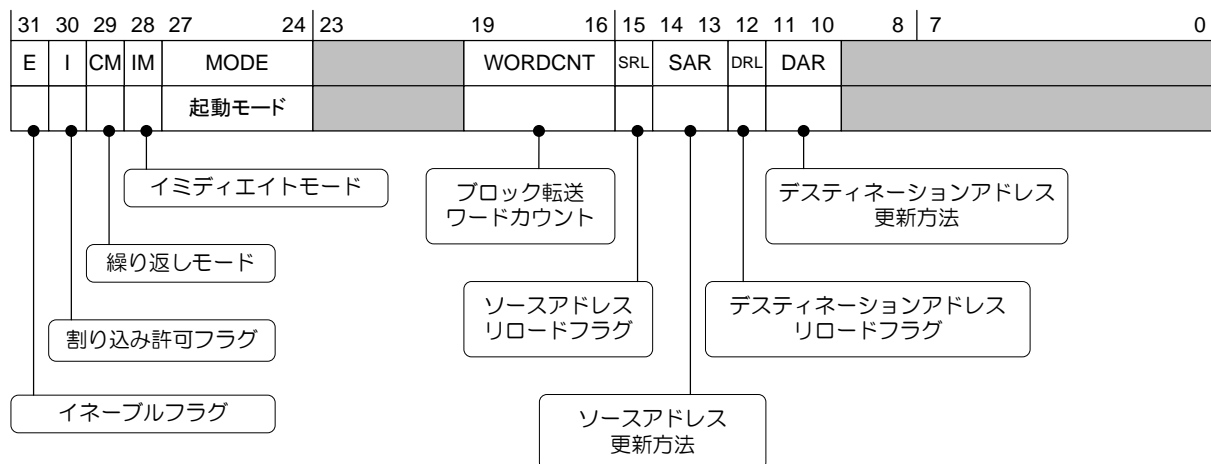
アドレス 0x04004118, 0x04004134, 0x04004150, 0x0400416C 属性 R/W 初期値 0x00000000

31	24	23	16	15	8	7	0
FDATA							
フィルデータ							

NDMAx コントロールレジスタ (x=0~3)

名称 NDMAxCNT (x=0~3)

アドレス 0x0400411C, 0x04004138, 0x04004154, 0x04004170 属性 R/W 初期値 0x00000000



● E[d31] : DMA イネーブルフラグ

0	ディセーブル
1	イネーブル

● I[d30] : 割り込み許可フラグ

0	ディセーブル
1	イネーブル

● CM[d29] : 繰り返しモード選択フラグ

0	繰り返しモードでない
1	繰り返しモード

● IM[d28] : イミディエイトモード選択フラグ

0	イミディエイトモードでない
1	イミディエイトモード

● MODE[d27~d24] : DMA 起動モード選択

0000	タイマ 0
0001	タイマ 1
0010	タイマ 2
0011	タイマ 3
0100	TWL / NITRO カード
0101	設定禁止
0110	V ブランクで起動
0111	H ブランクで起動 (V ブランク期間中における H ブランクでは起動しません)
1000	表示に同期して起動 (表示期間における各 H ライン描画開始に同期して起動)
1001	ワーク RAM
1010	ジオメトリコマンド FIFO
1011	カメラ
1100	設定禁止
1101	設定禁止
1110	設定禁止
1111	設定禁止

● WORCNT[d19~d16]：ブロック転送ワードカウント

0000	1 ワード
0001	2 ワード
0010	4 ワード
0011	8 ワード
0100	16 ワード
0101	32 ワード
0110	64 ワード
0111	128 ワード
1000	256 ワード
1001	512 ワード
1010	1,024 ワード
1011	2,048 ワード
1100	4,096 ワード
1101	8,192 ワード
1110	16,384 ワード
1111	32,768 ワード

● SRL[d15]：ソースアドレスリロードフラグ

0	リロードしない
1	リロードする

● SAR[d14~d13]：ソースアドレス更新方法選択

00	インクリメント
01	デクリメント
10	固定アドレス
11	アドレスなし

● DRL[d12]：デスティネーションアドレス
リロードフラグ

0	リロードしない
1	リロードする

● DAR[d11~d10]：デスティネーションアドレス
更新方法選択

00	インクリメント
01	デクリメント
10	固定アドレス
11	設定禁止

● 調停方式の違いについて

グローバルコントロールレジスタ（NDMAGCNT）で設定することのできる DMA 調停方式には、NDMA0>NDMA1>NDMA2>NDMA3 の順に優先順位がつけられた固定方式と、NDMA0~3 を循環的にバス所有権が移動するラウンドロビン方式があります。

固定方式では、起動中の DMA 転送よりも優先順位の高い起動要求が来ると、指定された転送ワード数のブロック転送が完了した時点で一時中断となり、優先順位の高い DMA 転送が開始されます。逆に、起動中の DMA 転送よりも優先順位の低い起動要求は、起動中の DMA 転送が完了するまで実行が保留されます。

ラウンドロビン方式では、起動要求のかかった DMA チャンネルを調べ、各チャンネルの指定された転送ワード数のブロック転送が完了した時点で、起動中の DMA チャンネルを含めて NDMA0、NDMA1、NDMA2、NDMA3、DSP または ARM9 の順番にバス所有権を移動させて各チャンネルの DMA 転送および DSP、ARM9 のバスアクセスを開始します。

また、ラウンドロビン方式では、DSP または ARM9 のバスアクセス可能なサイクル数をグローバルコントロールレジスタのサイクル選択で指定することができます。

● イミディエイトモードと繰り返しモードについて

イミディエイトモードはイミディエイトモードを選択し、イネーブルフラグをイネーブルに設定することで起動します。イミディエイトモードで起動された DMA 転送は、転送ワードカウンタレジスタ（NDMAxWCNT）で指定された転送ワード数の転送が終了した時点で、イネーブルフラグがディセーブルに設定されます。イミディエイトモードで起動された場合は、繰り返しモード選択フラグと総転送ワード数の設定は無視されます。

繰り返しモードは繰り返しモードを選択し、イネーブルフラグをイネーブルに設定後、DMA 起動モード選択で選択されたハードウェアの DMA 起動要求で起動します。繰り返しモードで起動された DMA 転送は、転送ワード数の転送をイネーブルフラグがディセーブルに設定されるまで繰り返します。繰り返しモードで起動された場合は、総転送ワード数の設定は無視されます。

イミディエイトモードでも繰り返しモードでもない DMA 転送は、転送ワード数の転送を総転送ワード数まで実行し、終了後にイネーブルフラグをディセーブルに設定します。

● ブロック転送について

新 DMA の DMA 転送は、転送を開始すると選択されたブロック転送ワード数の転送が完了するまでバスを占有し、ブロック転送サイクルが分断されることはありません。

また、1 回目のブロック転送が終了すると、次のブロック転送の開始はインターバルタイマで指定されたサイクルの後となります。インターバルタイマはダウンカウンタで、選択されたプリスケアラ毎にカウントダウンされ、0 となった時点で次のブロック転送の DMA 起動要求が発生します。なお、インターバルタイマに 0 を設定した場合は、指定された転送ワード数を最後まで連続して転送します。

- アドレスのリロードについて

ソースアドレスおよびデスティネーションアドレスをリロードするに設定した場合は、転送ワード数の転送終了後にインクリメントまたはデクリメントされたアドレスを転送開始時のアドレスに設定しなおします。

- フィルデータレジスタについて

ソースアドレスの更新方法に「アドレスなし」を選択すると、リードサイクルを省略して、デスティネーションアドレスへのライトサイクルのみでフィルデータレジスタ（NDMAxFDATA）の設定値を書き込むことができます。ライトサイクルのみのため、通常の DMA 転送に比べて高速に領域を固定値でクリアすることができます。

9. タイマ

TWL では ARM9 側に 16 ビットタイマを 4 チャンネル装備しています。
タイマをイネーブルにすると、コントロールレジスタで指定したプリスケアラ（分周器）の周期に従ってカウントレジスタをカウントアップします。
カウントレジスタがオーバーフローしたタイミングで割り込み要求を発生させることができます。
カウントレジスタがオーバーフローするとカウント開始時に設定した値がリロードされ、カウントアップが再開されます。

タイマ 0 カウントレジスタ

名称 TM0CNT_L アドレス 0x04000100 属性 R/W 初期値 0x0000

15								8	7								0
タイマ 0 カウンタ																	

● [d15～d00]：タイマ 0 カウンタ

タイマ 0 コントロールレジスタ

名称 TM0CNT_H アドレス 0x04000102 属性 R/W 初期値 0x0000

15								8	7	6								1	0
									E	I								PS	
																		プリスケアラ	

● [d07～d00]：タイマ 0 コントロール

● E[d07]：タイマ 0 イネーブルフラグ

0	ディセーブル
1	イネーブル

● I[d06]：割り込み要求発生イネーブルフラグ

0	ディセーブル
1	イネーブル

● PS[d01～d00]：プリスケアラ選択フラグ

00	システムクロック（33.514Mhz）
01	システムクロックの 64 分周
10	システムクロックの 256 分周
11	システムクロックの 1,024 分周

タイマ x カウントレジスタ (x=1~3)

名称 TMxCNT_L (x = 1~3) アドレス 0x04000104, 0x04000108, 0x0400010C 属性 R/W 初期値 0x0000

15								8		7								0
タイマ x カウンタ																		

- [d15~d00] : タイマ x カウンタ

タイマ x コントロールレジスタ (x=1~3)

名称 TMxCNT_H (x = 1~3) アドレス 0x04000106, 0x0400010A, 0x0400010E 属性 R/W 初期値 0x0000

15								8		7	6					2	1	0
										E	I					CH		PS
																多段		プリスケアラ

- [d07~d00] : タイマ x コントロール
 - E[d07] : タイマ x イネーブルフラグ

0	ディセーブル
1	イネーブル

- I[d06] : 割り込み要求発生イネーブルフラグ

0	ディセーブル
1	イネーブル

- CH[d02] : 多段カウンタ選択フラグ

0	プリスケアラの設定に従う
1	プリスケアラの設定に関わらず、 タイマ (x-1) のオーバーフローで カウントアップする

- PS[d01~d00] : プリスケアラ選択フラグ

00	システムクロック (33.514Mhz)
01	システムクロックの 64 分周
10	システムクロックの 256 分周
11	システムクロックの 1,024 分周

10. 割り込み

メインプロセッサである ARM9 に対するハードウェア割り込みを以下に説明します。

各ハードウェアからの割り込み要求信号が発生すると、割り込みリクエストレジスタのハードウェアに対応するビットがセットされ、割り込みがイネーブルであれば CPU へ割り込み発生が通知されます。

各ハードウェアの割り込み要求信号に対して割り込みイネーブルレジスタによって、個別にディセーブルにすることができます。

10.1 割り込みマスターイネーブルレジスタ

割り込み全体をディセーブルにすることができます。

割り込みをすべてディセーブルにするか、割り込みイネーブルレジスタの設定をイネーブルにするかを設定します。

割り込みマスターイネーブルレジスタ

名称 IME アドレス 0x04000208 属性 R / W 初期値 0x0000

15	8	7	0
			IME

● IME[d00] : 割り込みマスターイネーブルフラグ

0	すべての割り込みをディセーブル
1	割り込みイネーブルレジスタの設定をイネーブルにする

10.2 割り込みイネーブルレジスタ

各ハードウェアの割り込み要求を個別にディセーブルにすることができます。

各ビットをセットすることで、対応するハードウェアからの割り込み要求を許可します。

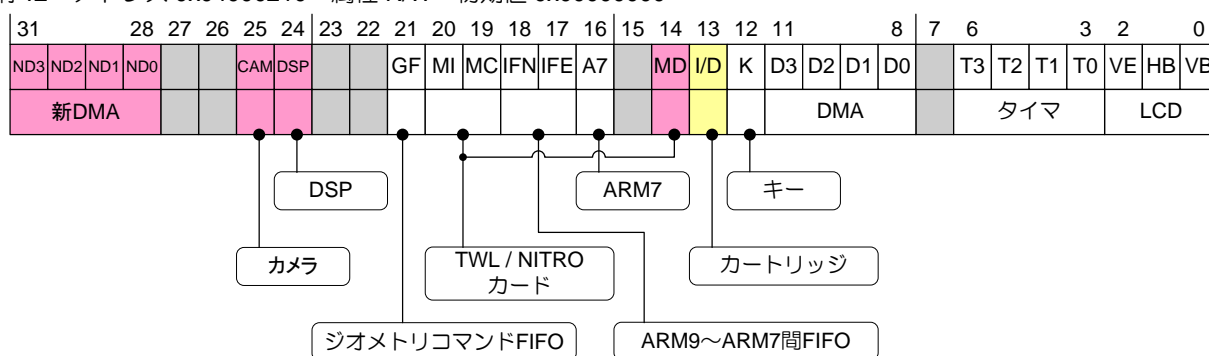
逆に、各ビットをリセットすることで、対応するハードウェアからの割り込み要求を禁止します。

レジスタ表でピンク色に塗られたビットは TWL で新たに追加されたビットです。TWL モードのシステム設定で拡張機能を OFF にしている場合は、0 固定となります。

レジスタ表で黄色に塗られたビットは TWL で仕様が変更されたビットです。

割り込みイネーブルレジスタ

名称 IE アドレス 0x04000210 属性 R/W 初期値 0x00000000



- ND3[d31]: 新 DMA3 割り込み許可フラグ
- ND2[d30]: 新 DMA2 割り込み許可フラグ
- ND1[d29]: 新 DMA1 割り込み許可フラグ
- ND0[d28]: 新 DMA0 割り込み許可フラグ
「DMA」を参照してください。
- CAM[d25]: カメラ割り込み許可フラグ
- DSP[d24]: DSP 割り込み許可フラグ
- GF[d21]: ジオメトリコマンド FIFO 割り込み許可フラグ
「3D グラフィックス」の「ステータス」を参照してください。
- MI[d20]: TWL / NITRO カード IREQ_MC 割り込み許可フラグ
- MC[d19]: TWL / NITRO カードデータ転送終了割り込み許可フラグ
- IFN[d18]: ARM9~ARM7 間 FIFO ノットエンプティ割り込み許可フラグ
- IFE[d17]: ARM9~ARM7 間 FIFO エンプティ割り込み許可フラグ
- A7[d16]: ARM7 割り込み許可フラグ
- MD[d14]: TWL / NITRO カード MC_DET 割り込み許可フラグ
- I/D[d13]: カートリッジ IREQ/DREQ 割り込み許可フラグ
カートリッジ機能が削除されたため、この割り込みは発生しません。
- K[d12]: キー割り込み許可フラグ
「キー」の「キー入力への割り込み処理」を参照してください。
- D3[d11]: DMA3 割り込み許可フラグ
- D2[d10]: DMA2 割り込み許可フラグ
- D1[d09]: DMA1 割り込み許可フラグ
- D0[d08]: DMA0 割り込み許可フラグ
「DMA」を参照してください。
- T3[d06]: タイマ 3 割り込み許可フラグ
- T2[d05]: タイマ 2 割り込み許可フラグ
- T1[d04]: タイマ 1 割り込み許可フラグ
- T0[d03]: タイマ 0 割り込み許可フラグ
「タイマ」を参照してください。
- VE[d02]: V カウンター致割り込み許可フラグ
- HB[d01]: H ブランク割り込み許可フラグ
- VB[d00]: V ブランク割り込み許可フラグ
「表示」の「表示ステータス」を参照してください。

10.3 割り込みリクエストレジスタ

各ハードウェアの割り込み要求が発生すると、割り込みリクエストレジスタのハードウェアに対応するビットがセットされます。

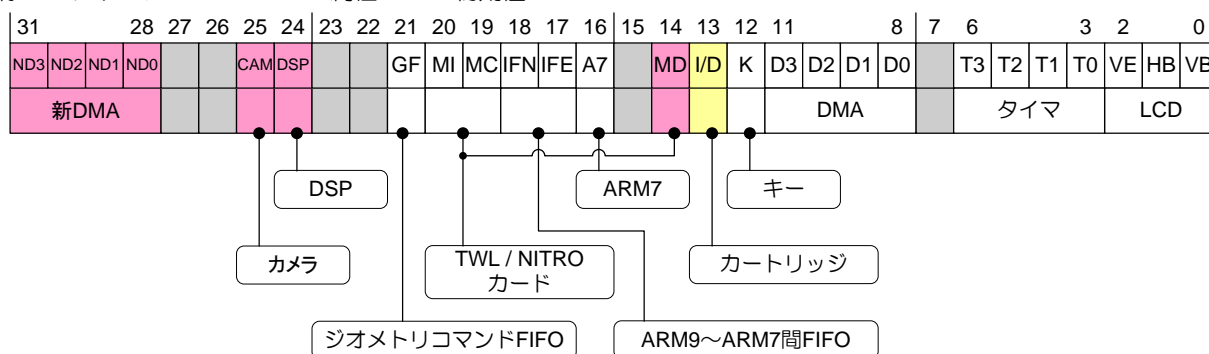
また、割り込みリクエストフラグがセットされているビットに 1 を書き込むと、その割り込みリクエストフラグをリセットすることができます。

レジスタ表でピンク色に塗られたビットは TWL で新たに追加されたビットです。TWL モードのシステム設定で拡張機能を OFF にしている場合は、0 固定となります。

レジスタ表で黄色に塗られたビットは TWL で仕様が変更されたビットです。

割り込みリクエストレジスタ

名称 IF アドレス 0x04000214 属性 R/W 初期値 0x00000000



- ND3[d31]: 新 DMA3 割り込み要求フラグ
- ND2[d30]: 新 DMA2 割り込み要求フラグ
- ND1[d29]: 新 DMA1 割り込み要求フラグ
- ND0[d28]: 新 DMA0 割り込み要求フラグ
「DMA」を参照してください。
- CAM[d25]: カメラ割り込み要求フラグ
- DSP[d24]: DSP 割り込み要求フラグ
- GF[d21]: ジオメトリコマンド FIFO 割り込み要求フラグ
「3D グラフィックス」の「ステータス」を参照してください。
- MI[d20]: TWL / NITRO カード IREQ_MC 割り込み要求フラグ
- MC[d19]: TWL / NITRO カードデータ転送終了割り込み要求フラグ
- IFN[d18]: ARM9~ARM7 間 FIFO ノットエンプティ割り込み要求フラグ
- IFE[d17]: ARM9~ARM7 間 FIFO エンプティ割り込み要求フラグ
- A7[d16]: ARM7 割り込み要求フラグ
- MD[d14]: TWL / NITRO カード MC_DET 割り込み要求フラグ
- I/D[d13]: カートリッジ IREQ/DREQ 割り込み要求フラグ
カートリッジ機能が削除されたため、この割り込みは発生しません。
- K[d12]: キー割り込み要求フラグ
「キー」の「キー入力への割り込み処理」を参照してください。
- D3[d11]: DMA3 割り込み要求フラグ
- D2[d10]: DMA2 割り込み要求フラグ
- D1[d09]: DMA1 割り込み要求フラグ
- D0[d08]: DMA0 割り込み要求フラグ
「DMA」を参照してください。
- T3[d06]: タイマ 3 割り込み要求フラグ
- T2[d05]: タイマ 2 割り込み要求フラグ
- T1[d04]: タイマ 1 割り込み要求フラグ
- T0[d03]: タイマ 0 割り込み要求フラグ
「タイマ」を参照してください。
- VE[d02]: V カウンター一致割り込み要求フラグ
- HB[d01]: H ブランク割り込み要求フラグ
- VB[d00]: V ブランク割り込み要求フラグ
「表示」の「表示ステータス」を参照してください。

10.4 割り込みに関する注意事項

10.4.1 IME および IE のクリア

IME や IE レジスタの各フラグをクリアする命令実行中にも、該当する割り込みは発生します。
IE のフラグをクリアする場合は、あらかじめ IME をクリアして、割り込みチェックの不整合が発生しないよう注意してください。

10.4.2 多重割り込み

IME のクリアと割り込みのタイミングが一致した場合、その割り込み中は多重割り込みが掛からなくなりますので、割り込みルーチン中で IME を退避後に IME をセット（許可）するようにしてください。

10.4.3 DMA 動作中の割り込み遅延

CPU は DMA 動作中に TCM およびキャッシュ以外の RAM へアクセスすることができません。
そのため、DMA が停止するまでの間、割り込み処理を TCM 以外で行っている場合は割り込みが遅延します。

10.4.4 ARM7 からの割り込み

A7、IFE、IFN の 3 つの割り込みはサブプロセッサとの通信のためサブプロセッサの API が使用しています。
これらの割り込みを禁止したり、割り込みリクエストをリセットしたりすると正しくサブプロセッサ API が動作しません。

10.4.5 DSP からの割り込み

DSP 割り込みは DSP との通信のため DSP の API が使用しています。
これらの割り込みを禁止したり、割り込みリクエストをリセットしたりすると正しく DSP API が動作しません。

10.4.6 カメラからの割り込み

CAM 割り込みはカメラとの通信のためカメラの API が使用しています。
これらの割り込みを禁止したり、割り込みリクエストをリセットしたりすると正しくカメラ API が動作しません。

11. パワーマネジメント

NITRO と同様に、TWL はパワーマネジメントの API を使用することにより、アプリケーションからスリープモードへの移行、各種回路の電源制御、バッテリー低下状態のチェック、本体の開閉状態のチェックをすることができます。

11.1 スリープモード

パワーマネジメントの API を使用することにより、アプリケーションでスリープモードへ移行することができます。スリープモードでは TWL プロセッサのすべての回路が停止します。LCD およびサウンドの電源は OFF となり、非表示かつ無音状態となります。ただし、TWL プロセッサ内部メモリおよびメインメモリの内容については保持されます。

スリープモード時はパワーLED が低速点滅（明滅）します。
スリープモード時の電池寿命はフル充電時で約 10 日間となります。

スリープモードからの復帰要因とタイミングは表 11-1 の通りです。

復帰要因	タイミング
TWL 本体を開く	閉じられていたパネルが開かれたとき
RTC のアラーム機能	設定されていたアラーム時刻になったとき
TWL / NITRO カード	TWL / NITRO カードが誤って抜かれたとき
キー入力	X、Y キーを除く、あらかじめ設定したキーが押されたとき

表 11-1 スリープモードからの復帰要因とタイミング

注意事項 1

カートリッジスロットが削除され、カートリッジからの割り込みが常に発生するため、カートリッジの割り込みを許可している状態ではスリープモードに移行することができません。

注意事項 2

スリープモード時には、システム全体の状態がアクティブモード時とは異なるため、アクティブモード時と同様なシャットダウン処理を行うこと（保証すること）が困難です。そのため、スリープモード時にはシャットダウン処理を行わず、スリープモード移行時に必要な処理を事前に行うようにしています。
TWL では、スリープモード移行時に、パワーマネジメントの API で登録したシャットダウン処理が自動的に実行されます。そのため、スリープモード時に突然電源が落とされても、結果としてアクティブモードで電源を落としたときと同じ振る舞いになります。

11.2 各種電源制御

サウンド、LCD バックライト、LCD、マイク、システム、グラフィックスの電源供給を制御することができます。

11.2.1 サウンド

サウンド回路への電源供給を制御することができます。ただし、現在その API は非公開となっております。

11.2.2 LCD バックライト

パワーマネジメントの API を使用して上画面 LCD、下画面 LCD それぞれのバックライトへの電源供給を制御することができます。

アプリケーションで LCD 画面を 1 画面しか使用しない場合は、使用しない LCD 画面のバックライトへの電源供給を OFF にすることでバッテリーの消費を抑えることができます。

電池寿命を考慮して、ゲームプレイせず、無線待機をしない状態で両方の LCD を表示させずに LCD バックライトへの電源供給を OFF にする場合、より効率的に電源供給を抑えることのできるスリープモードへ移行することを推奨します。

11.2.3 LCD

パワーマネジメントの API を使用して上画面および下画面の両方の LCD への電源供給を制御することができます。また、LCD バックライトの設定に関わらずバックライトも消灯します。なお、LCD バックライトの設定は保持されます。

アプリケーションでワイヤレス通信の待ち受けを行う場合など LCD 画面表示しない場合は、LCD への電源供給を OFF にすることでバッテリーの消費を抑えることができます。

電池寿命を考慮して、無線待機をしない状態で LCD への電源供給を OFF にする場合、より効率的に電源供給を抑えることのできるスリープモードへ移行することを推奨します。

注意事項

次頁のグラフィックスパワーコントロールレジスタを使用することで、直接 LCD への電源供給を制御することができますが、LCD の ON/OFF のタイミングによっては LCD 回路が破壊されてしまう可能性がありますので、レジスタの値を直接変更することは禁止します。LCD への電源供給制御は必ず API を使用してください。

また、LCD を OFF にすると、サウンドアンプへの電源供給も止まりますので、スピーカーから音が出なくなります。ただし、LCD ON 状態においてヘッドフォンを挿入後 LCD OFF 状態にした場合は、ヘッドフォンから音を出すことができます。なお、NITRO と異なり、TWL では LCD OFF 時にヘッドフォンを接続した場合でも、ヘッドフォンから音が出ることが保証されています。

11.2.4 マイク

マイクを使用する場合はプログラマブルゲインアンプ（PGA）への電源供給を ON にする必要があります。

PGA への電源供給はパワーマネジメントの API を使用して制御することができます。

注意事項

マイクへの電源供給を ON にしてから 3sec はマイクを使わないようにしてください。

11.2.5 システム

パワーマネジメントの API を使用して、TWL のシステムリセットまたはパワーを OFF にする（シャットダウン）ことができます。

注意事項

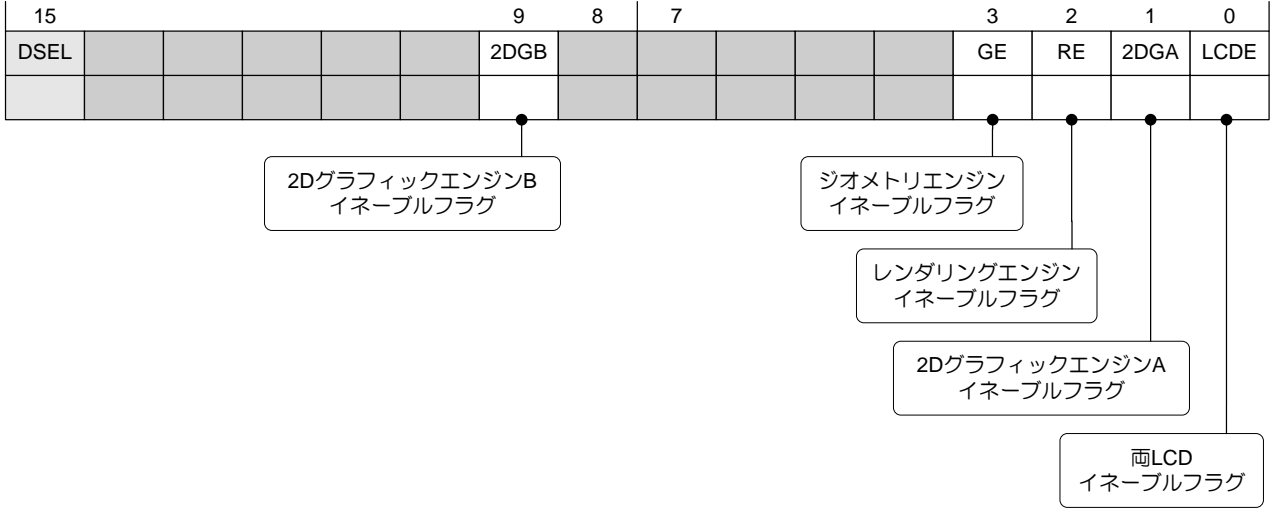
LCD OFF の状態でシャットダウンした場合、本体によっては確実にシャットダウンすることが保証されません（ごく稀にシステムが再起動することがあります）ので、必ず LCD ON の状態でシャットダウンするようにしてください。

11.2.6 グラフィックス

ジオメトリエンジン、レンダリングエンジン、2D グラフィックスエンジンの各回路へのクロック供給を制御し、使用しない回路をディセーブルにすることによって消費電力を抑えることができます。

グラフィックスパワーコントロールレジスタ

名称 POWCNT アドレス 0x04000304 属性 R/W 初期値 0x0000



- 2DGB[d09] : 2D グラフィックスエンジン B
イネーブルフラグ
2D グラフィックスエンジン B を使用しない場合、消費電力を抑えることができます。

0	ディセーブル
1	イネーブル

- GE[d03] : ジオメトリエンジンイネーブルフラグ
レンダリングエンジンをイネーブルにした場合は、SwapBuffers コマンドを 1 回発行してください。

0	ディセーブル
1	イネーブル

- RE[d02] : レンダリングエンジンイネーブルフラグ
レンダリングエンジンをイネーブルにした場合は、SwapBuffers コマンドを 1 回発行してください。

0	ディセーブル
1	イネーブル

- 2DGA[d01] : 2D グラフィックスエンジン A
イネーブルフラグ
3D グラフィックスのみ使用する場合などにおいて、消費電力を抑えることができます。

0	ディセーブル
1	イネーブル

- LCDE[d00] : 両 LCD イネーブルフラグ **<使用禁止>**
ディセーブルにすると、メイン LCD およびサブ LCD コントローラへのクロック供給が止まると共に、メイン LCD およびサブ LCD への電源供給も止まります。

0	ディセーブル
1	イネーブル

注意事項
LCD のイネーブル／ディセーブルの切替は必ず API を使用してください。また、他のビットにデータをライトする場合は、このビットが変化しないよう注意してください。

各フラグのディセーブルによってクロック供給が停止されるメモリやレジスタ

- 2D グラフィックエンジン A ディセーブル時
 - 0x04000008～0x0400004D
 - 0x04000050～0x04000055
 - 2D グラフィックエンジン A 側の OAM とパレット RAM
- 2D グラフィックエンジン B ディセーブル時
 - 0x04001008～0x0400104D
 - 0x04001050～0x04001055
 - 2D グラフィックエンジン B 側の OAM とパレット RAM
- ジオメトリエンジンディセーブル時
 - 0x04000400～0x04000473
 - 0x04000480～0x040004AF
 - 0x040004C0～0x040004D3
 - 0x04000500～0x04000507
 - 0x04000540～0x04000543
 - 0x04000580～0x04000583
 - 0x040005C0～0x040005CB
 - 0x04000600～0x04000607
 - 0x04000610～0x04000611
 - 0x04000620～0x04000635
 - 0x04000640～0x040006A3
- レンダリングエンジンディセーブル時
 - 0x04000320～0x04000321
 - 0x04000330～0x04000341
 - 0x04000350～0x0400035D
 - 0x04000360～0x040003BF

クロック供給を停止したメモリやレジスタへアクセスした場合、表 11-2 の挙動となります。

	ライト	リード
メモリ	無効	ALL ゼロ
レジスタ	無効	リード可能

表 11-2 クロック供給停止時のメモリ、レジスタへのアクセス

11.3 パワーステータス

11.3.1 バッテリー低下状態とバッテリー残量

バッテリー残量が 10～20%を切り、バッテリー低下状態になるとパワーLED が赤色に変わります。さらにバッテリー残量が減り、残り 1%程度になると赤色の低速点滅（明滅）に変わります。アプリケーション側でこれを制御する事は出来ません。

パワーマネジメントの API を使用して表 11-3 のバッテリー状態データをリードし、バッテリー低下状態または 6 段階のバッテリー残量を直接チェックすることができます。なお、パワーLED が赤色または赤色の低速点滅（明滅）に変わる時のバッテリー残量は本体・電池のばらつき、動いているアプリケーション、周囲の温度によって異なりますのであくまで目安として使用してください。

データ種別	データ内容
バッテリー低下状態	バッテリー低下状態フラグ（0～1）
バッテリー残量	バッテリー残量（0～5）

表 11-3 バッテリー状態データ

PMIC 状態データの詳細を以下に示します。

- バッテリー低下状態フラグ
 - 0：バッテリー残量あり
 - 1：バッテリー残量低下
- バッテリー残量
 - 0：バッテリー残量 0% (Empty)
 - 1：バッテリー残量 0～1% (Low)
 - 2：バッテリー残量 1～10%
 - 3：バッテリー残量 10～30%
 - 4：バッテリー残量 30～60%
 - 5：バッテリー残量 60～100%

また、バッテリー残量が 2 から 1 に、または 1 から 0 に変化した時にコールバック関数を呼び出すようにすることができます。これを利用することで、バッテリー残量をポーリングすることなくバッテリー残量の低下を検知できます。

注意事項

NAND フラッシュメモリ、SD メモリカードにアクセスするアプリケーションでは NAND フラッシュメモリ、SD メモリカードのファイル管理領域を保護する目的で、バッテリー残量に少し余裕がある状態で自動シャットダウンを行います。そのため、バッテリーを使い切ることはできません。

ゲームカードやカード上のバックアップメモリにしかアクセスしないアプリケーションについては、バッテリー残量低下による自動シャットダウン処理は行わず、NITRO 同様、バッテリーを使い切るまで遊べます。この場合、バッテリー残量 0 を検知することはできず、対応するコールバック関数も呼び出されません。さらに、バッテリー残量が 2 の状態で急激にバックライト輝度を変更するような場合に、いきなり電源が切れることがあります。

11.3.2 TWL 本体の開閉状態

パワーマネジメントの API を使用して表 11-4 の本体開閉状態データをリードし、TWL 本体の開閉状態をチェックすることができます。

データ種別	データ内容
本体開閉状態	本体開閉状態フラグ（0～1）

表 11-4 本体開閉状態データ

本体開閉状態データの詳細を以下に示します。

- 本体開閉状態フラグ
 - 0：TWL 本体は開かれています
 - 1：TWL 本体は閉じられています

12. アクセラレータ

TWLは除算および平方根演算アクセラレータを備えています。除算器の回路修正以外は、NITROから変更はありません。

12.1 除算器（ディバイダ）

除算器のディバイダデータの設定レジスタ、除算モード指定および除算器の状態を表すコントロールレジスタを以下に示します。

ディバイダデータ（被除数、除数、商、剰余）レジスタ

名称 DIV_NUMER	アドレス 0x04000290	属性 R/W	初期値 0x00000000_00000000	備考 被除数
名称 DIV_DENOM	アドレス 0x04000298	属性 R/W	初期値 0x00000000_00000000	備考 除数
名称 DIV_RESULT	アドレス 0x040002A0	属性 R/W	初期値 0x00000000_00000000	備考 商
名称 DIVREM_RESULT	アドレス 0x040002A8	属性 R/W	初期値 0x00000000_00000000	備考 剰余

31	24	23	16	15	8	7	0
ディバイダデータ下位ワード							

31	24	23	16	15	8	7	0
ディバイダデータ上位ワード							

符号付き整数（符号+整数部 63ビット）

ディバイダコントロールレジスタ

名称 DIVCNT アドレス 0x04000280 属性 R/W 初期値 0x0000

15	14	8	7	1	0
BUSY	DIV0				MODE
ビジー	ゼロ除算				除算モード

● BUSY[d15]：ビジーフラグ

0	ディバイダレディ
1	ディバイダビジー

● DIV0[d14]：ゼロ除算エラーフラグ

0	ゼロ除算エラーなし
1	ゼロ除算エラーあり

● MODE[d01~d00]：除算モード

00	32ビット(DIV_NUMER)/32ビット(DIV_DENOM)→ 商 32ビット(DIV_RESULT), 余 32ビット(DIVREM_RESULT)
01	64ビット(DIV_NUMER)/32ビット(DIV_DENOM)→ 商 64ビット(DIV_RESULT), 余 32ビット(DIVREM_RESULT)
10	64ビット(DIV_NUMER)/64ビット(DIV_DENOM)→ 商 64ビット(DIV_RESULT), 余 64ビット(DIVREM_RESULT)
11	設定禁止

TWLモードではシステム設定の SCFG_EXT レジスタにより、除算器の回路修正の有効/無効を指定することができます。有効にした場合は、以下の注意事項を考慮する必要はありません（NITRO 互換モードではこの問題は回避されません）。

注意事項

除算モードに関わらず、除数（DIV_DENOM）の 64 ビットすべてが 0 であるときにのみ、ゼロ除算エラーフラグが立ちます。

このため除算モードが 32 ビット/32 ビットや 64 ビット/32 ビットである場合においても、除数（DIV_DENOM）の上位 32 ビットを 0 に設定してください。

除数（DIV_DENOM）の上位 32 ビットを 0 に設定しなければ、ゼロ除算エラーフラグが正しく機能しません。

12.1.1 計算サイクル数

ディバイダデータレジスタへの書き込み後、除算モードによって表 12-1 に示すサイクルの間は DIVCNT レジスタのビジーフラグがセットされた状態となります。
ビジーフラグがクリアされた状態で結果が格納されたレジスタを読み出すことで、計算結果を得ることができます。

除算 モード	計算ビット数	計算サイクル数
00	32 ビット(DIV_NUMER)/32 ビット(DIV_DENOM)→ 商 32 ビット(DIV_RESULT), 余 32 ビット(DIVREM_RESULT)	18 サイクル
01	64 ビット(DIV_NUMER)/32 ビット(DIV_DENOM)→ 商 64 ビット(DIV_RESULT), 余 32 ビット(DIVREM_RESULT)	34 サイクル
10	64 ビット(DIV_NUMER)/64 ビット(DIV_DENOM)→ 商 64 ビット(DIV_RESULT), 余 64 ビット(DIVREM_RESULT)	34 サイクル

表 12-1 除算モード別 計算ビット数と計算サイクル数

12.2 平方根演算器

平方根演算器のデータレジスタ、演算結果レジスタ、演算モード指定および平方根演算器の状態を表すコントロールレジスタを以下に示します。

SQRT データレジスタ

名称 SQRT_PARAM アドレス 0x040002B8 属性 R/W 初期値 0x00000000_00000000

31	24	23	16	15	8	7	0
SQRT データ下位ワード							

31	24	23	16	15	8	7	0
SQRT データ上位ワード							

符号なし整数（整数部 64 ビット）

SQRT 演算結果レジスタ

名称 SQRT_RESULT アドレス 0x040002B4 属性 R/W 初期値 0x00000000

31	24	23	16	15	8	7	0
SQRT 演算結果データ							

SQRT コントロールレジスタ

名称 SQRTCNT アドレス 0x040002B0 属性 R/W 初期値 0x0000

15	8	7	0
BUSY			MODE
ビジー			モード

● BUSY[d15]：ビジーフラグ

0	平方根演算器レディ
1	平方根演算器ビジー

● MODE[d00]：SQRT 演算モード

0	入力 32 ビット
1	入力 64 ビット

12.2.1 計算サイクル数

データレジスタへの書き込み後、演算モードによって表 12-2 に示すサイクルの間は SQRTCNT レジスタのビジーフラグがセットされた状態となります。
ビジーフラグがクリアされた状態で結果が格納されたレジスタを読み出すことで、計算結果を得ることができます。

SQRT 演算モード	入力ビット数	計算サイクル数
0	入力 32 ビット	13 サイクル
1	入力 64 ビット	13 サイクル

表 12-2 演算モード別 入力ビット数と計算サイクル数

13. キー

TWL は A、B、L、R、十字、START、SELECT、X、Y のデジタルキーを持ちます。

13.1 キー入力

キーのうち、A、B、L、R、十字、START、SELECT については、キー入力レジスタ（KEYINPUT）を読み込めば、各ビットの状態からキーの入力状態を確認することができます。

X、Y の各キーについては、サブプロセッサに接続してあるため API を利用してキーの入力状態を確認する必要があります。

API を利用することにより、サブプロセッサの動作について特に留意することなくアプリケーションですべてのキーデータをリードすることができます。

キー入力レジスタ

名称 KEYINPUT アドレス 0x04000130 属性 R 初期値 0x0000

15						9	8	7							0
						L	R	DOWN	UP	LEFT	RIGHT	START	SEL	B	A
						キー入力									

● [d09～d00]：キー入力

0	キーが押されている
1	キーが押されていない

注意事項

ユーザーがキーを 1 回押した場合でも、瞬時的に ON-OFF を複数回繰り返す場合があります。ボタンの 2 度押し（チャタリング）を防ぐため、キーの読み込みは間隔を空けてください（1 フレームに 1 回程度）。

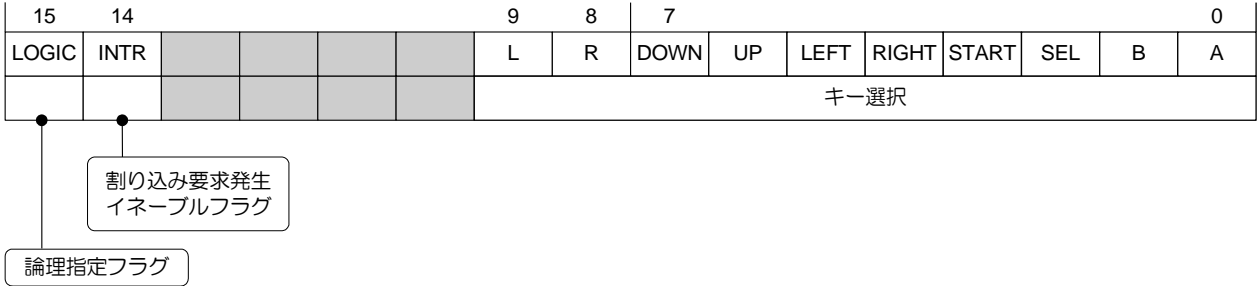
X、Y の各キーについては、キー入力状態を直接レジスタから読み込むことができません。

13.2 キー入力への割り込み処理

キーのうち、A、B、L、R、十字、START、SELECT については、キー入力によって割り込みを発生させることができます。
キーコントロールレジスタ（KEYCNT）で割り込みの対象とするキーの組み合わせや条件を設定することができます。

キーコントロールレジスタ

名称 KEYCNT アドレス 0x04000132 属性 R/W 初期値 0x0000



- LOGIC[d15]：論理指定フラグ

0	指定されたキーのどれか一つでも押されたことを検出する
1	指定されたキーのすべてが押されたことを検出する
- INTR[d14]：割り込み要求発生イネーブルフラグ

0	ディセーブル
1	イネーブル

- [d09～d00]：キー選択

0	キーを指定しない
1	キーを指定する

注意事項

X、Y の各キーについては、キー入力に対して割り込み処理を設定することができません。

14. サウンド

TWL には NITRO と同様に、16 チャンネルの同時発音処理が可能なサウンド回路、特定チャンネルまたはミキサーからの出力をメモリに書き出すことのできる 2 つのサウンドキャプチャ、サウンドを出力する左右のスピーカーおよびヘッドホン出力端子が搭載されています。それに加えて、DSP で生成された DSP サウンドとのミキシング機能、カメラ撮影時の強制シャッター音のためのレジスタボリューム機能を有しています。

TWL ではミキサーの仕様変更となり、PWM 形式の出力から I2S 形式になりました。また、NITRO ではスピーカーおよびヘッドホンへの出力は PMIC で行われていましたが、CODEC と呼ばれるモジュールに変更されています。いくつかのモジュールに変更はありましたが、NITRO と同様にサブプロセッサでシーケンス処理や発音処理を行いますので、サウンド処理を行ってもメインプロセッサに大きな負担をかけることはありません。

サウンド回路の概要図を図 14-1 に示します。

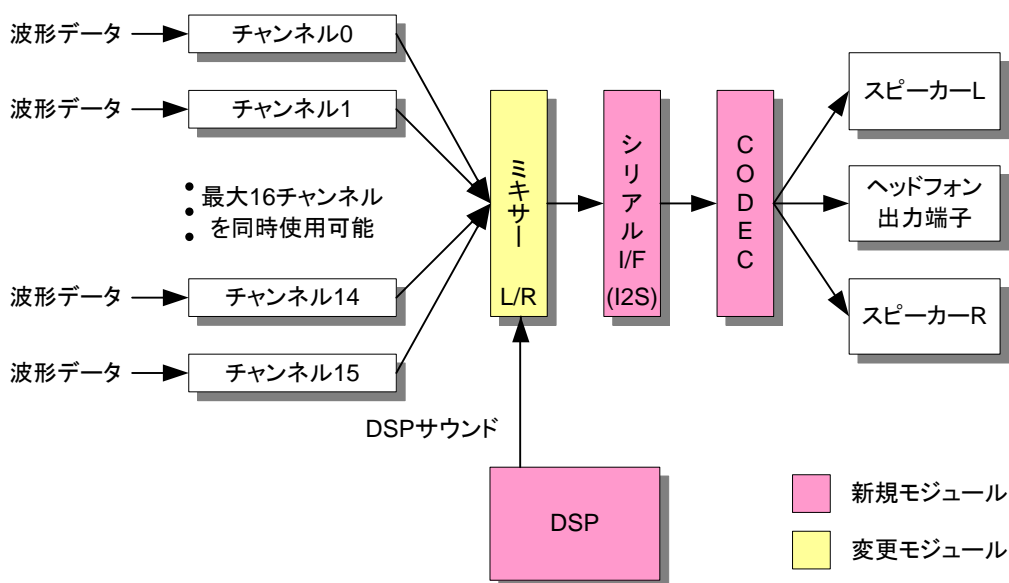


図 14-1 サウンド回路概要図

14.1 ハードウェアスペック

搭載されているサウンド回路のハードウェア仕様は以下の通りです。

14.1.1 データ形式

波形データのフォーマットとして、8bitPCM、16bitPCM、IMA-ADPCM、PSG 矩形波、ノイズを使用することができます。

8bitPCM、16bitPCM、IMA-ADPCM のデータフォーマットと、PSG 矩形波、ノイズについて以下に示します。

14.1.1.1 8bitPCM のデータフォーマット

8bitPCM のデータフォーマットを以下に示します。

8bitPCM データフォーマット

31	24	23	16	15	8	7	0
データ 3		データ 2		データ 1		データ 0	

14.1.1.2 16bitPCM のデータフォーマット

16bitPCM のデータフォーマットを以下に示します。

16bitPCM データフォーマット

31	24	23	16	15	8	7	0
データ 1				データ 0			

14.1.1.3 IMA-ADPCM のデータフォーマット

IMA-ADPCM のヘッダおよびデータ部のデータフォーマットを以下に示します。

IMA-ADPCM ヘッダ (先頭 32 ビット) フォーマット

31							24	23	22							16							15	8							7	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			

IMA-ADPCM データ (33 ビット目以降) フォーマット

31							24							23							16							15							8							7							0						
データ 7							データ 6							データ 5							データ 4							データ 3							データ 2							データ 1							データ 0						

注意事項

ADPCM をリピート機能で繰り返し演奏する場合は、ヘッダ部ではなく必ずデータ部のアドレスに対して繰り返しポイントを設定してください。

また、ADPCM では再生開始後に繰り返しポイントを変更した場合、正常に繰り返し演奏が行われません。繰り返しポイントを変更する場合は、必ず演奏を止めてから行うようにしてください。

14.1.1.4 PSG 矩形波

PSG とはプログラマブルサウンドジェネレータ（Programmable Sound Generator）の略で、出力する矩形波（方形波）の周波数やデューティ比を変化させることで音色を作り出しています。

TWL で使用できる PSG 矩形波はデューティ比を表 14-1 のように変化させることができます。

デューティ比	波形
12.5%	
25.0%	
37.5%	
50.0%	
62.5%	
75.0%	
87.5%	

表 14-1 デューティ比と PSG 矩形波の波形

14.1.1.5 ノイズ

ノイズは設定項目がありません。

ノイズに設定されたチャンネルではホワイトノイズを発生させることができます。

14.1.2 チャンネル

16 個のチャンネルから波形データを同時に再生することが可能です。
 ただし、PSG 矩形波は 16 個のチャンネルのうち、特定の 6 個のチャンネルでのみ再生可能です。
 また、ノイズは 16 個のチャンネルのうち、特定の 2 個のチャンネルでのみ再生可能です。
 チャンネル毎に、音量、音程、パン（定位）が設定可能です。

データ形式毎の再生可能チャンネルを表 14-2 に示します。

データ形式	再生可能チャンネル
8bitPCM	チャンネル 0～15 のすべてのチャンネルで再生可能です
16bitPCM	
ADPCM	
PSG 矩形波	チャンネル 8～13 のチャンネルで再生可能です PSG 矩形波を再生させているチャンネルでは、8bitPCM、16bitPCM、ADPCM を同時に再生することはできません
ノイズ	チャンネル 14～15 のチャンネルで再生可能です ノイズを再生させているチャンネルでは、8bitPCM、16bitPCM、ADPCM を同時に再生することはできません

表 14-2 データ形式と再生可能チャンネルの一覧

14.1.3 ミキサー

NITRO ではミキサーの出力がパルス幅変調（Pulse Width Modulation）でしたが、TWL では I2S へと変更されています。
 I2S は、デバイス間でデジタル音声データを受け渡すために用いられる標準的なインタフェースです。TWL では CPU-CODEC 間が I2S バスで接続されており、ステレオ音声データやマイク入力音声データの受け渡しを行います。TWL においてはデータ長が 16bit 固定、サンプリングレートを 32.73kHz、47.61kHz から選択することができます。

14.1.3.1 DSP サウンドとの合成

TWL では、サウンド回路から出力される NITRO サウンドと DSP で生成された DSP サウンドを、表 14-3 にあるように 9 パターンの比率から選択して合成することができます。

設定パターン	NITRO サウンド比率	DSP サウンド比率	備考
0	8 / 8	0 / 8	NITRO サウンドのみ
1	7 / 8	1 / 8	
2	6 / 8	2 / 8	
3	5 / 8	3 / 8	
4	4 / 8	4 / 8	
5	3 / 8	5 / 8	
6	2 / 8	6 / 8	
7	1 / 8	7 / 8	
8	0 / 8	8 / 8	DSP サウンドのみ

表 14-3 DSP サウンドとの合成比率

14.1.3.2 レジスタボリューム

TWL は、ボリュームを変更することのできる通常のアナログサウンドボリュームとは別に、カメラ撮影時のシャッター音などのためにアナログサウンドボリュームを無視して、プログラムから任意のボリュームでサウンドを再生するレジスタボリューム機能を有しています。

14.1.3.3 CODEC

TWL の CPU に接続された CODEC と呼ばれるモジュールの役割は、アナログ信号とデジタル信号の変換です。ミキサーから出力されたデジタル信号である I2S 形式のサウンドをアナログに変換してスピーカーやヘッドフォンに出力するほかにも、マイクやタッチパネルからのアナログ入力をデジタルに変換しています。

CODEC には 2 つの動作モードが用意されています。1 つは、NITRO 互換用に用意された CODEC-DS モード。もう 1 つは、より改善された CODEC-TWL モードです。アプリケーションはどちらかの動作モードを選択しなければなりません。

CODEC の動作モードは ROM 内登録情報に基づいて IPL が設定します。アプリケーションの起動後に動作モードを変更することはできません。

前述のレジスタボリューム機能は CODEC-TWL モードでのみサポートされています。アプリケーションで強制的にシャッター音を鳴らす必要がある場合は、CODEC-TWL モードを選択してください。

CODEC-DS モードと CODEC-TWL モードの違いをまとめると、表 14-4 のようになります。

カテゴリ	項目	CODEC-DS モード	CODEC-TWL モード
マイクサンプリング関連	データ長（振幅分解能）	8 ビットまたは 12 ビット	16 ビット
	サンプリングレート	可変	47.61 kHz / 32.73 kHz 23.81 kHz / 16.36 kHz 15.87 kHz / 10.91 kHz 11.90 kHz / 8.18 kHz
	サブプロセッサの処理負荷	比較的重い（サンプリングレートに比例）	比較的軽い
	ゲイン設定	26, 32, 38, 44 dB	10.5~70.0 dB（0.5 dB 単位）
その他	レジスタボリューム機能	使用不可	使用可能

表 14-4 CODEC-DS モードと CODEC-TWL モードの比較表

14.1.4 マスターボリューム

マスターボリュームによってスピーカーからの出力を 0~127 の 128 段階で調整することができます。

14.1.5 サウンドキャプチャ

TWL には出力波形データをメモリに書き出せる 2 機のサウンドキャプチャが搭載されています。サウンドキャプチャ 0 ではミキサー L の出力またはチャンネル 0 の出力をメモリに書き出すことができます。サウンドキャプチャ 1 ではミキサー R の出力またはチャンネル 2 の出力をメモリに書き出すことができます。

サンプリング周波数は 1.04876MHz まで指定できます。また、振幅分解能は 8 ビットまたは 16 ビットからの指定となります。

14.1.6 注意事項

サンプリングレートの高い波形データの再生や、早回し再生によるオリジナルデータよりも高い音程での再生は、より頻繁な DMA 転送を引き起こします。

DMA 転送が頻繁に行われると、ワイヤレス通信、マイクなどのサブプロセッサ処理や、メインプロセッサの処理にも影響を与えることがあります。

14.2 サウンドブロック図

TWL に搭載されているサウンド回路のブロック図を示します。

14.2.1 サウンド全体

サウンド回路全体のブロック図を図 14-2 に示します。

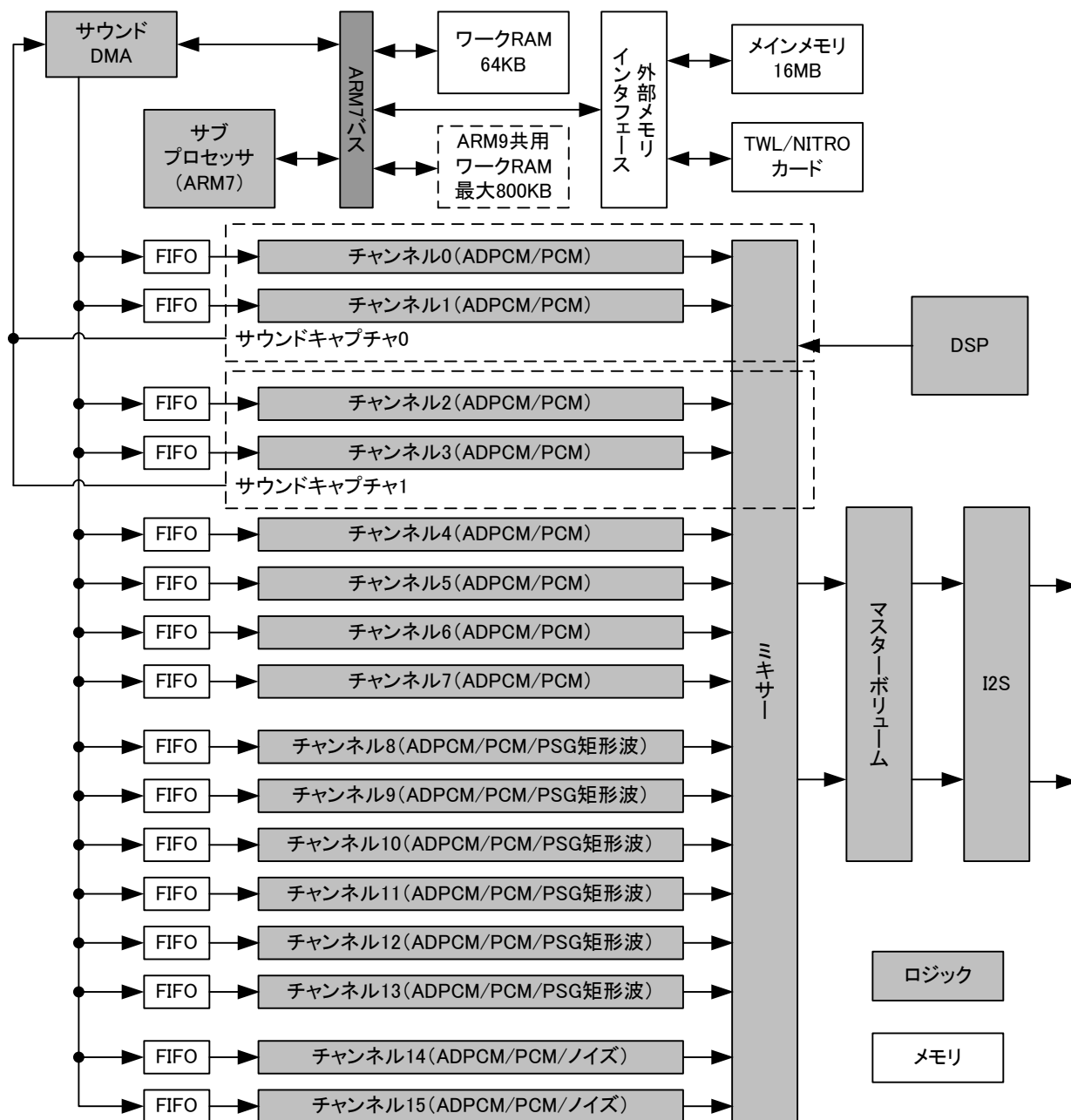


図 14-2 サウンドブロック全体図

14.2.2 チャンネル 0～3、サウンドキャプチャ 0～1

チャンネル 0～3 およびサウンドキャプチャ 0～1 のブロック図を図 14-3 に示します。

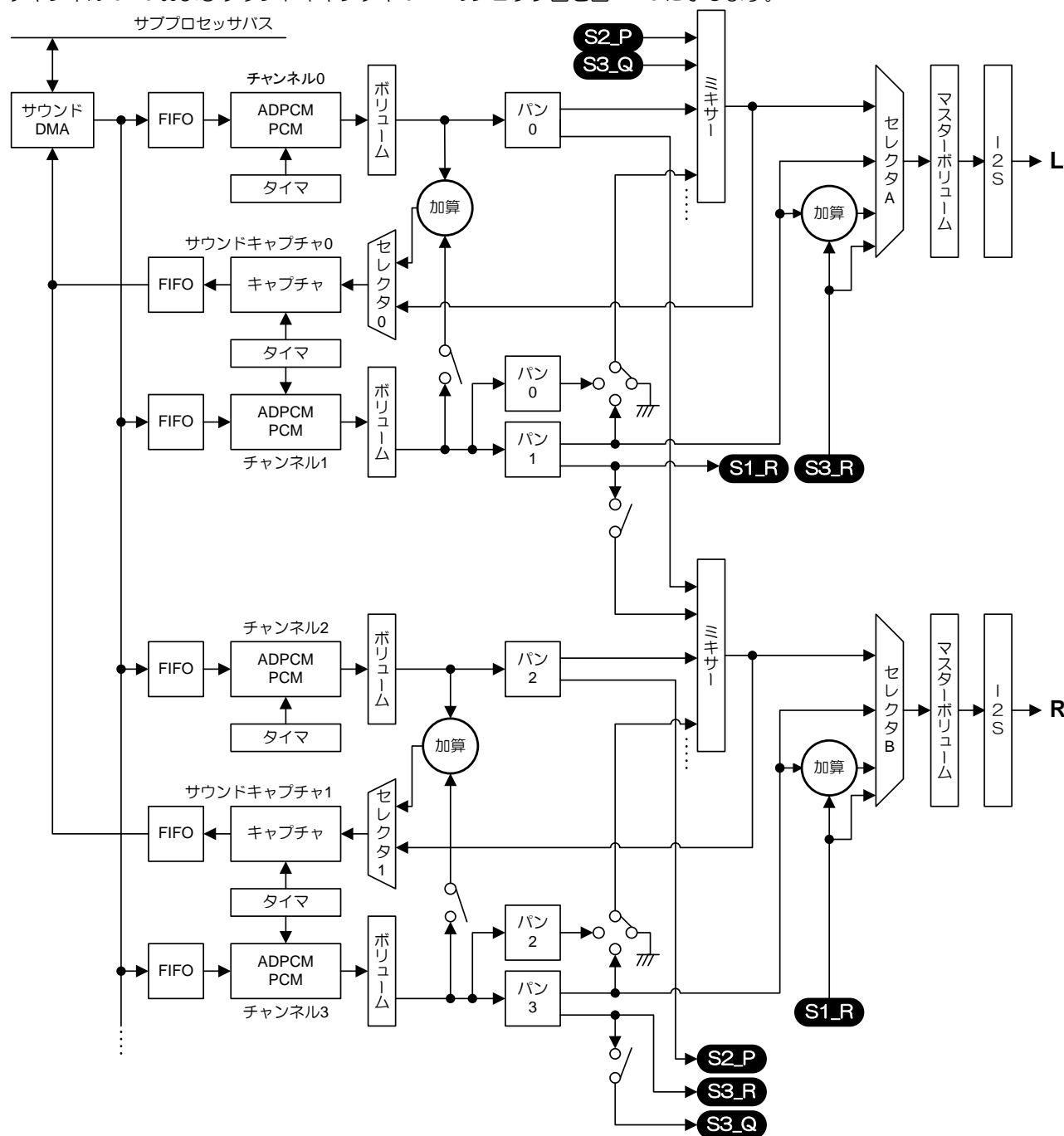


図 14-3 チャンネル 0～3 およびサウンドキャプチャ 0～1 のブロック図

注意事項 1

チャンネル 1 および 3 のパンブロック前後のスイッチ部分は常に最終段出力選択のセクタ A および B まで接続している回路となっていたため、最終段出力としてチャンネル 1 または 3 を選択した場合、サウンドが出力してしまいます。サウンドを出力したくない場合はこの点に注意してください。

注意事項 2

チャンネル 1 および 3 のパンブロックからミキサーへの入力信号が表 14-5 に示す優先順位で決まるため、チャンネル 1 および 3 を最終段出力へバイパスする設定にした場合、加算されたチャンネルをミキサーへ入力してそのミキサー出力をキャプチャする設定としていても、実際にはチャンネル 1 および 3 はミキサーへ入力されず（パンブロック直後のスイッチが GND へ接続され、この場合のみミキサーの入力は常に 0 となります）、結果としてリバーブが掛かりません。チャンネル 1 および 3 を最終段出力へバイパスする設定とした場合、ミキサーと加算器を併用したリバーブはできませんので注意してください。（下記の注意事項 3 を考慮するとミキサーのみ使用することを推奨します）。

優先度	ミキサーへの入力	スイッチの状態
高 ↑↓ 低	0 入力（GND へ接続）	サウンド最終段出力バイパス設定
	パン 0 ならびにパン 2	チャンネル加算&キャプチャ設定
	パン 1 ならびにパン 3	通常設定

表 14-5 チャンネル 1 ならびに 3 からミキサーへの入力スイッチの優先順位

注意事項 3

TWL には TWL モードと NITRO 互換モードがあります。NITRO 互換モードでは修正が適用されないため、以下の不具合が発生します。

チャンネル加算器の出力をキャプチャするロジック（セレクト 0 および 1 の前段）に不具合があり、以下の 2 つの症状が発生します。

- チャンネル 0 と 1 ならびにチャンネル 2 と 3 を加算する場合、それぞれの加算結果がオーバーフローを起こすと、符号反転したデータが出力される。
- チャンネル 0 と 1 ならびにチャンネル 2 と 3 を加算しない場合、チャンネル 0 と 1 ならびにチャンネル 2 と 3 の符号が共に負となると、キャプチャデータが強制的に負の最大値へ変更される。

これらの結果としてサウンドにノイズが出力されます。この不具合の対策としましては、加算器を使用する場合は加算データが飽和しないようにし、加算器を使用しない場合はチャンネル 0 と 1 ならびにチャンネル 2 と 3 のデータが共に負の値にならないようにしてください。

14.2.3 チャンネル 4～7

チャンネル 4～7 のブロック図を図 14-4 に示します。

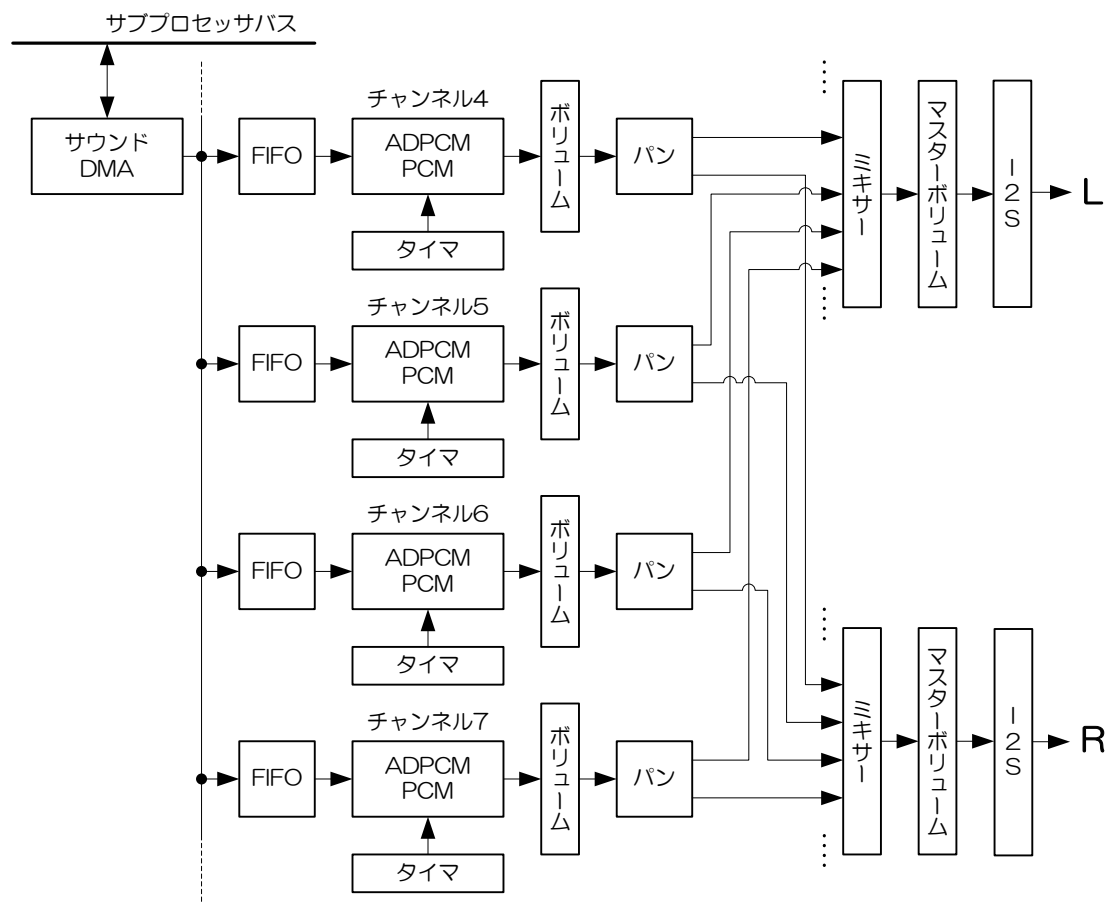


図 14-4 チャンネル 4～7 のブロック図

14.2.4 チャンネル 8～15

チャンネル 8～15 のブロック図を図 14-5 に示します。
 チャンネル 8～13 では PSG 矩形波の再生が可能です。
 チャンネル 14～15 ではノイズの再生が可能です。

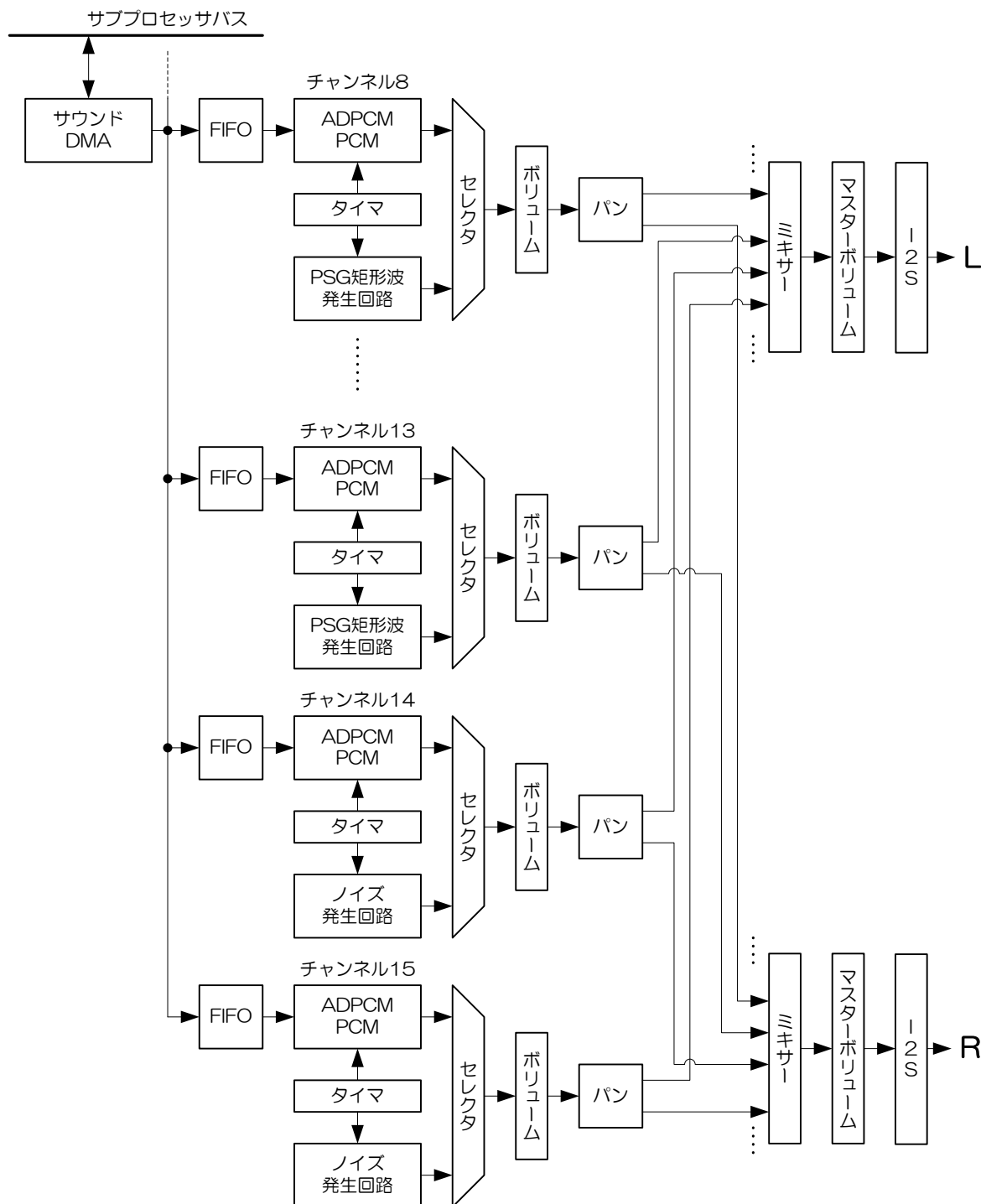


図 14-5 チャンネル 8～15 のブロック図

14.2.5 サウンド使用例

通常時およびサウンドキャプチャ回路を利用したリバーブ効果、エフェクトのサウンド回路使用例を示します。

14.2.5.1 通常使用例

通常使用時は、メインメモリからリードしたサウンド波形データを各チャンネルで再生し、ミキサーを通してスピーカーへと出力されます。

通常使用時のサウンド回路使用例を図 14-6 に示します。なお、図中の赤字で示す数字はブロック入出力時のデータのビット数を表します。

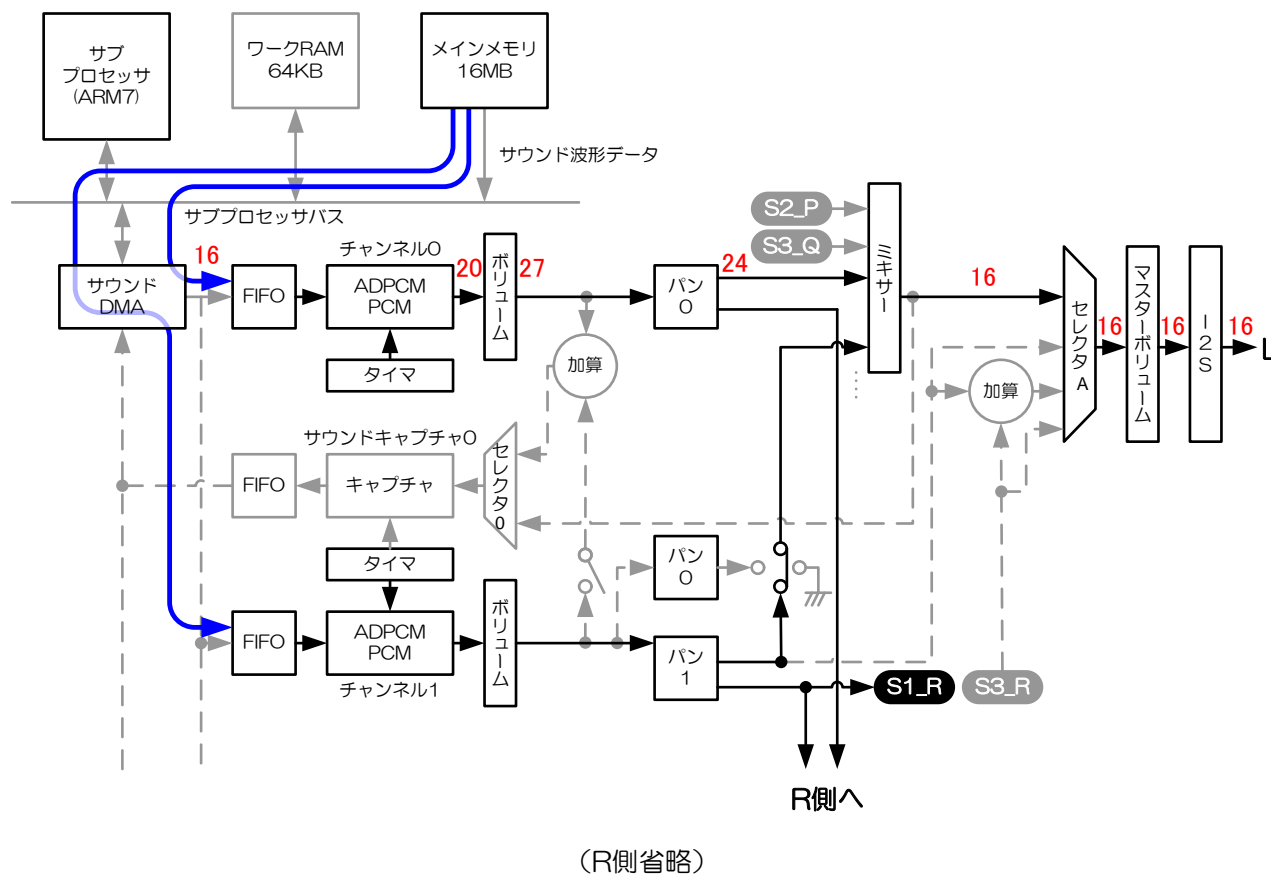


図 14-6 サウンド使用例（通常）

14.2.5.2 リバース例

サウンドキャプチャ機能を使用してリバース（反響）効果を出すことができます。ミキサーからの出力をサウンドキャプチャでワーク RAM に蓄え、蓄えたキャプチャデータをチャンネルで再生し、ミキサーを通してスピーカーへと出力するとリバース効果が出ます。

リバース効果時のサウンド回路使用例を図 14-7 に示します。なお、図中の赤字で示す数字はブロック入出力時のデータのビット数を表します。

図 14-7 の例では、チャンネル 1 をリバース効果用に使用しています。

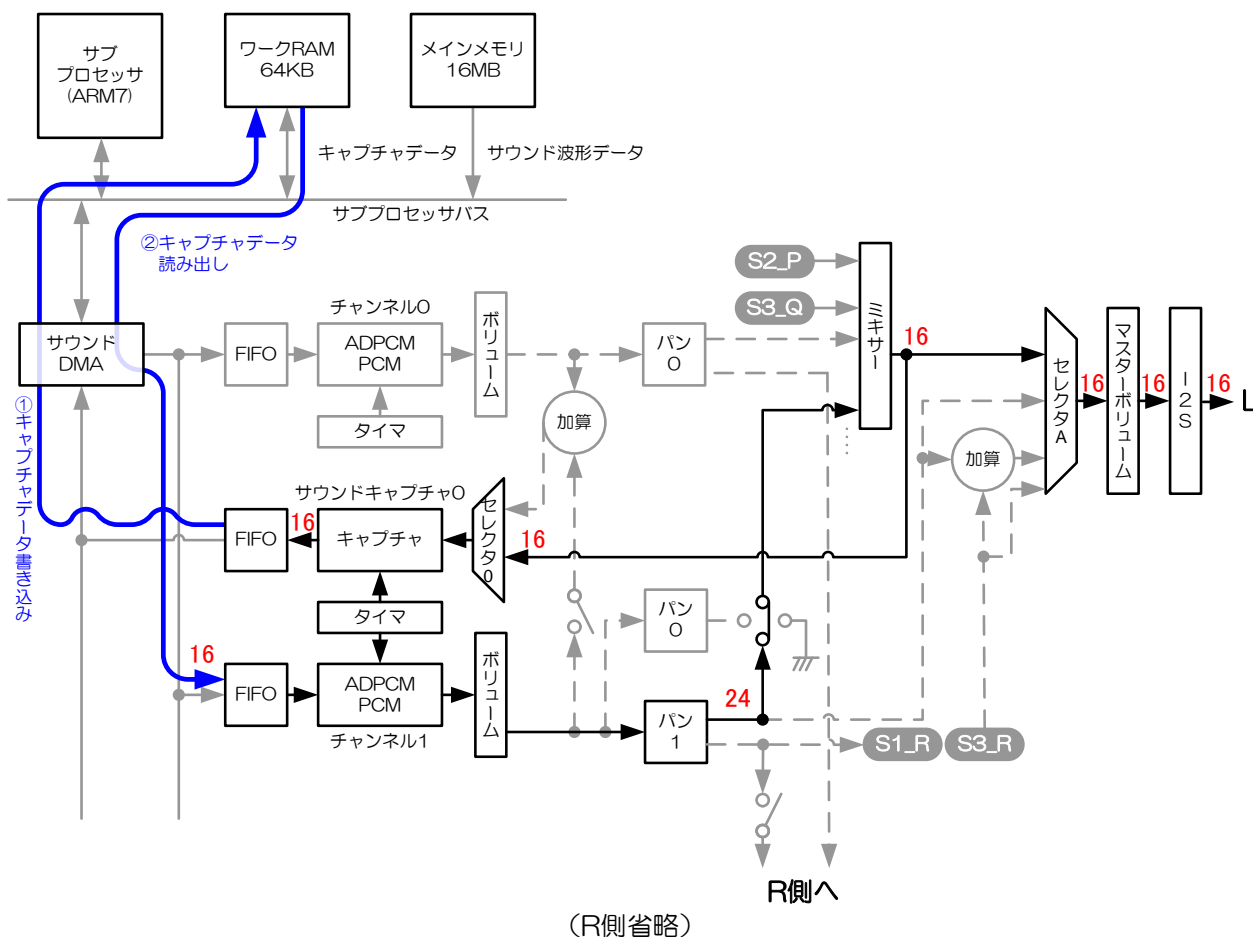


図 14-7 サウンド使用例（リバース）

14.2.5.3 エフェクト例

サウンドキャプチャ機能を使って、サウンド出力前にデータの加工を行うことができます。
キャプチャしたデータをワーク RAM に蓄え、サブプロセッサで加工した後、チャンネルから出力します。

エフェクト使用例を図 14-8 に示します。なお、図中の赤字で示す数字はブロック入出力時のデータのビット数を表します。

図 14-8 の例では、チャンネル 1 をエフェクト用に使用しています。

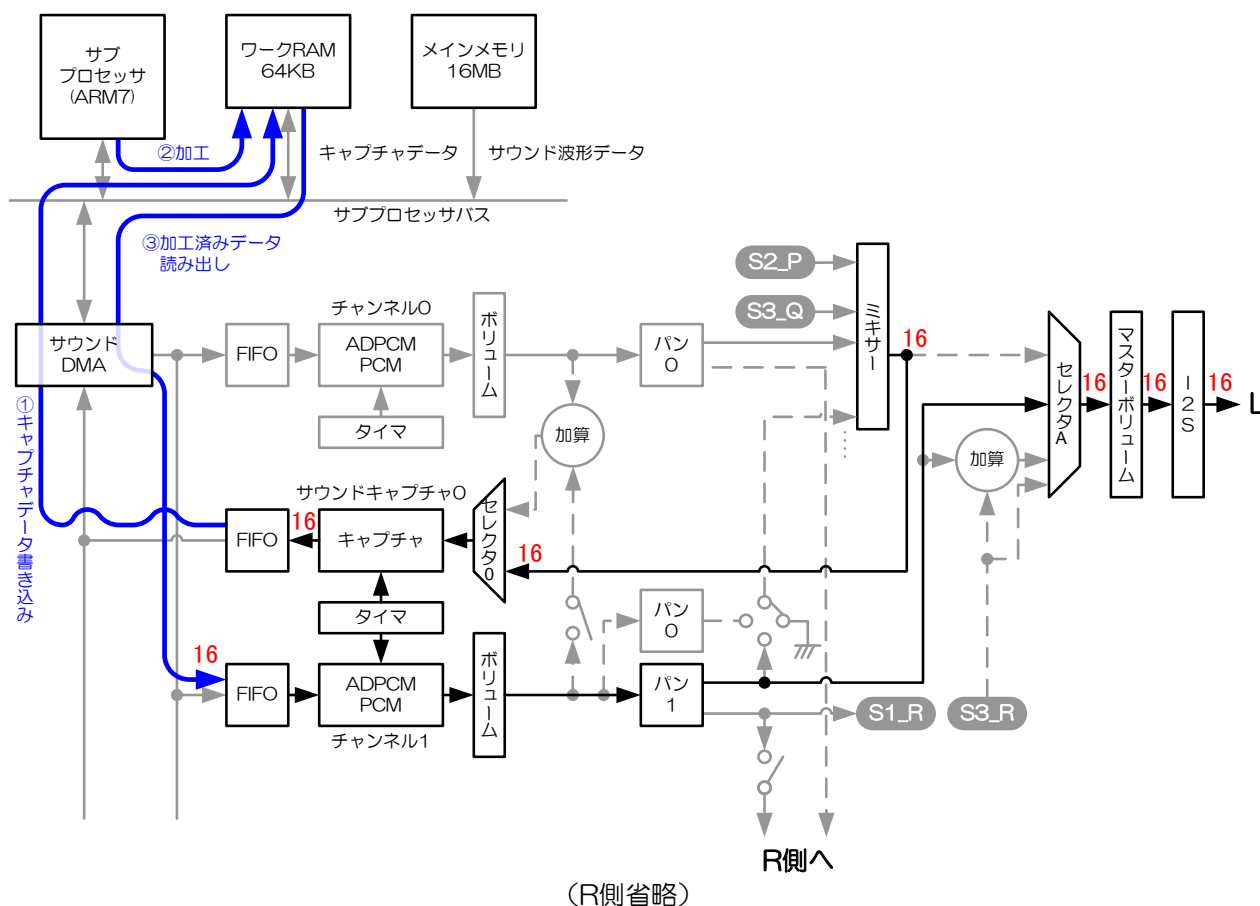


図 14-8 サウンド使用例（エフェクト）

14.3 NITRO-Composer

NITRO-Composer を使うことで、サブプロセッサの動作について特に留意する必要がなくなります。
また、BGM のような複雑な音でも簡単に再生することができます。

14.3.1 NITRO-Composer の再生形式

シーケンス再生、波形再生、ストリーム再生の 3 種類の再生形式を使用することができます。

14.3.1.1 シーケンス再生

BGM や効果音などのように、色々な音を組み合わせた再生を行います。

最大 16 個のシーケンスを同時に再生できます。

例えば、1 シーケンスを割り当てた BGM の再生中に、効果音を 15 個まで同時に再生することができます。

アプリケーション側から、各シーケンスに対して個別に、テンポやボリュームなど様々なパラメータを制御することができます。

14.3.1.2 波形再生

波形データを直接再生する方式です。

マイクでサンプリングした波形データなどを再生することができます。

14.3.1.3 ストリーム再生

ムービーの音楽再生など、長時間の再生を行います。

TWL / NITRO カードに記録されているデータを順次ロードしながら波形データを再生することができます。

14.3.1.4 CODEC 内部での遅延について

サウンド再生時は、CODEC にデータが入力されてからアナログ信号が出力されるまでに、 $21 / F_s$ (秒) の演算遅延が発生します。

マイクサンプリング時は、CODEC にアナログ信号が入力されてからデータとして出力されるまでに、 $17 / F_s$ (秒) の遅延が発生します。

※ F_s はサンプリングレート (32.73kHz / 47.61kHz)

15. ワイヤレス通信

TWL には、2.4GHz 帯を使用して無線通信を行うことができるワイヤレス通信ハードウェアが 2 基搭載されています。1 つは NintendoDS および DS Lite に搭載されているものと同じもの（NITRO 無線）で、もう 1 つは高速通信・高セキュリティに対応したもの（TWL 無線）です。
これら 2 つの無線モジュールを同時に使用することはできません。

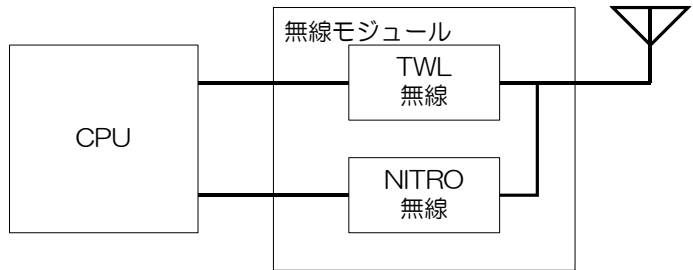


図 15-1 無線モジュール

15.1 ハードウェア仕様

ワイヤレス通信のハードウェア仕様は表 15-1 の通りです。

項目	内容	
	NITRO 無線（NITRO 互換無線）	TWL 無線
使用通信帯	2.4GHz 帯	（左に同じ）
通信プロトコル	IEEE802.11 （インフラストラクチャーモード） 任天堂独自プロトコル （DS ワイヤレスプレイモード） 任天堂独自プロトコル （DS ダウンロードプレイモード）	IEEE802.11b/g （インフラストラクチャーモード）
セキュリティ	WEP40 ビット／128 ビット対応	WEP40 ビット／128 ビット対応 WPA-PSK(TKIP/AES) WPA2-PSK(TKIP/AES)
無線チャンネル数	13ch	国によって異なります 米国・台湾：1～11ch 日本・欧州・豪州：1～13ch
通信速度	1Mbps または 2Mbps	1, 2, 5.5, 11Mbps 6, 9, 12, 18, 24, 36, 48, 54Mbps
通信到達距離	10～30m 周囲の環境や本体の向きによって大きく 変動することがあります。	（左に同じ）
MAC アドレス	個々の TWL で個別に持っていますので、 個々の識別が可能です。	（左に同じ）
干渉機器	2.4GHz 帯を使用している機器（コードレ ス電話、電子レンジ、ゲームボーイアドバ ンス専用ワイヤレスアダプタ、ウェーブバ ード、他の Wi-Fi 機器など）	（左に同じ）

表 15-1 ワイヤレス通信のハードウェア仕様

注意事項

通信をする場合、消費電力が増加するため、バッテリーの消費も早くなります。このため、通信をしない時は、ワイヤレス通信の API を使用して STOP ステートに遷移させてください。
ゲームボーイアドバンス専用ワイヤレスアダプタとの通信はできません。
他の機器との干渉のため通信が困難となる場合があります。そのため、通信パケットサイズをなるべく小さくしてください。

15.2 ワイヤレスマネージャ

ワイヤレスマネージャの API をコールすることによって、サブプロセッサの動作について特に留意することなくワイヤレスシステムをコントロールすることができます。

インフラストラクチャーモード（どちらの無線モジュールを使用するかという選択も可能です）、DS ワイヤレスプレイモード、DS ダウンロードプレイモードを使用することができます。

15.2.1 インフラストラクチャーモード

無線 LAN（IEEE802.11b/g）のアクセスポイントを利用して、インターネットに接続できるモードです。

15.2.2 DS ワイヤレスプレイモード

最大 16 台の本体間でワイヤレス通信が可能です。

NITRO 独自の通信方式を使用していますので、1 フレーム以下でデータの交換が可能となります。

ただし、DS ワイヤレスプレイモードでの最大通信データサイズは 512 バイトとなります。

15.2.3 DS ダウンロードプレイモード

TWL / NITRO カードのない子機が、TWL / NITRO カードのある親機からゲームをダウンロードするモードです。

親機がアドレス等の情報を含んだヘッダ付きのデータを送信すると、子機はシステム ROM がヘッダで指定された領域にデータを格納します。

任天堂独自のプロトコルを使用します。

注意事項

DS ダウンロードプレイでダウンロードされたゲームは、NITRO モードで動作します。

16. タッチパネル

下画面 LCD には抵抗膜方式のタッチパネルが搭載されており、ドット単位で座標を取得することができます。
タッチパネルは指で操作することもできますが、ペン先半径 1.0mm のタッチペン（図 16-1 を参照）が TWL に標準装備されています。

タッチパネルの API を利用することにより、サブプロセッサの動作について特に留意することなくアプリケーションでタッチパネルを使用することができます。
タッチパネルを使用する前に API 側で座標位置データのばらつきを吸収するため、IPL の補正プログラムで内蔵フラッシュメモリに保存されたキャリブレーションデータを読み出して、API でセットする必要があります。

API を使用して、表 16-1 に示すタッチパネル入力データをリードすることができます。
API には、4 回／フレーム間隔でリードする「オートサンプリング」と、リクエストに対してリアルタイムにリードする「リクエストサンプリング」の 2 種類のリード形態があります。
ただし、リクエストサンプリングで発行するリードリクエストは、リードリクエスト毎に 4.17msec 以上の間隔をあげなければ正しい値が読み取れない場合があります。

データ種別	データ内容
タッチパネル入力データ	X 座標（8bit）、Y 座標（8bit）、 接触判定フラグ（1bit）、 データ有効性フラグ（2bit）

表 16-1 タッチパネル入力データ

タッチパネル入力データの詳細を以下に示します。

- X 座標、Y 座標
X 座標：0～255（ドット単位）
Y 座標：0～191（ドット単位）
- 接触判定フラグ
0：タッチパネルに触れていない
1：タッチパネルに触れている
- データ有効性フラグ
00：X 座標、Y 座標ともに有効なデータ
01：X 座標が無効なデータ
10：Y 座標が無効なデータ
11：X 座標、Y 座標ともに無効なデータ

注意事項

抵抗膜方式のタッチパネルは構造上 1 度に検出できる座標は 1 ヶ所のみとなります。
このため複数の場所を同時にタッチした場合、各点の座標を検出することはできません。
なお、位置の検出には付属のタッチペン使用時約 80 g の力でタッチパネルを押す必要があります。
タッチパネルの TWL 本体への組み込み誤差とタッチペンの先端形状の制限によって、画面端 6 ドット以内を触ることができない場合があります。
画面を触った直後、ならびに離れた直後に、異常な座標データが読み込まれる場合があります。そのような場合（特に、画面に表示されたボタンを操作する等）は、同じような座標が連続して読まれたときのみ有効データとして処理する等、アプリケーション側で対処してください。また、接触判定フラグとデータ有効性フラグは独立したデータであるため、無効なデータが格納された場合でもタッチされているという状況があり得ることにご注意ください。例えば、一筆書き中に無効なデータが格納される場合がありますが、その場合でもユーザーが画面からタッチペンを離れたと判断せず、接触判定フラグをリードして判断するようにしてください。

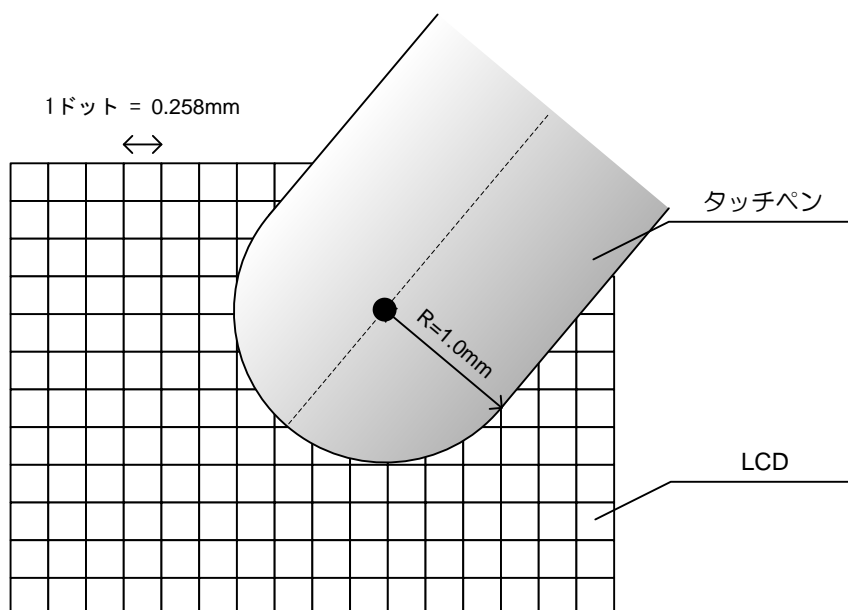


図 16-1 LCD のドットとタッチペンの大きさ対比図

16.1 タッチパネルの構造

抵抗膜方式のタッチパネルの構造は図 16-2 の通りです。

通常時は、透明導電膜（ITO 膜：Indium Tin Oxide 膜）が生膜された上部フィルムと下部フィルムが距離を空けて形成されているため通電することはありません。

指やタッチペンでパネルを押すと、圧力により上部フィルムと下部フィルムが接触し通電する構造となっています。ドットスペーサーは常時 ON 状態や誤入力を防止します。

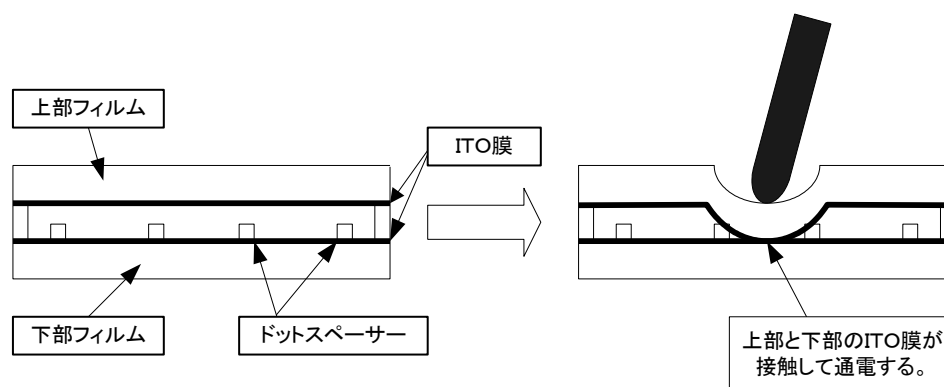


図 16-2 タッチパネルの構造

17. マイク

TWL には音声サンプリング可能な無指向性コンデンサマイクが標準装備されており、マイクからの音声入力を音声サンプリング処理することができます。

また、切り替えにより「ニンテンドーDSシリーズ専用 イヤホンマイク」を使用することができます。

マイクの感度は本体によって最大2倍程度の個体差誤差があります。

マイクの概要図を図 17-1 に示します。

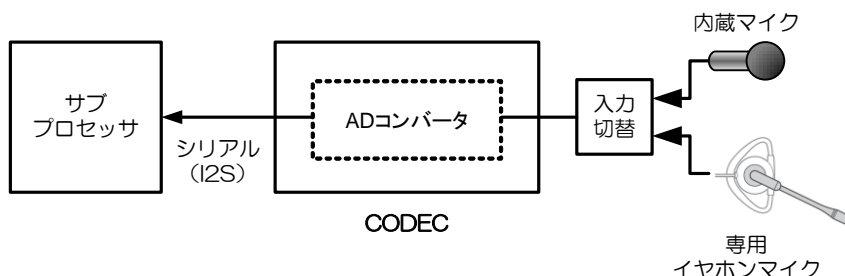


図 17-1 マイク概要図

17.1 NITRO からの変更点

NITRO ではタッチパネル IC で行われていたアナログ・デジタル変換と PMIC で行われていたマイク入力のゲイン調整が、TWL では仕様の変更により CODEC と呼ばれるモジュールで行われるようになります。

CODEC で変換したマイク入力データ (I2S 形式) は NITRO と違い※、データ長が 16 ビット固定、サンプリングレートが 32.73 kHz / 47.61 kHz を基準とした 8 段階となっています。また、ゲイン調整も 10.5dB~70.0 dB の間で 0.5 dB 単位で調整することができるようになります。

※NITRO ではデータ長が 8 ビットまたは 12 ビット、可変サンプリングレート、4 段階のゲイン調整でした。

17.1.1 CODEC の起動モード

CODEC には 2 つの動作モードが用意されています。1 つは、NITRO 互換用に用意された CODEC-DS モード。もう 1 つは、より改善された CODEC-TWL モードです。アプリケーションはどちらかの動作モードを選択しなければなりません。

CODEC の動作モードは ROM 内登録情報に基づいて IPL が設定します。アプリケーションの起動後に動作モードを変更することはできません。

CODEC-TWL モードと CODEC-DS モードの違いについては、「CODEC」にある表 14-4 にまとめられています。

マイクの API を利用することにより、サブプロセッサの動作について特に留意することなくアプリケーションでマイクを使用することができます。

17.1.2 CODEC-TWL モードのマイク

マイクのサンプリングレートを4段階から選択することができ、ミキサーで設定されたサウンドサンプリングレートとの組み合わせでサンプリングレートが変化します。

設定パターン	サウンドサンプリングレート	
	47.61 kHz	32.73 kHz
0	47.61 kHz	32.73 kHz
1	23.81 kHz	16.36 kHz
2	15.87 kHz	10.91 kHz
3	11.90 kHz	8.18 kHz

表 17-1 マイクのサンプリングレート

※マイク API を使用することで、上記表以外のサンプリングレートを使用することもできます。

マイクのゲイン設定には、10.5～70.0 dB の値を 0.5 dB 単位で指定することができます。

17.1.3 CODEC-DS モードのマイク

NITRO でのマイクの仕様に従います。

設定できるサンプリングレートは概算で数 kHz～32kHz となりますが、正常動作できる数値はサブプロセッサの使用状況に依存します。

また、記録時間はアプリケーションで用意するメモリサイズとサンプリングレートに依存します。

マイク入力には V フランクに同期した 60Hz のノイズが重畳しますが、周波数自体が低いこと、ならびに音声入力に対してノイズレベルが十分小さいことから、聞こえる音として問題になることはありません。

注意事項

ワイヤレス通信、サウンドの処理はサブプロセッサを使用しているため、これらと同時にマイクを使用する場合は、サブプロセッサの負荷に合わせて適度なサンプリングレートを設定する必要があります。

タッチパネルデータの読み込み、内蔵フラッシュメモリへのアクセス、PMIC 制御は同じシリアルバスを共有しているため、これらと同時にマイクを使用する場合は、それぞれが競合しないように設定する必要があります。

マイクへの電源供給を ON にしてから 3sec はマイクを使わないようにしてください。

17.1.4 マイクの入力値

マイク入力にはノイズ成分が含まれます。そのため、マイク入力がない状態であっても、表 17-2 および表 17-3 に示す範囲でマイク入力が入ることがあります。

入力がないにも関わらず、マイク入力ありと誤判定してしまうことを防ぐため、表 17-2 および表 17-3 に示す範囲の値でマイク入力ありと判定することは避けてください。また、ゲインが大きい場合、ボタン操作や筐体を擦る摩擦ノイズがマイク入力に顕著に表れます。しきい値の設定には注意してください。

振幅分解能		マイク入力あり 判定禁止領域 (ノイズ成分)	ゲイン	
			倍率	dB
8 ビット	符号付き	±11	20	+26
		±13	40	+32
		±15	80	+38.1
		±20	160	+44.1
	符号なし	117～139	20	+26
		115～141	40	+32
		113～143	80	+38.1
		108～148	160	+44.1
16 ビット	符号付き	±2368	20	+26
		±2880	40	+32
		±3392	80	+38.1
		±4672	160	+44.1
	符号なし	30400～35136	20	+26
		29888～35648	40	+32
		29376～36160	80	+38.1
		28096～37440	160	+44.1

表 17-2 DS 本体のサウンド回路／TWL 本体の CODEC-DS モードの無音時におけるマイク入力値

振幅分解能		マイク入力あり 判定禁止領域 (ノイズ成分)	ゲイン	
			倍率	dB
8 ビット	符号付き	±3	3～20	10.5～26.0
		±5	21～40	26.5～32.0
		±9	42～80	32.5～38.0
		±16	84～160	38.5～44.0
		±32	168～320	44.5～50.5
		±51	355～640	51.0～56.0
		±110	669～1280	56.5～62.0
		±128	1334～2560	62.5～70.0
	符号なし	125～131	3～20	10.5～26.0
		123～133	21～40	26.5～32.0
		119～137	42～80	32.5～38.0
		112～144	84～160	38.5～44.0
		96～160	168～320	44.5～50.5
		77～179	355～640	51.0～56.0
		18～238	669～1280	56.5～62.0
		0～256	1334～2560	62.5～70.0
16 ビット	符号付き	±768	3～20	10.5～26.0
		±1280	21～40	26.5～32.0
		±2304	42～80	32.5～38.0
		±4096	84～160	38.5～44.0
		±8192	168～320	44.5～50.5
		±13056	355～640	51.0～56.0
		±28160	669～1280	56.5～62.0
		±32768	1334～2560	62.5～70.0
	符号なし	32000～33536	3～20	10.5～26.0
		31488～34048	21～40	26.5～32.0
		30464～35072	42～80	32.5～38.0
		28672～36864	84～160	38.5～44.0
		24576～40960	168～320	44.5～50.5
		19712～45824	355～640	51.0～56.0
		4608～60928	669～1280	56.5～62.0
		0～65535	1334～2560	62.5～70.0

表 17-3 TWL 本体の CODEC-TWL モードの無音時におけるマイク入力値

注意事項 1

TWL のマイクの API で有効ビット幅 12 ビットのサンプリングを行った場合、実際には下位 4 ビットが 0 パディングされた 16 ビットデータとして取得されます。そのため表 17-2 および表 17-3 の「マイク入力あり判定禁止領域」は 16 ビットの値を参照してください。

注意事項 2

CODEC-TWL モードでの+44.5dB（168 倍）以上のゲイン設定時にはノイズ成分が顕著に表れますので、振幅レベルによるマイク入力あり／なしの判定は行わないでください。

注意事項 3

TWL 本体で動作するときに CODEC-TWL モードを使う TWL 対応ソフトの場合、DS 本体で動作するときには DS 本体のサウンド回路を用いることになるため、DS 本体での動作時と TWL 本体での動作時でノイズ成分が異なることに注意してください。

また、マイク入力値は振幅分解能のビット数で表せる数値のすべての範囲の値を取るとは限らず、本体によって表 17-4 および表 17-5 に示す範囲の個体差があります。入力保証外範囲に含まれる値を期待するようなマイク入力判定は避けてください。

振幅分解能		入力保証外範囲 下限側	入力保証外範囲 上限側	マイク入力値の保証範囲
8 ビット	符号付き	-128~-101	100~127	-100~+99
	符号なし	0~27	228~255	28~227
16 ビット	符号付き	-32768~-25664	25648~32752	-25663~+25647
	符号なし	0~7104	58416~65520	7105~58415

表 17-4 DS 本体のサウンド回路/TWL 本体の CODEC-DS モードのマイク入力値の保証範囲

振幅分解能		入力保証外範囲 下限側	入力保証外範囲 上限側	マイク入力値の保証範囲
8 ビット	符号付き	-128~-124	123~127	-123~+122
	符号なし	0~4	251~255	5~250
16 ビット	符号付き	-32768~-31744	31743~32752	-31743~+31742
	符号なし	0~1024	64512~65520	1025~64511

表 17-5 TWL 本体の CODEC-TWL モードのマイク入力値の保証範囲

マイクから入力された音声の録音とその録音された音声の再生を同時に連続して行った場合、本体によってはハウリングを引き起こし、正しい音声を再生できない恐れがありますので十分注意してください。

注意事項 1

TWL のマイクの API で有効ビット幅 12 ビットのサンプリングを行った場合、実際には下位 4 ビットが 0 パディングされた 16 ビットデータとして取得されます。そのため表 17-4 および表 17-5 の「入力保証外範囲」は 16 ビットの値を参照してください。

18. RTC（リアルタイムクロック）

TWL は本体内に RTC（リアルタイムクロック）を内蔵しています。
2099 年まではオートカレンダー機能により、うるう年に対応して計時されます。
計時の最大誤差は約 ±4 秒/日です。

以下のタイミングで時計のセットが行われます。

- 1) 本体購入後の初回電源投入時
- 2) バッテリー交換直後の電源投入時
- 3) バッテリー放電後（本体バッテリー切れ数ヵ月放置後）の再起動時
- 4) ブートメニューからの日付、時計セット時

RTC の API を利用することにより、サブプロセッサの動作について特に留意することなくアプリケーションで RTC を使用することができます。

API を使用して、表 18-1 に示すリアルタイムデータをリードすることができます。

データ種別	データ内容
リアルタイムデータ	年（00～99）、月（01～12）、日（01～31）、曜日（00～06）、 時（00～23）、分（00～59）、秒（00～59） それぞれの数値は BCD（2 進化 10 進符号）で格納されています。

表 18-1 リアルタイムデータ

また、2 系統のアラーム機能が用意されています。

アプリケーション側からアラームをセットし、スリープモードにすることで特定時刻にスリープモードから復帰させることができます。

スリープモードについては「パワーマネジメント」の章を参照してください。

API を使用して、表 18-2 に示すアラーム 1、アラーム 2 設定データのリードとライトができます。

データ種別	データ内容
アラーム 1、 アラーム 2	年（00～99）、月（01～12）、日（01～31）、曜日（00～06）、 時（00～23）、分（00～59） アラーム 1 とアラーム 2 のそれぞれを個別に設定可能です。 それぞれの数値は BCD（2 進化 10 進符号）で格納されています。 曜日、時、分のそれぞれについて有効/無効を設定することができます。 例、曜日の設定を無効にすると毎日同じ時刻にアラームを発生させることができます。

表 18-2 アラーム 1、アラーム 2 設定データ

注意事項

アプリケーション側から RTC にリアルタイムデータを書き込むことはできません。

19. 内蔵フラッシュメモリ

TWL には、タッチパネルのキャリブレーションや、オーナー情報、TWL 初期設定データ、RTC 操作情報データを保存するためのフラッシュメモリが内蔵されています。
内蔵フラッシュメモリは TWL 設定情報の保存専用メモリですので、アプリケーションから書き込むことができません。

19.1 タッチパネルのキャリブレーションデータ

タッチパネル個々の座標位置データのばらつきを吸収するための補正用データです。
アプリケーションでタッチパネルを使用する場合は、API を利用してキャリブレーションデータを読み込んでセットする必要があります。

19.2 オーナー情報データ

オーナー情報データには、TWL 所有者に関する情報が保存されています。
API を使用して、表 19-1 に示すオーナー情報データをリードすることができます。

データ種別	データ内容
オーナー情報データ	ユーザーID (22 バイト)、ユーザーカラー (1 バイト)、 誕生日 (2 バイト)、コメント (46 バイト)

表 19-1 オーナー情報データ

オーナー情報データの詳細を以下に示します。

- ユーザーID (ニックネーム)
ニックネームの文字列：UNICODE (UTF16) で最大 10 文字 (20 バイト)、終端コードはありません。
文字列の長さ：ニックネームの文字列長 (2 バイト)
- ユーザーカラー (好きな色)
0～15：IPL で決められた色セット 16 種類の中から選択 (括弧内は RGB の値)

0：GRAY	(12,16,19)	1：BROWN	(23, 9, 0)
2：RED	(31, 0, 3)	3：PINK	(31,17,31)
4：ORANGE	(31,18, 0)	5：YELLOW	(30,28, 0)
6：LIME GREEN	(21,31, 0)	7：GREEN	(0,31, 0)
8：DARK GREEN	(0,20, 7)	9：SEA GREEN	(9,27,17)
10：TURQUOISE	(6,23,30)	11：BLUE	(0,11,30)
12：DARK BLUE	(0, 0,18)	13：PURPLE	(17, 0,26)
14：VIOLET	(26, 0,29)	15：MAGENTA	(31, 0,18)
- 誕生日 (生月日) (それぞれの数値は BCD (2 進化 10 進符号) で格納されています)
月 (1 バイト)：生月日の月 (01～12)
日 (1 バイト)：生月日の日 (01～31)
- コメント
UNICODE (UTF16) で最大 23 文字×2 行 (23 バイト×2=46 バイト) のコメント。

19.3 TWL 初期設定データ

TWL 初期設定データには、使用言語の設定が保存されています。

API を使用して、表 19-2 に示す TWL 初期設定データをリードすることができます。

データ種別	データ内容
TWL 初期設定データ	使用言語設定（4 バイト）

表 19-2 TWL 初期設定データ

TWL 初期設定データの詳細を以下に示します。

- 使用言語設定
 - 0：日本語
 - 1：英語
 - 2：フランス語
 - 3：ドイツ語
 - 4：イタリア語
 - 5：スペイン語

19.4 RTC 操作情報データ

RTC の時計セットの際に、最初にセットしていたデータから何秒ずらしたのかが設定されます。これにより、ユーザーが RTC のデータを改変したことが分かります。

API を使用して、表 19-3 に示す RTC 操作情報データをリードすることができます。

データ種別	データ内容
RTC 操作情報データ	RTC オフセット値。 RTC 設定を変更する度に値が増減します。

表 19-3 RTC 操作情報データ

20. AES エンジン

TWL は、本体内に AES (Rijndael) 暗号化、復号化エンジンを内蔵しています。

AES とは、アメリカ合衆国の新暗号規格(Advanced Encryption Standard)として規格化された共通鍵方式のブロック暗号です。詳細は、『FIPS-197』を参照してください。

搭載されている AES エンジンの概要は次の通りです(図 20-1 参照)。

- 暗号モード CTR(カウンター)モード、CCM(CBC-MAC 付きカウンター)モード
それぞれ、暗号化、復号化ともにサポートしています。
- 鍵長 128 ビット
- ブロックサイズ 128 ビット

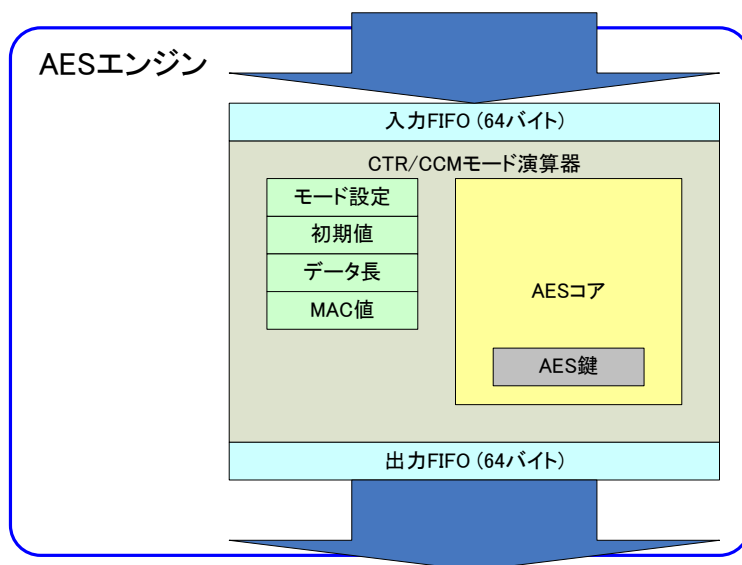


図 20-1 AES エンジン

AES の操作は、すべて API 経由で行ってください。

20.1 CTR モード

ブロック暗号方式の一種です(図 20-2 参照)。詳しくは、『NIST SP800-38A』の 6.5 を参照してください。

本 AES エンジンの制限として、入出力データは 16 バイト単位でなければなりません。

CTRモード (暗号化、復号化共通)

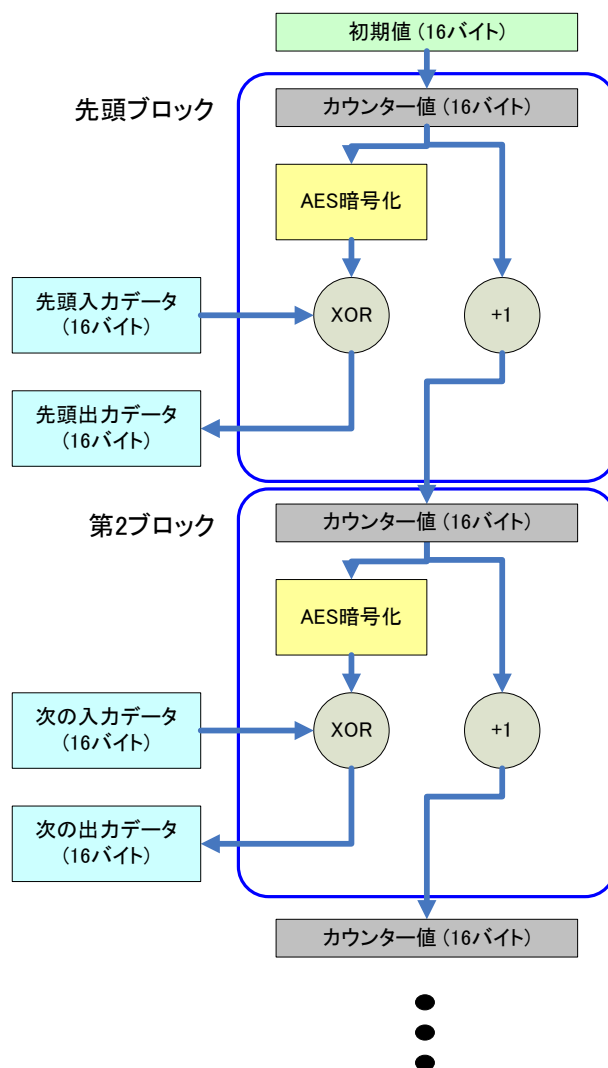


図 20-2 CTR モード

20.2 CCM モード

ブロック暗号方式の一種で、CTR モードと CBC-MAC (CBC モードの最後の出力データを MAC 値として利用したもの)を組み合わせたものです(図 20-3 参照)。詳しくは、『NIST SP800-38C』を参照してください。

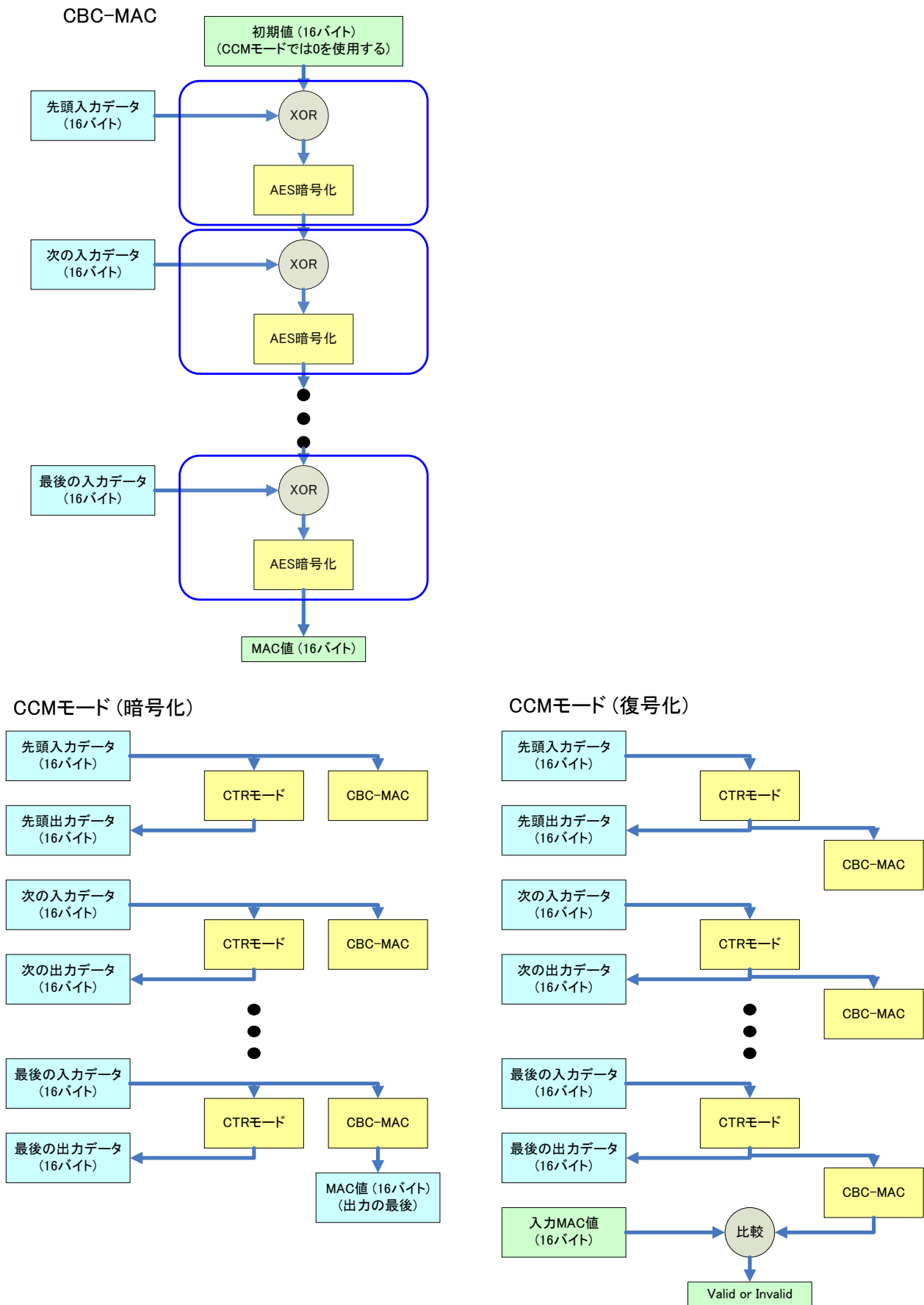


図 20-3 CCM モード

暗号化、復号化対象のデータ (Payload) とは別に、暗号化しないが CBC-MAC の計算には利用するデータ (Associated Data) を入力することもできます。

その場合、Associated Data、Payload の順に入力することになります。

また、それらのデータに先立って内部的に先頭ブロック(B0)を生成しています。

先頭ブロック(B0)の生成方法は、『NIST SP800-38C』の Appendix A で $q=3$ (固定)とした場合として実装されています。

上記仕様により、Nonce として指定できるサイズは 96 ビットとなります。

また、Associated Data については、サイズ情報の付加など、特別な処理は何も行いません。

必要であれば、あらかじめフォーマット処理をしたものを Associated Data として入力してください。

本 AES エンジンの制限として、入出力データは、Associated Data および Payload のそれぞれについて、16 バイト単位でなければなりません。

本 AES エンジンの機能として、Associated Data の出力を抑制することもできます。

20.3 注意事項

一般的な AES ライブラリでは、鍵データ、MAC データ、入出力データはビッグエンディアンであるとして演算されます。それに対し、本 AES エンジンではすべて 128 ビットまたは 96 ビットのリトルエンディアンとして取り扱われますので、注意が必要です。

21. カメラ

TWL は、本体内に 2 つのカメラを内蔵しています。

2 つのカメラは個別に制御できますが、同時に 2 つのカメラの画像を得ることはできません。

カメラの操作は、すべて API 経由で行ってください。

カメラの基本スペックは次の通りです。

- 絞り F2.8（固定）
- 画角（最大解像度で撮影時の値）

	最小	平均	最大
対角	63.0°	66.0°	69.0°
水平	52.2°	54.9°	57.6°
垂直	40.4°	42.6°	44.8°

- 撮影可能範囲：20cm～無限遠（パンフォーカス。マクロスイッチ非搭載）
- 最大解像度：VGA
- 最大フレームレート：30fps (fps: 1 秒間のフレーム数)
- 出力フォーマット：YCrYCb (YUV→RGB 変換回路により RGB555 として出力可能)
- 解像度の設定（設定の高速変更に対応）
- フレームレートの設定
- エフェクト処理（設定の高速変更に対応）
- 反転処理（設定の高速変更に対応）
- 撮影モードの設定
- ホワイトバランスの設定
- 露光の設定
- シャープネスの設定

このうち、解像度、エフェクト、反転処理については、それぞれのカメラでそれぞれ 2 つの設定を保持することができ、個別に設定する場合に比べて高速に設定を変更することができます。

注意事項

カメラが撮影して得られる画像データには、カメラ個体のばらつき、環境光によるばらつき、被写体自体の色のばらつきを含むことになります。例えば、カメラ個体のばらつきには画角、解像度、色再現性、回転、ディストーション等を含みます。そのため、同じアングル、同じカメラ設定で撮影して得られる画像は、本体毎に異なります。

21.1 カメラの設定

解像度やフレームレートなどのカメラの設定には以下のような種類があります。

- 解像度の設定
 - ・VGA (横:640 ピクセル、縦:480 ピクセル)
 - ・QVGA (横:320 ピクセル、縦:240 ピクセル)
 - ・QQVGA (横:160 ピクセル、縦:120 ピクセル)
 - ・CIF (横:352 ピクセル、縦:288 ピクセル)
 - ・QCIF (横:176 ピクセル、縦:144 ピクセル)
 - ・DS の液晶サイズ (横:256 ピクセル、縦:192 ピクセル)
 - ・DS の液晶サイズ 4 倍角 (横:512 ピクセル、縦:384 ピクセル)

● フレームレートの設定

- 15 fps 固定
- 15～5 fps 可変
- 15～2 fps 可変
- 8.5 fps 固定
- 5 fps 固定
- 20 fps 固定
- 20～5 fps 可変
- 30～5 fps 可変
- 30 fps 固定

● エフェクト処理

- 白黒 白黒(モノクロ)になります。
- セピア調 (黄) セピア調になります。全体的に黄土色のフィルタがかかり、懐かしい感じとなります。
- セピア調 (赤) セピア調になります。全体的に赤褐色のフィルタがかかり、懐かしい感じとなります。
- ネガ 明暗や色が反転し、青紫に色味がかかります。
- ネガ (フィルム調) 明暗や色が反転し、赤紫に色味がかかります。ネガの YUV データのうち U と V の順序を逆にした形になります。

● 反転処理

- 上下反転
- 左右反転
- 180 度回転

● 撮影モードの設定

- 文字撮影(QR コード、letter)
文字の輪郭を強調するため、シャープネスとコントラストを上げています。
名刺や新聞、答案用紙など比較的細かい文字を通常よりも視認しやすく撮像することができます。
ある程度のレベルの QR コードも利用することができます。
- 人物(portrait)
コントラストを下げ、標準よりも輪郭強調を下げています。
- 風景(landscape)
屋外での撮影を想定して、露出(AE)のレベルを標準より下げ、ホワイトバランスを DayLight 固定にしています。
- 暗視(night view)
低照度下で高いフレームレートでも撮像できるようにゲインを上げています。その分、画像ノイズが増えます。
シャープネスは標準よりも少し下げています。ホワイトバランスは自動のままです。
この設定で明るい環境下で撮像すると標準よりも画像ノイズは多くなります。
- 夜景(night snap)
専用のモードとして用意していませんが、風景モードと 15～2 fps 可変設定を組み合わせることで、夜景用の撮影モードにすることができます。
低照度下ではホワイトバランスを自動にしていると正常に働かないため、DayLight 固定にしています。また、フレームレートは低い方が画像ノイズが減るため、低照度では低いフレームレートになるよう 15～2fps 可変設定にしています。シャープネスは標準よりも少し上げています。

● ホワイトバランスの設定

- 標準 (自動調整)
- Tungsten (3200K)
- White fluorescent light (4150K)
- Day Light (5200K)
- Cloudy/Horizon (6000K)
- Shade (7000K)

また、自動調整(AWB)を一時的に停止させることもできます。

● 露光の設定

- 5～+5 で設定できます。
- また、自動調整(AE)を一時的に停止させることもできます。

- シャープネスの設定

-3～+5 で設定できます。

- RGB 変換

カメラからの画像データは、YCbCr (YUV422)フォーマットまたは RGB (RGB555)で得られます。

YUV422-RGB888 変換式は次の通りです。

$$\begin{aligned} R &= 1.000 \times Y && + 1.402 \times (V - 128) \\ G &= 1.000 \times Y &- 0.344 \times (U - 128) &- 0.714 \times (V - 128) \\ B &= 1.000 \times Y &+ 1.772 \times (U - 128) \end{aligned}$$

計算結果が 0 以下または 255 以上となった場合は丸められます。
その後、0～31 にスケーリングされ、RGB555 となります。

なお、逆変換式は次のようになります。

$$\begin{aligned} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ U &= -0.169 \times R - 0.331 \times G + 0.500 \times B + 128 \\ V &= 0.500 \times R - 0.419 \times G - 0.081 \times B + 128 \end{aligned}$$

21.2 トリミング

取り込む画像の一部をトリミングすることができます。

21.3 DMA 転送

カメラが取り込んだ画像をメインメモリに転送するために、TWL で新たに追加された DMA（新 DMA）が必ず使用されます。

DMA 転送の際、1 回の転送サイズは 2KB（1 ピクセル=2 バイトのため、1024 ピクセル）以内で指定する必要があります。

21.4 スリープモードからの復帰

スリープモードへ移行する際に、カメラは自動的にスタンバイ状態に入ります。スリープモードからの復帰時にカメラを有効にしたい場合は、アプリケーション側で対処する必要があります。

22. DSP

TWL は、CEVA 社製 XpertTeak™（以下、™ は省略）コアに、メインおよびサブのプロセッサとワーク RAM として共有することのできるローカルメモリを付加した DSP（Digital Signal Processor）ブロックを搭載しています。

22.1 CEVA-Teak コア

CEVA-Teak コアは、デュアル MAC（Multiply and Accumulation）、ワードサイズ 16 ビット、命令長 16 ビットの DSP コアで、以下のような特徴を持っています。

- 40 ビット×4 本のアキュムレータ
- 2 基の 16×16 ビット→32 ビット乗算器
- 単一サイクルでの乗算・加算の並列実行が可能
- ゼロオーバーヘッド・ループ

また、8 チャンネルの DMA や割り込みコントローラといった周辺回路を備えており、メインプロセッサとの間には複数の通信インタフェースが用意されています。

22.2 DSP メモリ

DSP コアに搭載されている 16 枚の 32KB メモリブレンは、DSP コアのローカルメモリとして使用するほかにも、メインプロセッサ、サブプロセッサのワーク RAM として使用することができます。

また、レジスタの設定により、メモリブレン毎にアクセスすることのできるシステムを指定（複数のコアが同時に同じメモリブレンにアクセスすることはできません）することができ、マスター（メモリブレンにアクセス可能なシステム）を切り替えてもメモリの内容は保持されます。

メモリのマッピングやマスターの切り替えはシステム制御のレジスタで行います。詳細については、「ワーク RAM」を参照してください。

22.3 メインプロセッサとの通信

DSP コアは、インタフェースレジスタによってメインプロセッサから制御することができます。このインタフェースは以下の機能を有しています。

- DSP のリセット
- DSP 内部メモリ空間へのデータの読み書き（DSP メモリ転送）
- コマンド・リプライレジスタによる少量データの授受
- セマフォレジスタによる ARM9・DSP 間の排他制御と割り込みの発生

22.3.1 リセット

DSP のコンフィギュレーションレジスタ (DSP_CFG) を操作することで、メインプロセッサから DSP にリセットをかけることができます。このリセット操作で DSP コアとその周辺回路が初期化されますが、インタフェースレジスタ自体はリセットされません。

また、コンフィギュレーションレジスタでは、DSP メモリ転送でアクセスするメモリ空間の選択や、読み込み・書き込み FIFO の状態による割り込み通知、読み出しデータ長などの設定を行うことができます。

DSP コンフィギュレーションレジスタ

名称 DSP_CFG アドレス 0x04004308 属性 R/W 初期値 0x0000

15	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMSEL			PRIE2	PRIE1	PRIE0	WFEIE	WFFIE	RFNEIE	RFFIE	RS	DRS	AIM	DSPR

- MEMSEL[d15~d12]: メモリセレクト

DSP 内部メモリ空間への読み書き (DSP メモリ転送) でアクセスするメモリを選択します。

	設定内容
0000	データメモリ
0001	ペリフェラル (MMIO) レジスタ
0101	プログラムメモリ (書き込みアクセスのみ)

- PRIE0~2[d09],[d10],[d11]: リプライレジスタ書き込み通知フラグ

1 に設定すると、対応するリプライレジスタ (DSP_REPx) に DSP コアがデータを書き込んだときに、メインプロセッサに割り込みが通知されます。

- WFEIE[d08]: 書き込み FIFO エンプティ通知フラグ

1 に設定すると、書き込み FIFO が空になったときに、メインプロセッサに割り込みが通知されます。

- WFFIE[d07]: 書き込み FIFO フル通知フラグ

1 に設定すると、書き込み FIFO が一杯になったときに、メインプロセッサに割り込みが通知されます。

- RFNEIE[d06]: 読み込み FIFO ノットエンプティ通知フラグ

1 に設定すると、読み込み FIFO が空でなくなったときに、メインプロセッサに割り込みが通知されます。

- RFFIE[d05]: 読み込み FIFO フル通知フラグ

1 に設定すると、読み込み FIFO が一杯になったときに、メインプロセッサに割り込みが通知されます。

- RS[d04]: DSP メモリ読み出し開始フラグ

1 を書き込むと、DSP メモリ空間からの読み出しアクセスを開始します (このとき読み込み FIFO がクリアされます)。0 を書き込むと、実行中の読み出しアクセスを中断します。

- DRS[d03~d02]: DSP メモリ読み出しデータ長

読み出しアクセスで読み出すデータの長さを指定します。

	設定内容
00	シングルアクセス (1 ワード)
01	8 ワードバースト読み出し
10	16 ワードバースト読み出し
11	フリーランニング (中断されるまで読み出しアクセスを続けます)

- AIM[d01]: アドレスオートインクリメントモード

1 に設定すると、DSP メモリ転送で転送アドレスを転送毎に自動的にインクリメントします。

- DSPR[d00]: DSP リセット

1 に設定すると、DSP コアとその周辺回路をリセットします。一度 1 に設定した場合は、少なくとも DSP クロックで 8 サイクルの間は 0 を書き込まないでください。

22.3.2 ステータスレジスタ

ステータスレジスタ（DSP_PSTS）によって、DSP とメインプロセッサの通信時の各状態を知ることができます。DSP メモリ転送で DSP からの割り込みをメインプロセッサに通知しないように設定した場合には、このレジスタのビット（WFEI, WFFI, RFNEI, RFFI）を調べることで読み込み・書き込み FIFO の状態を知ることができます。

DSP ステータスレジスタ

名称 DSP_PSTS アドレス 0x0400430C 属性 R 初期値 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCMD IM2	RCMD IM1	RCMD IM0	PRI2	PRI1	PRI0	PSEMI	WFEI	WFFI	RFNEI	RFFI			PRST	WTIP	RTIP

- RCMDIM0～2[d13],[d14],[d15]：コマンドレジスタ読み込みフラグ
メインプロセッサが DSP コマンドレジスタ（DSP_CMDx）に書き込んだ値を DSP が読み込んだかどうかを示します。

	内容
0	DSP はコマンドレジスタを読み込んだ
1	DSP はコマンドレジスタを読み込んでいない

- PRI0～2[d10],[d11],[d12]：リプライレジスタ更新フラグ
DSP が DSP リプライレジスタ（DSP_REPx）を更新したかどうかを示します。

	内容
0	DSP はリプライレジスタを更新した
1	DSP はリプライレジスタを更新していない

- PSEMI[d09]：セマフォレジスタ割り込み要求フラグ
1 のとき、DSP からのセマフォレジスタ（DSP_SEM）による割り込み要求がある（DSP セマフォマスクレジスタ（DSP_PMASK）でマスクされていないビットに 1 が書き込まれた）ことを示します。
- WFEI[d08]：書き込み FIFO エンプティフラグ
1 のとき、書き込み FIFO が空であることを示します。
- WFFI[d07]：書き込み FIFO フルフラグ
1 のとき、書き込み FIFO が一杯であることを示します。
- RFNEI[d06]：読み込み FIFO ノットエンプティフラグ
1 のとき、読み込み FIFO が空でないことを示します。DSP 転送データレジスタ（DSP_PDATA）から少なくとも 1 回は読み出すことができます。
- RFFI[d05]：読み込み FIFO フルフラグ
1 のとき、読み込み FIFO が一杯であることを示します。
- PRST[d02]：ペリフェラルリセットフラグ
1 のとき、ペリフェラル（周辺回路）がリセット処理の最中であることを示します。DSP にリセットをかけた後で、このビットが 0 になったことを確認すると、DSP コアをリセット状態にしたまま周辺回路にアクセスすることができます。
- WTIP[d01]：書き込み転送中フラグ
1 のとき、DSP メモリへの書き込み転送中であることを示します。
- RTIP[d00]：読み込み転送中フラグ
1 のとき、DSP メモリからの読み込み転送中であることを示します。

22.3.3 DSP メモリ転送

DSP メモリ転送を使うと、DSP 内部のメモリ空間へのアクセスが可能になります。アクセス可能なのは以下のメモリ空間です。

- すべてのデータメモリ
- プログラムメモリ（書き込みのみ）
- DSP 内部のペリフェラル（周辺回路）レジスタ（一部を除く）

TWL では、データメモリとプログラムメモリへのアクセスは、メモリバンクを切り替えて行う方が高速ですが、DSP メモリ転送は DSP が動作しているときに、DSP が参照しているメモリに同時にアクセスすることができます。

DSP メモリ転送では、メモリアccessの効率化のために読み込み用と書き込み用それぞれに 16 段の FIFO が用意されています。DSP コンフィギュレーションレジスタ（DSP_PCFG）では、FIFO の状態によってメインプロセッサに割り込みを通知させることができます。また、DSP ステータスレジスタ（DSP_PSTS）では、FIFO の状態を調べることができます。

DSP メモリ転送では、以下の 2 レジスタを使用します。

DSP 転送データレジスタ

名称 DSP_PDATA アドレス 0x04004300 属性 R/W 初期値 0x0000

15	8	7	0
PDATA			

メインプロセッサがこのレジスタを読み出したときは、読み込み FIFO から値がポップされ、その値を返します。書き込み転送中にこのレジスタに値を書き込んだときは、書き込み FIFO に値がプッシュされ、DMA 転送が開始されます。

DSP 転送アドレスレジスタ

名称 DSP_PADR アドレス 0x04004304 属性 W 初期値 0x0000

15	8	7	0
PADR			

アクセスする DSP メモリ空間のアドレスを指定します。このレジスタでは、下位 16 ビットを指定します。上位 16 ビットを指定するには、DSP 内部の DMA レジスタに値を設定する必要があります。このレジスタに書き込んだ値を読み出すことはできません。必ず 0x0000 が返されます。

22.3.4 コマンド・リプライレジスタ

メインプロセッサから DSP への通信用にコマンドレジスタ (DSP_CMDx)、DSP からメインプロセッサへの通信用にリプライレジスタ (DSP_REPx) をそれぞれ 3 組の 16 ビットレジスタとして備えています。

コマンドレジスタへの書き込みは DSP コアに割り込みで通知されます。また、DSP ステータスレジスタ (DSP_PSTS) のステータスビット (RCMDIM0~2) を調べることで、DSP がコマンドレジスタを読み出したかどうかを知ることができます。

DSP からリプライレジスタに書き込みが行われたときには、DSP コンフィギュレーションレジスタ (DSP_PCFG) の設定によっては、メインプロセッサに割り込みを通知することができますし、通知をせずに DSP ステータスレジスタをポーリングすることで書き込みを検知することもできます。

DSP コマンドレジスタ 0~2 (ARM9→DSP)

名称 DSP_CMDx (x=0, 1, 2) アドレス 0x04004320, 0x04004328, 0x04004330 属性 R/W 初期値 0x0000

15	8	7	0
APBP_CMDx			

メインプロセッサから DSP へ渡すデータを書き込みます。

DSP がこのレジスタを読み出したかどうかは、DSP ステータスレジスタの対応するビット (RCMDIMx) で調べることができます。

DSP リプライレジスタ 0~2 (DSP→ARM9)

名称 DSP_REPx (x=0, 1, 2) アドレス 0x04004324, 0x0400432C, 0x04004334 属性 R 初期値 0x0000

15	8	7	0
APBP_REPx			

DSP からメインプロセッサへ渡されるデータを読み出します。

DSP がこのレジスタを更新したときは、DSP ステータスレジスタの対応するビット (PRIx) に 1 がセットされます。また、DSP コンフィギュレーションレジスタの対応するビット (RRIEx) が 1 に設定されている場合は、メインプロセッサに割り込みを通知します。

この更新は DSP によるこのレジスタへの書き込みに対応していますので、値が変化していない (同じ値が書き込まれた) 場合でも更新されたことになります。

22.3.5 セマフォレジスタ

割り込みを使った、メインプロセッサと DSP 相互の通信用に 16 ビット（1 ビット×16）のセマフォレジスタを備えています。

DSP セマフォレジスタ（ARM9→DSP）

名称 DSP_PSEM アドレス 0x04004310 属性 R/W 初期値 0x0000

15	8	7	0
PSEM			

ビットに 1 を書き込むと、DSP 側のセマフォマスクレジスタ（APBP_MASK）でそのビットがマスクされていない限り、DSP 側に割り込みが通知されます。

DSP セマフォマスクレジスタ（DSP→ARM9）

名称 DSP_PMASK アドレス 0x04004314 属性 R/W 初期値 0x0000

15	8	7	0
PMASK			

ビットに 1 を設定しておく、DSP 側のセマフォレジスタ（APBP_PSEM）の対応するビットに 1 が書き込まれても、メインプロセッサに割り込みが通知されません。

DSP セマフォクリアレジスタ（DSP→ARM9）

名称 DSP_PCLEAR アドレス 0x04004318 属性 W 初期値 0x0000

15	8	7	0
PCLEAR			

ビットに 1 を書き込むと、セマフォレジスタ（DSP_PSEM）の対応するビットが 0 にクリアされます。書き込まれた内容はレジスタに残りませんので、クリアした後に 0 を書き込む必要はありません。

DSP セマフォデータレジスタ（DSP→ARM9）

名称 DSP_SEM アドレス 0x0400431C 属性 R 初期値 0x0000

15	8	7	0
APBP_PSEM			

DSP 側のセマフォレジスタ（APBP_PSEM）の内容を読み出すことができます。セマフォマスクレジスタ（DSP_PMASK）の、DSP コアが書き込んだ値の 1 のビットに対応するビットがすべて 1 でなければメインプロセッサに割り込みが通知されます。

22.4 DSP の起動手順

DSP ブロックを起動するには、以下の手順を踏んでください。

1. プログラム、データの準備
あらかじめ、ワーク RAM の WRAM-B に DSP 用のプログラムを、WRAM-C に DSP 用のデータを書き込み、それぞれのメモリブロックを DSP に割り当てます。
2. DSP のクロック、リセット設定による起動準備
 - i. システム設定の SCFG_RST レジスタの DSP_RSTB ビットが 0 であることを確認します。
 - ii. システム設定の SCFG_CLK レジスタの DSP_HCLK ビットを 1 にし、DSP ブロックにクロックを供給します。
 - iii. クロック供給後に DSP クロック (134MHz) で 8 サイクル以上待ってから、SCFG_RST レジスタの DSP_RSTB ビットを 1 にして DSP ブロックのリセットを解除します。
 - iv. この段階で DSP コアが一瞬起動しますが、ハードロジックによりスリープ状態となります。
3. ブート処理
 - i. DSP コンフィギュレーションレジスタ (DSP_PCFG) の DSPR ビットに 1 を書き込みます。
 - ii. DSP ステータスレジスタ (DSP_PSTS) の PRST ビットが 0 になるのを待ちます。(この間に、WRAM-B や WRAM-C のバンクを切り替えて内容を変更することができます)
 - iii. DSP コンフィギュレーションレジスタの DSPR ビットに 0 を書き込みます。
 - iv. DSP プログラム用メモリの 0 番地 (WRAM-B のマスターが DSP かつオフセットが 0 に設定されたメモリブロックの先頭番地) からプログラムの実行が開始されます。

システムリセット後に、初めて DSP ブロックを使用する場合は、上記の手順 1 と 2 を行った (どちらの手順が先でも構いません) 後で手順 3 を行ってください。二回目以降は、手順 3 を実行するだけで DSP ブロックの初期化 (DSP インタフェースレジスタを除く) を行うことができます。

DSP インタフェースレジスタを初期化するには、上記の手順 3-ii の段階で以下の手順を踏んでください。

1. DSP コンフィギュレーションレジスタの PRIE0~2 のビットすべてに 0 を書き込みます。
2. DSP セマフォレジスタ (DSP_PSEM) に 0x0000 を書き込みます。
3. DSP セマフォクリアレジスタ (DSP_PCLEAR) に 0xFFFF を書き込みます。
4. DSP リプライレジスタ 0~2 (DSP_REPx) を一回ずつ読み捨てます。

23. NAND フラッシュメモリ

TWL には、画像やダウンロードしたアプリケーションの保存のために、NAND フラッシュメモリが搭載されています。

TWL に搭載されている NAND フラッシュメモリは、以下のような特徴を持っています。

- 容量は 256MB
- 読み出し速度は毎秒 0.8～4.0MB
- 書き込み速度は毎秒 0.3～2.4MB

読み出し速度は毎回のばらつきが比較的少なく、書き込み速度は毎回のばらつきが比較的大きくなります。

NAND フラッシュメモリは読み出しや書き込みを繰り返すことで劣化し、その回数が増えてくると読み出しに 512 バイト単位で最大 13 ミリ秒の遅延が発生する頻度が高くなります。書き込み速度には影響がありません。

例：1900 バイトの読み出しでは、最大 39 ミリ秒の遅延が発生する頻度が高くなります。

注意事項

上記特性に関連して、SDK では NAND フラッシュメモリが劣化していなくても、稀に 2～13 ミリ秒のウェイトが入るようになっています。

また、読み出し、書き込みとも、ファイルポインタが 4 バイトアラインメントされていない場合はパフォーマンスが著しく低下してしまいます。

NAND フラッシュメモリは SD メモリカードとインタフェースを共有しているため、SD メモリカードと同時にアクセスすることはできません。

NAND フラッシュメモリにアクセスする場合は、必ず SDK で用意された API を使用してください。

24. SD メモリカード

TWL には、SD メモリカードのためのスロットが一機搭載されています。

TWL の SD メモリカードスロットは、以下のような特徴を持っています。

- 対応するメディアは、SD メモリカードおよび 2~32GB の SDHC メモリカード
- SPI モード、暗号認証 (CPRM) およびハイスピードモードには**非対応** (ハイスピードカードを使用しても、下記以上の速度ではアクセスできません)
- 最大読み出し速度は毎秒 6.0MB
- 最大書き込み速度は毎秒 2.4MB

画像や NAND アプリケーションをバックアップすることができます。
ファイルシステムは SD 標準フォーマットに対応しています。

アプリケーションは、通常のアクセスに加え、SD メモリカードの挿抜をきっかけとした処理を行うことができます。また、ライトプロテクト状態も知ることができます。

アクセス速度に関しては、ホスト側の制御だけでなく、SD カード側内部コントローラの制御にも依存するため、SD メモリカードにより異なります。

miniSD、microSD は SD カードアダプタを装着することでアクセスすることができます。ただし、microSD に miniSD アダプタを装着し、さらに SD カードアダプタを装着することは禁止されており、正常にアクセスできない場合があります。また、読み出し、書き込みとも、ファイルポインタが 4 バイトアラインメントされていない場合はパフォーマンスが著しく低下してしまいます。

SD メモリカードは NAND フラッシュメモリとインタフェースを共有しているため、NAND フラッシュメモリと同時にアクセスすることはできません。

スロットに挿入された SD メモリカードにアクセスする場合は、必ず SDK で用意された API を使用してください。

付録

付録A. レジスタ一覧表

A.1 0x04000000 番地以降

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x000	DISPCNT	78	2D グラフィックスエンジン A の表示コントロール
0x002			
0x004	DISPSTAT	75	表示ステータス
0x006	VCOUNT	76	V カウント比較
0x008	BG0CNT	98	2D グラフィックスエンジン A の BG0 コントロール
0x00A	BG1CNT	98	2D グラフィックスエンジン A の BG1 コントロール
0x00C	BG2CNT	100	2D グラフィックスエンジン A の BG2 コントロール
0x00E	BG3CNT	100	2D グラフィックスエンジン A の BG3 コントロール
0x010	BG0HOFS	121	2D グラフィックスエンジン A の BG0 表示 H オフセット
0x012	BG0VOFS	121	2D グラフィックスエンジン A の BG0 表示 V オフセット
0x014	BG1HOFS	121	2D グラフィックスエンジン A の BG1 表示 H オフセット
0x016	BG1VOFS	121	2D グラフィックスエンジン A の BG1 表示 V オフセット
0x018	BG2HOFS	121	2D グラフィックスエンジン A の BG2 表示 H オフセット
0x01A	BG2VOFS	121	2D グラフィックスエンジン A の BG2 表示 V オフセット
0x01C	BG3HOFS	121	2D グラフィックスエンジン A の BG3 表示 H オフセット
0x01E	BG3VOFS	121	2D グラフィックスエンジン A の BG3 表示 V オフセット
0x020	BG2PA	123	2D グラフィックスエンジン A の BG2 アフィン変換パ ラメータ (同ライン X 方向参照移動量 dx)
0x022	BG2PB	123	2D グラフィックスエンジン A の BG2 アフィン変換パ ラメータ (次ライン X 方向参照移動量 dmx)
0x024	BG2PC	123	2D グラフィックスエンジン A の BG2 アフィン変換パ ラメータ (同ライン Y 方向参照移動量 dy)
0x026	BG2PD	123	2D グラフィックスエンジン A の BG2 アフィン変換パ ラメータ (次ライン Y 方向参照移動量 dmy)
0x028	BG2X	123	2D グラフィックスエンジン A の BG2 参照開始点 (x 座標)
0x02A			
0x02C	BG2Y	123	2D グラフィックスエンジン A の BG2 参照開始点 (y 座標)
0x02E			
0x030	BG3PA	123	2D グラフィックスエンジン A の BG3 アフィン変換パ ラメータ (同ライン X 方向参照移動量 dx)
0x032	BG3PB	123	2D グラフィックスエンジン A の BG3 アフィン変換パ ラメータ (次ライン X 方向参照移動量 dmx)
0x034	BG3PC	123	2D グラフィックスエンジン A の BG3 アフィン変換パ ラメータ (同ライン Y 方向参照移動量 dy)
0x036	BG3PD	123	2D グラフィックスエンジン A の BG3 アフィン変換パ ラメータ (次ライン Y 方向参照移動量 dmy)
0x038	BG3X	123	2D グラフィックスエンジン A の BG3 参照開始点 (x 座標)
0x03A			
0x03C	BG3Y	123	2D グラフィックスエンジン A の BG3 参照開始点 (y 座標)
0x03E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x040	WIN0H	154	2D グラフィックスエンジン A のウィンドウ 0H サイズ
0x042	WIN1H	154	2D グラフィックスエンジン A のウィンドウ 1H サイズ
0x044	WIN0V	155	2D グラフィックスエンジン A のウィンドウ 0V サイズ
0x046	WIN1V	155	2D グラフィックスエンジン A のウィンドウ 1V サイズ
0x048	WININ	154	2D グラフィックスエンジン A のウィンドウ内側
0x04A	WINOUT	154	2D グラフィックスエンジン A のウィンドウ外側
0x04C	MOSAIC	160	2D グラフィックスエンジン A のモザイクサイズ
0x04E			
0x050	BLDCNT	157	2D グラフィックスエンジン A のカラー特殊効果
0x052	BLDALPHA	158	2D グラフィックスエンジン A の α ブレンディング係数
0x054	BLDY	159	2D グラフィックスエンジン A の輝度変更係数
0x056			
0x058			
0x05A			
0x05C			
0x05E			
0x060	DISP3DCNT	164	3D 表示コントロール
0x062			
0x064	DISPCAPCNT	87	表示キャプチャ
0x066			
0x068	DISP_MMEN_FIFO	86	メインメモリ表示 FIFO
0x06A			
0x06C	MASTER_BRIGHT	90	画像出力 A のマスター輝度
0x06E			
0x070			
0x072			
0x074			
0x076			
0x078			
0x07A			
0x07C			
0x07E			
0x080			
0x082			
0x084			
0x086			
0x088			
0x08A			
0x08C			
0x08E			
0x090			
0x092			
0x094			
0x096			
0x098			
0x09A			
0x09C			
0x09E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x0A0			
0x0A2			
0x0A4			
0x0A6			
0x0A8			
0x0AA			
0x0AC			
0x0AE			
0x0B0	DMA0SAD	266	DMA0 ソースアドレス
0x0B2			
0x0B4	DMA0DAD	266	DMA0 デスティネーションアドレス
0x0B6			
0x0B8	DMA0CNT	267	DMA0 コントロール
0x0BA			
0x0BC	DMA1SAD	266	DMA1 ソースアドレス
0x0BE			
0x0C0	DMA1DAD	266	DMA1 デスティネーションアドレス
0x0C2			
0x0C4	DMA1CNT	267	DMA1 コントロール
0x0C6			
0x0C8	DMA2SAD	266	DMA2 ソースアドレス
0x0CA			
0x0CC	DMA2DAD	266	DMA2 デスティネーションアドレス
0x0CE			
0x0D0	DMA2CNT	267	DMA2 コントロール
0x0D2			
0x0D4	DMA3SAD	266	DMA3 ソースアドレス
0x0D6			
0x0D8	DMA3DAD	266	DMA3 デスティネーションアドレス
0x0DA			
0x0DC	DMA3CNT	267	DMA3 コントロール
0x0DE			
0x0E0			
0x0E2			
0x0E4			
0x0E6			
0x0E8			
0x0EA			
0x0EC			
0x0EE			
0x0F0			
0x0F2			
0x0F4			
0x0F6			
0x0F8			
0x0FA			
0x0FC			
0x0FE			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x100	TM0CNT_L	276	タイマ 0 カウンタ
0x102	TM0CNT_H	276	タイマ 0 コントロール
0x104	TM1CNT_L	277	タイマ 1 カウンタ
0x106	TM1CNT_H	277	タイマ 1 コントロール
0x108	TM2CNT_L	277	タイマ 2 カウンタ
0x10A	TM2CNT_H	277	タイマ 2 コントロール
0x10C	TM3CNT_L	277	タイマ 3 カウンタ
0x10E	TM3CNT_H	277	タイマ 3 コントロール
0x110			
0x112			
0x114			
0x116			
0x118			
0x11A			
0x11C			
0x11E			
0x120			
0x122			
0x124			
0x126			
0x128			
0x12A			
0x12C			
0x12E			
0x130	KEYINPUT	291	キー入力
0x132	KEYCNT	292	キーコントロール
0x134			
0x136			
0x138			
0x13A			
0x13C			
0x13E			
0x140			
0x142			
0x144			
0x146			
0x148			
0x14A			
0x14C			
0x14E			
0x150			
0x152			
0x154			
0x156			
0x158			
0x15A			
0x15C			
0x15E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x160			
0x162			
0x164			
0x166			
0x168			
0x16A			
0x16C			
0x16E			
0x170			
0x172			
0x174			
0x176			
0x178			
0x17A			
0x17C			
0x17E			
0x180			
0x182			
0x184			
0x186			
0x188			
0x18A			
0x18C			
0x18E			
0x190			
0x192			
0x194			
0x196			
0x198			
0x19A			
0x19C			
0x19E			
0x1A0			
0x1A2			
0x1A4			
0x1A6			
0x1A8			
0x1AA			
0x1AC			
0x1AE			
0x1B0			
0x1B2			
0x1B4			
0x1B6			
0x1B8			
0x1BA			
0x1BC			
0x1BE			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x1C0			
0x1C2			
0x1C4			
0x1C6			
0x1C8			
0x1CA			
0x1CC			
0x1CE			
0x1D0			
0x1D2			
0x1D4			
0x1D6			
0x1D8			
0x1DA			
0x1DC			
0x1DE			
0x1E0			
0x1E2			
0x1E4			
0x1E6			
0x1E8			
0x1EA			
0x1EC			
0x1EE			
0x1F0			
0x1F2			
0x1F4			
0x1F6			
0x1F8			
0x1FA			
0x1FC			
0x1FE			
0x200			
0x202			
0x204	EXMEMCNT	30	外部メモリコントロール
0x206			
0x208	IME	278	割り込みマスターフラグ
0x20A			
0x20C			
0x20E			
0x210	IE	279	割り込み許可フラグ
0x212			
0x214	IF	280	割り込み要求フラグ
0x216			
0x218			
0x21A			
0x21C			
0x21E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x220			
0x222			
0x224			
0x226			
0x228			
0x22A			
0x22C			
0x22E			
0x230			
0x232			
0x234			
0x236			
0x238			
0x23A			
0x23C			
0x23E			
0x240	VRAMCNT	40	RAM バンクコントロール 0
0x242			
0x244	WVRAMCNT	42	RAM バンクコントロール 1
0x246			
0x248	VRAM_HI_CNT	44	RAM バンクコントロール 2
0x24A			
0x24C			
0x24E			
0x250			
0x252			
0x254			
0x256			
0x258			
0x25A			
0x25C			
0x25E			
0x260			
0x262			
0x264			
0x266			
0x268			
0x26A			
0x26C			
0x26E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x270			
0x272			
0x274			
0x276			
0x278			
0x27A			
0x27C			
0x27E			
0x280	DIVCNT	288	ディバイダコントロール
0x282			
0x284			
0x286			
0x288			
0x28A			
0x28C			
0x28E			
0x290	DIV_NUMER	288	被除数
0x292			
0x294			
0x296			
0x298	DIV_DENOM	288	除数
0x29A			
0x29C			
0x29E			
0x2A0	DIV_RESULT	288	商
0x2A2			
0x2A4			
0x2A6			
0x2A8	DIVREM_RESULT	288	剰余
0x2AA			
0x2AC			
0x2AE			
0x2B0	SQRTCNT	290	平方根演算器コントロール
0x2B2			
0x2B4	SQRT_RESULT	290	平方根演算器演算結果
0x2B6			
0x2B8	SQRT_PARAM	290	平方根演算器データ
0x2BA			
0x2BC			
0x2BE			
0x2C0			
0x2C2			
0x2C4			
0x2C6			
0x2C8			
0x2CA			
0x2CC			
0x2CE			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x2D0			
0x2D2			
0x2D4			
0x2D6			
0x2D8			
0x2DA			
0x2DC			
0x2DE			
0x2E0			
0x2E2			
0x2E4			
0x2E6			
0x2E8			
0x2EA			
0x2EC			
0x2EE			
0x2F0			
0x2F2			
0x2F4			
0x2F6			
0x2F8			
0x2FA			
0x2FC			
0x2FE			
0x300			
0x302			
0x304	POWCNT	77	パワーコントロール
0x306			
0x308			
0x30A			
0x30C			
0x30E			
0x310			
0x312			
0x314			
0x316			
0x318			
0x31A			
0x31C			
0x31E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x320	RDLINES_COUNT	261	レンダリング最小充填度
0x322			
0x324			
0x326			
0x328			
0x32A			
0x32C			
0x32E			
0x330	EDGE_COLOR_0_L	254	エッジマーキングカラー (ポリゴン ID 上位 3 ビットが 000)
0x332	EDGE_COLOR_0_H	254	エッジマーキングカラー (ポリゴン ID 上位 3 ビットが 001)
0x334	EDGE_COLOR_1_L	254	エッジマーキングカラー (ポリゴン ID 上位 3 ビットが 010)
0x336	EDGE_COLOR_1_H	254	エッジマーキングカラー (ポリゴン ID 上位 3 ビットが 011)
0x338	EDGE_COLOR_2_L	254	エッジマーキングカラー (ポリゴン ID 上位 3 ビットが 100)
0x33A	EDGE_COLOR_2_H	254	エッジマーキングカラー (ポリゴン ID 上位 3 ビットが 101)
0x33C	EDGE_COLOR_3_L	254	エッジマーキングカラー (ポリゴン ID 上位 3 ビットが 110)
0x33E	EDGE_COLOR_3_H	254	エッジマーキングカラー (ポリゴン ID 上位 3 ビットが 111)
0x340	ALPHA_TEST_REF	253	α テスト
0x342			
0x344			
0x346			
0x348			
0x34A			
0x34C			
0x34E			
0x350	CLEAR_COLOR	228	カラーバッファ初期値
0x352			
0x354	CLEAR_DEPTH	228	デプスバッファ初期値
0x356	CLRIMAGE_OFFSET	230	クリアイメージオフセット

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x358	FOG_COLOR	255	フォグカラー
0x35A			
0x35C	FOG_OFFSET	255	フォグオフセット
0x35E			
0x360	FOG_TABLE_0_L	256	フォグ濃度テーブル (0、1)
0x362	FOG_TABLE_0_H	256	フォグ濃度テーブル (2、3)
0x364	FOG_TABLE_1_L	256	フォグ濃度テーブル (4、5)
0x366	FOG_TABLE_1_H	256	フォグ濃度テーブル (6、7)
0x368	FOG_TABLE_2_L	256	フォグ濃度テーブル (8、9)
0x36A	FOG_TABLE_2_H	256	フォグ濃度テーブル (10、11)
0x36C	FOG_TABLE_3_L	256	フォグ濃度テーブル (12、13)
0x36E	FOG_TABLE_3_H	256	フォグ濃度テーブル (14、15)
0x370	FOG_TABLE_4_L	256	フォグ濃度テーブル (16、17)
0x372	FOG_TABLE_4_H	256	フォグ濃度テーブル (18、19)
0x374	FOG_TABLE_5_L	256	フォグ濃度テーブル (20、21)
0x376	FOG_TABLE_5_H	256	フォグ濃度テーブル (22、23)
0x378	FOG_TABLE_6_L	256	フォグ濃度テーブル (24、25)
0x37A	FOG_TABLE_6_H	256	フォグ濃度テーブル (26、27)
0x37C	FOG_TABLE_7_L	256	フォグ濃度テーブル (28、29)
0x37E	FOG_TABLE_7_H	256	フォグ濃度テーブル (30、31)
0x380	TOON_TABLE_0_L	236	トゥーンテーブル (輝度 0 のときの RGB 変換値)
0x382	TOON_TABLE_0_H	236	トゥーンテーブル (輝度 1 のときの RGB 変換値)
0x384	TOON_TABLE_1_L	236	トゥーンテーブル (輝度 2 のときの RGB 変換値)
0x386	TOON_TABLE_1_H	236	トゥーンテーブル (輝度 3 のときの RGB 変換値)
0x388	TOON_TABLE_2_L	236	トゥーンテーブル (輝度 4 のときの RGB 変換値)
0x38A	TOON_TABLE_2_H	236	トゥーンテーブル (輝度 5 のときの RGB 変換値)
0x38C	TOON_TABLE_3_L	236	トゥーンテーブル (輝度 6 のときの RGB 変換値)
0x38E	TOON_TABLE_3_H	236	トゥーンテーブル (輝度 7 のときの RGB 変換値)
0x390	TOON_TABLE_4_L	236	トゥーンテーブル (輝度 8 のときの RGB 変換値)
0x392	TOON_TABLE_4_H	236	トゥーンテーブル (輝度 9 のときの RGB 変換値)
0x394	TOON_TABLE_5_L	236	トゥーンテーブル (輝度 10 のときの RGB 変換値)
0x396	TOON_TABLE_5_H	236	トゥーンテーブル (輝度 11 のときの RGB 変換値)
0x398	TOON_TABLE_6_L	236	トゥーンテーブル (輝度 12 のときの RGB 変換値)
0x39A	TOON_TABLE_6_H	236	トゥーンテーブル (輝度 13 のときの RGB 変換値)
0x39C	TOON_TABLE_7_L	236	トゥーンテーブル (輝度 14 のときの RGB 変換値)
0x39E	TOON_TABLE_7_H	236	トゥーンテーブル (輝度 15 のときの RGB 変換値)
0x3A0	TOON_TABLE_8_L	236	トゥーンテーブル (輝度 16 のときの RGB 変換値)
0x3A2	TOON_TABLE_8_H	236	トゥーンテーブル (輝度 17 のときの RGB 変換値)
0x3A4	TOON_TABLE_9_L	236	トゥーンテーブル (輝度 18 のときの RGB 変換値)
0x3A6	TOON_TABLE_9_H	236	トゥーンテーブル (輝度 19 のときの RGB 変換値)
0x3A8	TOON_TABLE_10_L	236	トゥーンテーブル (輝度 20 のときの RGB 変換値)
0x3AA	TOON_TABLE_10_H	236	トゥーンテーブル (輝度 21 のときの RGB 変換値)
0x3AC	TOON_TABLE_11_L	236	トゥーンテーブル (輝度 22 のときの RGB 変換値)
0x3AE	TOON_TABLE_11_H	236	トゥーンテーブル (輝度 23 のときの RGB 変換値)
0x3B0	TOON_TABLE_12_L	236	トゥーンテーブル (輝度 24 のときの RGB 変換値)
0x3B2	TOON_TABLE_12_H	236	トゥーンテーブル (輝度 25 のときの RGB 変換値)
0x3B4	TOON_TABLE_13_L	236	トゥーンテーブル (輝度 26 のときの RGB 変換値)
0x3B6	TOON_TABLE_13_H	236	トゥーンテーブル (輝度 27 のときの RGB 変換値)
0x3B8	TOON_TABLE_14_L	236	トゥーンテーブル (輝度 28 のときの RGB 変換値)
0x3BA	TOON_TABLE_14_H	236	トゥーンテーブル (輝度 29 のときの RGB 変換値)
0x3BC	TOON_TABLE_15_L	236	トゥーンテーブル (輝度 30 のときの RGB 変換値)
0x3BE	TOON_TABLE_15_H	236	トゥーンテーブル (輝度 31 のときの RGB 変換値)

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x3C0			
0x3C2			
0x3C4			
0x3C6			
0x3C8			
0x3CA			
0x3CC			
0x3CE			
0x3D0			
0x3D2			
0x3D4			
0x3D6			
0x3D8			
0x3DA			
0x3DC			
0x3DE			
0x3E0			
0x3E2			
0x3E4			
0x3E6			
0x3E8			
0x3EA			
0x3EC			
0x3EE			
0x3F0			
0x3F2			
0x3F4			
0x3F6			
0x3F8			
0x3FA			
0x3FC			
0x3FE			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x400	GXFIFO	174	ジオメトリコマンド FIFO
0x402			
0x404	GXFIFO のイメージ		
0x406			
0x408			
0x40A			
0x40C			
0x40E			
0x410			
0x412			
0x414			
0x416			
0x418			
0x41A			
0x41C			
0x41E			
0x420			
0x422			
0x424			
0x426			
0x428			
0x42A			
0x42C			
0x42E			
0x430			
0x432			
0x434			
0x436			
0x438			
0x43A			
0x43C			
0x43E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x440	MTX_MODE	185	カレント行列モード設定
0x442			
0x444	MTX_PUSH	189	カレント行列をスタックへプッシュ
0x446			
0x448	MTX_POP	189	カレント行列をスタックからポップ
0x44A			
0x44C	MTX_STORE	190	カレント行列をスタックの指定位置に格納
0x44E			
0x450	MTX_RESTORE	190	スタックの指定位置から行列を読み出す
0x452			
0x454	MTX_IDENTITY	186	単位行列に初期化
0x456			
0x458	MTX_LOAD_4x4	186	4×4 行列を設定
0x45A			
0x45C	MTX_LOAD_4x3	186	4×3 行列を設定
0x45E			
0x460	MTX_MULT_4x4	187	4×4 行列を乗算
0x462			
0x464	MTX_MULT_4x3	187	4×3 行列を乗算
0x466			
0x468	MTX_MULT_3x3	187	3×3 行列を乗算
0x46A			
0x46C	MTX_SCALE	188	スケール行列を乗算
0x46E			
0x470	MTX_TRANS	188	平行移動行列を乗算
0x472			
0x474			
0x476			
0x478			
0x47A			
0x47C			
0x47E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x480	COLOR	202	頂点カラー
0x482			
0x484	NORMAL	202	法線ベクトル
0x486			
0x488	TEXCOORD	206	テクスチャ座標
0x48A			
0x48C	VTX_16	203	頂点座標
0x48E			
0x490	VTX_10	203	頂点座標
0x492			
0x494	VTX_XY	203	頂点の XY 座標
0x496			
0x498	VTX_XZ	203	頂点の XZ 座標
0x49A			
0x49C	VTX_YZ	203	頂点の YZ 座標
0x49E			
0x4A0	VTX_DIFF	204	頂点座標の差分値指定
0x4A2			
0x4A4	POLYGON_ATTR	198	ポリゴン関連属性値
0x4A6			
0x4A8	TEXIMAGE_PARAM	207	テクスチャのパラメータ
0x4AA			
0x4AC	TEXPLTT_BASE	210	テクスチャパレットのベースアドレス
0x4AE			
0x4B0			
0x4B2			
0x4B4			
0x4B6			
0x4B8			
0x4BA			
0x4BC			
0x4BE			
0x4C0	DIF_AMB	195	材質の拡散反射色と環境反射色
0x4C2			
0x4C4	SPE_EMI	195	材質の鏡面反射色と放射光色
0x4C6			
0x4C8	LIGHT_VECTOR	192	ライトの方向ベクトル
0x4CA			
0x4CC	LIGHT_COLOR	192	ライトカラー
0x4CE			
0x4D0	SHININESS	196	鏡面反射輝度テーブル
0x4D2			
0x4D4			
0x4D6			
0x4D8			
0x4DA			
0x4DC			
0x4DE			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x4E0			
0x4E2			
0x4E4			
0x4E6			
0x4E8			
0x4EA			
0x4EC			
0x4EE			
0x4F0			
0x4F2			
0x4F4			
0x4F6			
0x4F8			
0x4FA			
0x4FC			
0x4FE			
0x500	BEGIN_VTXS	201	頂点リストの開始
0x502			
0x504	END_VTXS	202	頂点リストの終了
0x506			
0x508			
0x50A			
0x50C			
0x50E			
0x510			
0x512			
0x514			
0x516			
0x518			
0x51A			
0x51C			
0x51E			
0x520			
0x522			
0x524			
0x526			
0x528			
0x52A			
0x52C			
0x52E			
0x530			
0x532			
0x534			
0x536			
0x538			
0x53A			
0x53C			
0x53E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x540	SWAP_BUFFERS	183	データ群のスワップ
0x542			
0x544			
0x546			
0x548			
0x54A			
0x54C			
0x54E			
0x550			
0x552			
0x554			
0x556			
0x558			
0x55A			
0x55C			
0x55E			
0x560			
0x562			
0x564			
0x566			
0x568			
0x56A			
0x56C			
0x56E			
0x570			
0x572			
0x574			
0x576			
0x578			
0x57A			
0x57C			
0x57E			
0x580	VIEWPORT	184	ビューポート
0x582			
0x584			
0x586			
0x588			
0x58A			
0x58C			
0x58E			
0x590			
0x592			
0x594			
0x596			
0x598			
0x59A			
0x59C			
0x59E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x5A0			
0x5A2			
0x5A4			
0x5A6			
0x5A8			
0x5AA			
0x5AC			
0x5AE			
0x5B0			
0x5B2			
0x5B4			
0x5B6			
0x5B8			
0x5BA			
0x5BC			
0x5BE			
0x5C0	BOX_TEST	214	ボックステスト
0x5C2			
0x5C4	POS_TEST	215	位置座標テスト
0x5C6			
0x5C8	VEC_TEST	215	方向ベクトルテスト
0x5CA			
0x5CC			
0x5CE			
0x5D0			
0x5D2			
0x5D4			
0x5D6			
0x5D8			
0x5DA			
0x5DC			
0x5DE			
0x5E0			
0x5E2			
0x5E4			
0x5E6			
0x5E8			
0x5EA			
0x5EC			
0x5EE			
0x5F0			
0x5F2			
0x5F4			
0x5F6			
0x5F8			
0x5FA			
0x5FC			
0x5FE			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x600	GXSTAT	216	ジオメトリエンジンのステータス
0x602			
0x604	LISTRAM_COUNT	218	ポリゴンリスト RAM カウント
0x606	VTXRAM_COUNT	218	頂点 RAM カウント
0x608			
0x60A			
0x60C			
0x60E			
0x610	DISP_1DOT_DEPTH	200	1 ドットポリゴン表示境界デプス値
0x612			
0x614			
0x616			
0x618			
0x61A			
0x61C			
0x61E			
0x620	POS_RESULT_X	215	位置座標テストの結果（クリップ座標 X 成分）
0x622			
0x624	POS_RESULT_Y	215	位置座標テストの結果（クリップ座標 Y 成分）
0x626			
0x628	POS_RESULT_Z	215	位置座標テストの結果（クリップ座標 Z 成分）
0x62A			
0x62C	POS_RESULT_W	215	位置座標テストの結果（クリップ座標 W 成分）
0x62E			
0x630	VEC_RESULT_X	215	方向ベクトルテストの結果（X 成分）
0x632	VEC_RESULT_Y	215	方向ベクトルテストの結果（Y 成分）
0x634	VEC_RESULT_Z	215	方向ベクトルテストの結果（Z 成分）
0x636			
0x638			
0x63A			
0x63C			
0x63E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x640	CLIPMTX_RESULT_0	191	カレントクリップ座標行列（要素 m0）
0x642			
0x644	CLIPMTX_RESULT_1	191	カレントクリップ座標行列（要素 m1）
0x646			
0x648	CLIPMTX_RESULT_2	191	カレントクリップ座標行列（要素 m2）
0x64A			
0x64C	CLIPMTX_RESULT_3	191	カレントクリップ座標行列（要素 m3）
0x64E			
0x650	CLIPMTX_RESULT_4	191	カレントクリップ座標行列（要素 m4）
0x652			
0x654	CLIPMTX_RESULT_5	191	カレントクリップ座標行列（要素 m5）
0x656			
0x658	CLIPMTX_RESULT_6	191	カレントクリップ座標行列（要素 m6）
0x65A			
0x65C	CLIPMTX_RESULT_7	191	カレントクリップ座標行列（要素 m7）
0x65E			
0x660	CLIPMTX_RESULT_8	191	カレントクリップ座標行列（要素 m8）
0x662			
0x664	CLIPMTX_RESULT_9	191	カレントクリップ座標行列（要素 m9）
0x666			
0x668	CLIPMTX_RESULT_10	191	カレントクリップ座標行列（要素 m10）
0x66A			
0x66C	CLIPMTX_RESULT_11	191	カレントクリップ座標行列（要素 m11）
0x66E			
0x670	CLIPMTX_RESULT_12	191	カレントクリップ座標行列（要素 m12）
0x672			
0x674	CLIPMTX_RESULT_13	191	カレントクリップ座標行列（要素 m13）
0x676			
0x678	CLIPMTX_RESULT_14	191	カレントクリップ座標行列（要素 m14）
0x67A			
0x67C	CLIPMTX_RESULT_15	191	カレントクリップ座標行列（要素 m15）
0x67E			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x680	VECMTX_RESULT_0	191	カレント方向ベクトル行列（要素 m0）
0x682			
0x684	VECMTX_RESULT_1	191	カレント方向ベクトル行列（要素 m1）
0x686			
0x688	VECMTX_RESULT_2	191	カレント方向ベクトル行列（要素 m2）
0x68A			
0x68C	VECMTX_RESULT_3	191	カレント方向ベクトル行列（要素 m3）
0x68E			
0x690	VECMTX_RESULT_4	191	カレント方向ベクトル行列（要素 m4）
0x692			
0x694	VECMTX_RESULT_5	191	カレント方向ベクトル行列（要素 m5）
0x696			
0x698	VECMTX_RESULT_6	191	カレント方向ベクトル行列（要素 m6）
0x69A			
0x69C	VECMTX_RESULT_7	191	カレント方向ベクトル行列（要素 m7）
0x69E			
0x6A0	VECMTX_RESULT_8	191	カレント方向ベクトル行列（要素 m8）
0x6A2			
0x6A4			
0x6A6			
0x6A8			
0x6AA			
0x6AC			
0x6AE			
0x6B0			
0x6B2			
0x6B4			
0x6B6			
0x6B8			
0x6BA			
0x6BC			
0x6BE			
0x6C0			
0x6C2			
0x6C4			
0x6C6			
0x6C8			
0x6CA			
0x6CC			
0x6CE			
0x6D0			
0x6D2			
0x6D4			
0x6D6			
0x6D8			
0x6DA			
0x6DC			
0x6DE			

※ 0x04000000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x6E0			
0x6E2			
0x6E4			
0x6E6			
0x6E8			
0x6EA			
0x6EC			
0x6EE			
0x6F0			
0x6F2			
0x6F4			
0x6F6			
0x6F8			
0x6FA			
0x6FC			
0x6FE			

※ 0x04000000 からのオフセット値です。

A.2 0x04001000 番地以降（2D グラフィックスエンジン B 関連）

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x000	DB_DISPCNT	80	2D グラフィックスエンジン B の表示コントロール
0x002			
0x004			
0x006			
0x008	DB_BG0CNT	98	2D グラフィックスエンジン B の BG0 コントロール
0x00A	DB_BG1CNT	98	2D グラフィックスエンジン B の BG1 コントロール
0x00C	DB_BG2CNT	100	2D グラフィックスエンジン B の BG2 コントロール
0x00E	DB_BG3CNT	100	2D グラフィックスエンジン B の BG3 コントロール
0x010	DB_BG0HOFS	121	2D グラフィックスエンジン B の BG0 表示 H オフセット
0x012	DB_BG0VOFS	121	2D グラフィックスエンジン B の BG0 表示 V オフセット
0x014	DB_BG1HOFS	121	2D グラフィックスエンジン B の BG1 表示 H オフセット
0x016	DB_BG1VOFS	121	2D グラフィックスエンジン B の BG1 表示 V オフセット
0x018	DB_BG2HOFS	121	2D グラフィックスエンジン B の BG2 表示 H オフセット
0x01A	DB_BG2VOFS	121	2D グラフィックスエンジン B の BG2 表示 V オフセット
0x01C	DB_BG3HOFS	121	2D グラフィックスエンジン B の BG3 表示 H オフセット
0x01E	DB_BG3VOFS	121	2D グラフィックスエンジン B の BG3 表示 V オフセット
0x020	DB_BG2PA	123	2D グラフィックスエンジン B の BG2 アフィン変換パ ラメータ（同ライン X 方向参照移動量 dx）
0x022	DB_BG2PB	123	2D グラフィックスエンジン B の BG2 アフィン変換パ ラメータ（次ライン X 方向参照移動量 dmx）
0x024	DB_BG2PC	123	2D グラフィックスエンジン B の BG2 アフィン変換パ ラメータ（同ライン Y 方向参照移動量 dy）
0x026	DB_BG2PD	123	2D グラフィックスエンジン B の BG2 アフィン変換パ ラメータ（次ライン Y 方向参照移動量 dmy）
0x028	DB_BG2X	123	2D グラフィックスエンジン B の BG2 参照開始点 （x 座標）
0x02A			
0x02C	DB_BG2Y	123	2D グラフィックスエンジン B の BG2 参照開始点 （y 座標）
0x02E			
0x030	DB_BG3PA	123	2D グラフィックスエンジン B の BG3 アフィン変換パ ラメータ（同ライン X 方向参照移動量 dx）
0x032	DB_BG3PB	123	2D グラフィックスエンジン B の BG3 アフィン変換パ ラメータ（次ライン X 方向参照移動量 dmx）
0x034	DB_BG3PC	123	2D グラフィックスエンジン B の BG3 アフィン変換パ ラメータ（同ライン Y 方向参照移動量 dy）
0x036	DB_BG3PD	123	2D グラフィックスエンジン B の BG3 アフィン変換パ ラメータ（次ライン Y 方向参照移動量 dmy）
0x038	DB_BG3X	123	2D グラフィックスエンジン B の BG3 参照開始点 （x 座標）
0x03A			
0x03C	DB_BG3Y	123	2D グラフィックスエンジン B の BG3 参照開始点 （y 座標）
0x03E			

※ 0x04001000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x040	DB_WIN0H	154	2D グラフィックスエンジン B のウィンドウ 0H サイズ
0x042	DB_WIN1H	154	2D グラフィックスエンジン B のウィンドウ 1H サイズ
0x044	DB_WIN0V	155	2D グラフィックスエンジン B のウィンドウ 0V サイズ
0x046	DB_WIN1V	155	2D グラフィックスエンジン B のウィンドウ 1V サイズ
0x048	DB_WININ	154	2D グラフィックスエンジン B のウィンドウ内側
0x04A	DB_WINOUT	154	2D グラフィックスエンジン B のウィンドウ外側
0x04C	DB_MOSAIC	160	2D グラフィックスエンジン B のモザイクサイズ
0x04E			
0x050	DB_BLD CNT	157	2D グラフィックスエンジン B のカラー特殊効果
0x052	DB_BLD ALPHA	158	2D グラフィックスエンジン B の α ブレンディング係数
0x054	DB_BLD Y	159	2D グラフィックスエンジン B の輝度変更係数
0x056			
0x058			
0x05A			
0x05C			
0x05E			
0x060			
0x062			
0x064			
0x066			
0x068			
0x06A			
0x06C	DB_MASTER_BRIGHT	90	画像出力 B のマスター輝度

※ 0x04001000 からのオフセット値です。

A.3 0x04004000 番地以降（TWL 拡張関連）

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x000	SCFG_A9ROM	22	ROM ステータス
0x002			
0x004	SCFG_CLK	23	新規ブロッククロック制御
0x006	SCFG_RST	23	新規ブロックリセット制御
0x008	SCFG_EXT	24	拡張機能制御
0x00A			
0x00C			
0x00E			
0x010	SCFG_MC		メモリーカード IF ステータス
0x012			
0x014			
0x016			
0x018			
0x01A			
0x01C			
0x01E			
0x020			
0x022			
0x024			
0x026			
0x028			
0x02A			
0x02C			
0x02E			
0x030			
0x032			
0x034			
0x036			
0x038			
0x03A			
0x03C			
0x03E			
0x040	MBK1	49	WRAM バンクコントロールレジスタ 1
0x042			
0x044	MBK2	51	WRAM バンクコントロールレジスタ 2
0x046			
0x048	MBK3	52	WRAM バンクコントロールレジスタ 3
0x04A			
0x04C	MBK4	54	WRAM バンクコントロールレジスタ 4
0x04E			
0x050	MBK5	55	WRAM バンクコントロールレジスタ 5
0x052			
0x054	MBK6	50	WRAM バンクコントロールレジスタ 6
0x056			
0x058	MBK7	53	WRAM バンクコントロールレジスタ 7
0x05A			
0x05C	MBK8	56	WRAM バンクコントロールレジスタ 8
0x05E			
0x060	MBK9		WRAM バンクコントロールレジスタ 9
0x062			

※ 0x04004000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x064			
0x066			
0x068			
0x06A			
0x06C			
0x06E			
0x070			
0x072			
0x074			
0x076			
0x078			
0x07A			
0x07C			
0x07E			
0x080			
0x082			
0x084			
0x086			
0x088			
0x08A			
0x08C			
0x08E			
0x090			
0x092			
0x094			
0x096			
0x098			
0x09A			
0x09C			
0x09E			
0x0A0			
0x0A2			
0x0A4			
0x0A6			
0x0A8			
0x0AA			
0x0AC			
0x0AE			
0x0B0			
0x0B2			
0x0B4			
0x0B6			
0x0B8			
0x0BA			
0x0BC			
0x0BE			
0x0C0			
0x0C2			
0x0C4			
0x0C6			
0x0C8			
0x0CA			
0x0CC			
0x0CE			

※ 0x04004000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x0D0			
0x0D2			
0x0D4			
0x0D6			
0x0D8			
0x0DA			
0x0DC			
0x0DE			
0x0E0			
0x0E2			
0x0E4			
0x0E6			
0x0E8			
0x0EA			
0x0EC			
0x0EE			
0x0F0			
0x0F2			
0x0F4			
0x0F6			
0x0F8			
0x0FA			
0x0FC			
0x0FE			
0x100	NDMAGCNT	271	NDMA グローバルコントロール
0x102			
0x104	NDMA0SAD	272	NDMA0 ソースアドレス
0x106			
0x108	NDMA0DAD	272	NDMA0 デスティネーションアドレス
0x10A			
0x10C	NDMA0TCNT	272	NDMA0 総転送ワードカウンタ
0x10E			
0x110	NDMA0WCNT	272	NDMA0 ワードカウンタ
0x112			
0x114	NDMA0BCNT	272	NDMA0 ブロック転送インターバル
0x116			
0x118	NDMA0FDATA	273	NDMA0 フィルデータ
0x11A			
0x11C	NDMA0CNT	273	NDMA0 コントロール
0x11E			
0x120	NDMA1SAD	272	NDMA1 ソースアドレス
0x122			
0x124	NDMA1DAD	272	NDMA1 デスティネーションアドレス
0x126			
0x128	NDMA1TCNT	272	NDMA1 総転送ワードカウンタ
0x12A			
0x12C	NDMA1WCNT	272	NDMA1 ワードカウンタ
0x12E			
0x130	NDMA1BCNT	272	NDMA1 ブロック転送インターバル
0x132			
0x134	NDMA1FDATA	273	NDMA1 フィルデータ
0x136			

※ 0x04004000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x138	NDMA1CNT	273	NDMA1 コントロール
0x13A			
0x13C	NDMA2SAD	272	NDMA2 ソースアドレス
0x13E			
0x140	NDMA2DAD	272	NDMA2 デスティネーションアドレス
0x142			
0x144	NDMA2TCNT	272	NDMA2 総転送ワードカウンタ
0x146			
0x148	NDMA2WCNT	272	NDMA2 ワードカウンタ
0x14A			
0x14C	NDMA2BCNT	272	NDMA2 ブロック転送インターバル
0x14E			
0x150	NDMA2FDATA	273	NDMA2 フィルデータ
0x152			
0x154	NDMA2CNT	273	NDMA2 コントロール
0x156			
0x158	NDMA3SAD	272	NDMA3 ソースアドレス
0x15A			
0x15C	NDMA3DAD	272	NDMA3 デスティネーションアドレス
0x15E			
0x160	NDMA3TCNT	272	NDMA3 総転送ワードカウンタ
0x162			
0x164	NDMA3WCNT	272	NDMA3 ワードカウンタ
0x166			
0x168	NDMA3BCNT	272	NDMA3 ブロック転送インターバル
0x16A			
0x16C	NDMA3FDATA	273	NDMA3 フィルデータ
0x16E			
0x170	NDMA3CNT	273	NDMA3 コントロール
0x172			
0x174			
0x176			
0x178			
0x17A			
0x17C			
0x17E			
0x180			
0x182			
0x184			
0x186			
0x188			
0x18A			
0x18C			
0x18E			
0x190			
0x192			
0x194			
0x196			
0x198			
0x19A			

※ 0x04004000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x19C			
0x19E			
0x1A0			
0x1A2			
0x1A4			
0x1A6			
0x1A8			
0x1AA			
0x1AC			
0x1AE			
0x1B0			
0x1B2			
0x1B4			
0x1B6			
0x1B8			
0x1BA			
0x1BC			
0x1BE			
0x1C0			
0x1C2			
0x1C4			
0x1C6			
0x1C8			
0x1CA			
0x1CC			
0x1CE			
0x1D0			
0x1D2			
0x1D4			
0x1D6			
0x1D8			
0x1DA			
0x1DC			
0x1DE			
0x1E0			
0x1E2			
0x1E4			
0x1E6			
0x1E8			
0x1EA			
0x1EC			
0x1EE			
0x1F0			
0x1F2			
0x1F4			
0x1F6			
0x1F8			
0x1FA			
0x1FC			
0x1FE			

※ 0x04004000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x200	CAM_MCNT		カメラモジュールコントロール
0x202	CAM_CNT		カメラコントロール
0x204	CAM_DAT		カメラデータ
0x206			
0x208			
0x20A			
0x20C			
0x20E			
0x210	CAM_SOFS		カメラ トリミング開始位置設定
0x212			
0x214	CAM_EOFS		カメラ トリミング終了位置設定
0x216			
0x218			
0x21A			
0x21C			
0x21E			
0x220			
0x222			
0x224			
0x226			
0x228			
0x22A			
0x22C			
0x22E			
0x230			
0x232			
0x234			
0x236			
0x238			
0x23A			
0x23C			
0x23E			
0x240			
0x242			
0x244			
0x246			
0x248			
0x24A			
0x24C			
0x24E			
0x250			
0x252			
0x254			
0x256			
0x258			
0x25A			
0x25C			
0x25E			
0x260			
0x262			
0x264			
0x266			

※ 0x04004000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x268			
0x26A			
0x26C			
0x26E			
0x270			
0x272			
0x274			
0x276			
0x278			
0x27A			
0x27C			
0x27E			
0x280			
0x282			
0x284			
0x286			
0x288			
0x28A			
0x28C			
0x28E			
0x290			
0x292			
0x294			
0x296			
0x298			
0x29A			
0x29C			
0x29E			
0x2A0			
0x2A2			
0x2A4			
0x2A6			
0x2A8			
0x2AA			
0x2AC			
0x2AE			
0x2B0			
0x2B2			
0x2B4			
0x2B6			
0x2B8			
0x2BA			
0x2BC			
0x2BE			
0x2C0			
0x2C2			
0x2C4			
0x2C6			
0x2C8			
0x2CA			
0x2CC			
0x2CE			

※ 0x04004000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x2D0			
0x2D2			
0x2D4			
0x2D6			
0x2D8			
0x2DA			
0x2DC			
0x2DE			
0x2E0			
0x2E2			
0x2E4			
0x2E6			
0x2E8			
0x2EA			
0x2EC			
0x2EE			
0x2F0			
0x2F2			
0x2F4			
0x2F6			
0x2F8			
0x2FA			
0x2FC			
0x2FE			
0x300	DSP_PDATA	273	DSP 転送データ
0x302			
0x304	DSP_PADR	329	DSP 転送アドレス
0x306			
0x308	DSP_PCFG	327	DSP コンフィギュレーション
0x30A			
0x30C	DSP_PSTS	328	DSP ステータス
0x30E			
0x310	DSP_PSEM	331	DSP セマフォ
0x312			
0x314	DSP_PMASK	331	DSP セマフォマスク
0x316			
0x318	DSP_PCLEAR	273	DSP セマフォクリア
0x31A			
0x31C	DSP_SEM	331	DSP セマフォデータ
0x31E			
0x320	DSP_CMD0	330	DSP コマンドレジスタ 0
0x322			
0x324	DSP_REP0	330	DSP リプライレジスタ 0
0x326			
0x328	DSP_CMD1	330	DSP コマンドレジスタ 1
0x32A			
0x32C	DSP_REP1	330	DSP リプライレジスタ 1
0x32E			
0x330	DSP_CMD2	330	DSP コマンドレジスタ 2
0x332			
0x334	DSP_REP2	330	DSP リプライレジスタ 2
0x336			

※ 0x04004000 からのオフセット値です。

アドレス※ オフセット	ARM9 レジスタ名	ページ	説明
0x338			
0x33A			
0x33C			
0x33E			
0x340			
0x342			
0x344			
0x346			
0x348			
0x34A			
0x34C			
0x34E			
0x350			
0x352			
0x354			
0x356			
0x358			
0x35A			
0x35C			
0x35E			
0x360			
0x362			
0x364			
0x366			
0x368			
0x36A			
0x36C			
0x36E			
0x370			
0x372			
0x374			
0x376			
0x378			
0x37A			
0x37C			
0x37E			

※ 0x04004000 からのオフセット値です。

付録B. VRAM データ容量一覧表

(データ容量の単位：バイト)

フォーマット \ サイズ	8 x 8	16 x 16	32 x 32	64 x 64	128 x 128	256 x 192	256 x 256	512 x 256	512 x 512	1024 x 512	1024 x 1024
16 色キャラクタ	32	128	512	2K	×	×	×	×	×	×	×
256 色キャラクタ	64	256	1K	4K	×	×	×	×	×	×	×
ダイレクトカラービットマップ OBJ	128	512	2K	8K	×	×	×	×	×	×	×
通常キャラクタ BG スクリーン	×	×	×	×	×	×	2K	4K	8K	×	×
回転キャラクタ BG スクリーン	×	×	×	×	256	×	1 K	×	4K	×	16K
拡張回転キャラクタ BG スクリーン	×	×	×	×	512	×	2K	×	8K	×	32K
256 色ビットマップ BG	×	×	×	×	16K	×	64K	128K	256K	×	×
大画面 256 色ビットマップ BG	×	×	×	×	×	×	×	×	×	512K	×
ダイレクトカラービットマップ BG	×	×	×	×	32K	×	128K	256K	512K	×	×
クリアカラーイメージ	×	×	×	×	×	96K	×	×	×	×	×
クリアデプスイメージ	×	×	×	×	×	96K	×	×	×	×	×
表示キャプチャ	×	×	×	×	32K	96K	×	×	×	×	×
4 色テクスチャ	16	64	256	1K	4K	—	16K	—	64K	—	256K
16 色テクスチャ	32	128	512	2K	8K	—	32K	—	128K	—	512K
256 色テクスチャ	64	256	1K	4K	16K	—	64K	—	256K	512K	×
A3I5 半透明テクスチャ	64	256	1K	4K	16K	—	64K	—	256K	512K	×
A5I3 半透明テクスチャ	64	256	1K	4K	16K	—	64K	—	256K	512K	×
ダイレクトカラーテクスチャ	128	512	2K	8K	32K	—	128K	—	512K	×	×
圧縮テクスチャイメージ	16	64	256	1K	4K	—	16K	—	64K	—	256K
圧縮テクスチャインデックス	8	32	128	512	2K	—	8K	—	32K	—	128K
圧縮テクスチャ補間パレット最大数	16	64	256	1K	4K	—	16K	—	64K	(96K)	(96K)

× : 仕様外

— : 省略

太字: 最大値

括弧: データの最大値では RAM 容量を超えてしまうため、使用可能な RAM 容量の最大値を記載

付録C. データフォーマット

BG, OBJ キャラクタデータ

d フォーマット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16 色キャラクタ	P3				P2				P1				P0			
256 色キャラクタ	P1								P0							

ビットマップ OBJ

フォーマット \ d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ダイレクトカラー ビットマップ OBJ	ALP HA	BLUE				GREEN				RED						

BG スクリーンデータ

フォーマット \ d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
通常キャラクタ BG スクリーン	パレット				Vflip	Hflip	キャラクタネーム									
回転キャラクタ BG スクリーン										キャラクタネーム						
拡張回転キャラクタ BG スクリーン	パレット				Vflip	Hflip	キャラクタネーム									

ビットマップ BG データ

フォーマット \ d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
256 色ビットマップ BG									カラーNo.							
大画面 256 色 ビットマップ BG									カラーNo.							
ダイレクトカラー ビットマップ BG	ALP HA	BLUE				GREEN				RED						

パレットデータ

フォーマット \ d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BG, OBJ パレットカラー	GR EEN	BLUE				GREEN				RED						

※ 15 ビット目は GREEN の最下位ビットとして使用されます。BLUE と RED の最下位ビットは 0 で拡張されます。

その他グラフィックス機能のデータ

フォーマット \ d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
クリアカラーイメージ	ALP HA	BLUE					GREEN					RED				
クリアデプスイメージ	FOG	整数部												小数部		
		クリアデプス														
表示キャプチャ	ALP HA	BLUE					GREEN					RED				

テキストチャデータ

フォーマット	d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4色テキストチャ																	T0
16色テキストチャ																	T0
256色テキストチャ																	T0
A5I3 半透明テキストチャ														ALPHA		INDEX	
A3I5 半透明テキストチャ														ALPHA		INDEX	
ダイレクトカラーテキストチャ	ALPHA																RED

圧縮テキストチャデータ (注: 32 ビット表記)

フォーマット ^d	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
圧縮テキストチャ イメージ	T33		T32		T31		T30		T23		T22		T21		T20		T13		T12		T11		T10		T03		T02		T01		T00			
圧縮テキストチャ インデックス																	3/4	T	パレットアドレス															

テキストチャパレットデータ

フォーマット	d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
テキストチャパレットカラー		—															RED

OAM データ

フォーマット	d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OBJ アトリビュート 0			形状	カラーモード	モザイク	OBJ モード		倍角	回転	拡大縮小							Y 座標
OBJ アトリビュート 1			サイズ	Vflip	Hflip												X 座標
OBJ アトリビュート 2																	パレット No.
アフィン変換パラメータ PA																	優先順位
																	キャラクターネーム
アフィン変換パラメータ PB																	整数部
																	小数部
アフィン変換パラメータ PC																	同ライン X 方向移動量
																	次ライン X 方向移動量
アフィン変換パラメータ PD																	同ライン Y 方向移動量
																	次ライン Y 方向移動量

サウンドデータ

フォーマット	d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCM 8ビット																	データ 0
PCM16ビット																	データ 0
ADPCM																	データ 0

記載されている会社名、製品名等は、各社の登録商標または商標です。

©2004 – 2012 Nintendo

任天堂株式会社の許諾を得ることなく、本書に記載されている内容の一部あるいは全部を無断で複製・複写・転写・頒布・貸与することを禁じます。