参 考 資 料

TEXAS INSTRUMENTS
www.tij.co.jp

JAJA154– 2月2008年

TMS320DM643x ブートローダの使用方法

アプリケーション技術部

アブストラクト

本資料は、DM643x ROMブートローダの機能について説明します。ROMブートローダはコード/データをロードするための主要なフォーマットとしてアプリケーション・イメージ・スクリプトを使用することに注意してください。AISはTexas Instruments Inc独自のデータ・フォーマットです。AISは本資料の3 項で説明されています。

本資料はhttp://www-s.ti.com/sc/techlit/spraag0.zip からダウンロード可能なプロジェクト・コードを使用しています。

目次

図

TEXAS INSTRUMENTS

SPRAAG0C 翻訳版

最新の英語版資料
http://www-s.ti.com/sc/techlit/spraag0c.pdf

表

1 はじめに

ROMブートローダはROMアドレス0x00100000から始まるデバイス内のROMに配置されています。ROMブート・ローダ(RBL)は下記に記されたモードでブートを行うために実装されています。RBLはブート・モードを決定するために、BOOTCFGレジスタの内容をリードし、デバイスのブートを行うために適切な命令を実行します。もし、不適当なブート・モードが選択された場合、またはブート中にスレーブ・デバイスからブートを行っている間に何らかの理由でエラーが発生した場合、RBLはデフォルトのブート・デバイスとしてUARTを使って通信を行います。

マスタ・モードでブートを行っているとき、ブート・ローダは必要に応じてスレーブ・デバイスからブート情報をリードします。スレーブ・モードでブートしているとき、ブートローダはマスタ・デバイスによっては必要に応じてブート情報を送ります。すべてのブート・モードにおいて、ブート中はウオッチドッグ・タイマがディスエーブルされることに注意してください。すべてのアプリケーションはブート処理中にウオッチドッグ・タイマの設定を行ってはいけません。(AISコマンドまたはコードはブート中に設定をすべきではありません) 表 1 はこのアプリケーション・レポートで使われる用語及び略語のリストを示します。

- エミュレーション･ブート
- HPI
- PCI (DSPはスレーブ)
- EMIFA ROM ダイレクト･ブート
- AIS有り EMIFA ROM 高速ブート
- AIS無し EMIFA ROM 高速ブート
- NAND
- I2C (DSPはマスタ)
- SPI 16×8 (DSPはマスタ、SPIオペレーション毎に16ビット･アドレスを送信、64K×8 デバイスまでサポート)
- SPI 24×8 (DSPはマスタ、SPIオペレーション毎に24ビット･アドレスを送信、16M×8 デバイスまでサポート)
- フロー制御無し UART (DSPはスレーブ)
- フロー制御有り UART (DSPはスレーブ)
- VLYNQ （DSPはスレーブ)

表 1 用語と略語

用語	詳細
ブートローダ	ROM DM643xブートローダのためのソフトウェア/コード
AIS	アプリケーション･イメージ･スクリプト
BL	ブートローダ (本資料ではブートローダを指します)
DSP	デジタル･シグナル･プロセッサ (本資料ではDM643xを指します)
EMIF	外部メモリ･インターフェース
GPIO	汎用入力/出力
HPI	ホストポート･インターフェース
I2C	インター･インテグレーティッド･サーキット
NAND	否定論理積ゲート
OFD	オブジェクト･ファイル･ディスプレイ
PCI	ペリフェラル･コンポーネント･インターコネクト
ROM	リード･オンリー･メモリ
SPI	シリアル･ペリフェラル･インターフェース
SRGR	サンプル･レート･ジェネレータ･コントロール･レジスタ
UART	ユニバーサル･アシンクロナス･レシーバ/トランシーバ

2 ブート･モード概要

次に示すブート･モードは、ブート･デバイス･ピンの状態によって選択されます。これらのピンの状態は /PORリセットの立ち上がりエッジでキャプチャされBOOTCFGレジスタに格納されます。ブート･ローダBOOTCFGレジスタの内容をリードし、選択されたブートを実行するために適切なコードへ分岐します。
ブート･モードは3つのカテゴリに分類されます。これらは、非高速ブート、固定逓倍高速ブートおよびユーザ選択逓倍高速ブートです。

- **非高速ブート(FATBOOT=0):** デバイスはブート中にデフォルトのフェーズ･ロック･ループ(PLL) バイパス･モードで動作します。非高速ブートについて表 2 にまとめます。
- **固定逓倍高速ブート (FASTBOOT=1, AEM[2:0]=001b):** ブート･ローダ･コードはブート中に固定PLL逓倍数を使って高速に動作します。固定逓倍高速ブート表 3 にまとめます。
  注意: AEM[2:0]の時、EMIFAのアドレス･ポインタを選択するAEAW[2:0]の3 ピンと同様に、PLLMS[2:0]の設定は固定逓倍高速ブートでは影響を与えません。
- **ユーザ選択逓倍高速ブート (FASBOOT=1, AEM[2:0]=000b, 011b, 100b, 101b):** ブートローダはブート中に高速動作します。PLLの逓倍数はPLLMS[2:0]を使ってユーザにより選択されます。ユーザ選択固定逓倍高速ブートを表 4 にまとめます。

もし無効なブート･モードが選択された場合、ブート･ローダはBOOTCMPLTレジスタのERRフィールドにエラー･コードをライトし、その後すべての非ホスト･ブート･モード(たとえばI2C、SPI) に対してはUARTブートをデフォルトで行います。
ブート･デバイス･ピンは有効なモードのどれかひとつに設定されなければなりません。それぞれの有効なモードについては後述します。
上述の表に記載されていないすべてのモードは予約であり無効な設定です。

表 2 非高速ブート (FASTBOOT=0)

デバイス･ブート及び設定ピン		ブート詳細 (1)	DM643x DMP (マスタ/スレーブ)	ブート時のPLLC1クロック設定			DSPBOOTADDR (デフォルト) (1)
BOOTMODE[3:0]	PCIEN			PLL モード(2)	CLKDIV1ドメイン (SYSCLK1分周器)	デバイス周波数 (SYSCLK1)	
0000	0又は1	ブートなし (エミュレーション･ブート)	マスタ	バイパス	/1	CLKIN	0x0010 0000
0001	0又は1	予約	-	-	-	-	-
0010	0	HPIブート	スレーブ	バイパス	/1	CLKIN	0x0010 0000
	1	予約	-	-	-	-	-
0011	0又は1	予約	-	-	-	-	-
0100	0又は1	EMIFA ROMダイレクト･ブート [PLLバイパス･モード]	マスタ	バイパス	/1	CLKIN	0x4200 0000
0101	0又は1	I2Cブート [スタンダード･モード] (3)	マスタ	バイパス	/1	CLKIN	0x0010 0000
0110	0又は1	16ビットSPIブート [McBSP0]	マスタ	バイパス	/1	CLKIN	0x0010 0000
0111	0又は1	NANDフラッシュ･ブート	マスタ	バイパス	/1	CLKIN	0x0010 0000
1000	0又は1	ハードウェア･フロー制御なしUARTブート [UART0]	マスタ	バイパス	/1	CLKIN	0x0010 0000
1001	0又は1	予約	-	-	-	-	-
1010	0又は1	VLYNQブート	スレーブ	バイパス	/1	CLKIN	0x0010 0000
1011	0又は1	予約	-	-	-	-	-
1100	0又は1	予約	-	-	-	-	-
1101	0又は1	予約	-	-	-	-	-
1110	0又は1	ハードウェア･フロー制御ありUARTブート [UART0]	マスタ	バイパス	/1	CLKIN	0x0010 0000
1111	0又は1	24ビットSPIブート (McBSP0+GP[97]]	マスタ	バイパス	/1	CLKIN	0x0010 0000

(1) DSPBOOTADDR=0x0010 0000 をデフォルトとするブート･モード(つまりEMIFA ROMダイレクト･ブート,BOOTMODE[3:0]=0100,FASTBOOT=0を除いたすべてのモード) において、ブートローダはすべてのC64x+キャッシュ(L2、L1P及びL1D)をディスエーブルにします。そのため、ブート･ローダ･コードから抜けるにあたり、すべてのC64x+メモリはRAMに設定されます。もしキャッシュを使う必要があれば、アプリケーション･コードで明示的にキャッシュをイネーブルにする必要があります。

(2) この表に示されるように、非高速ブートのPLLモードは固定です。そのため、PLLMS[2:0]設定ピンはPLLモードに影響を与えません。

(3) I2Cブート(BOOTMODE[3:0]=0101b)は、MXI/CLKINの周波数が21MHzから30MHzの時のみ使用可能です。I2CブートはMXI/CLKINの周波数が21MHz未満の時は使用できません。

表 3 固定逓倍高速ブート (FASTBOOT = 1, AEM[2:0] = 001b)

デバイス·ブート及び設定ピン		ブート詳細 (1)	DM643x DMP (マスタ/スレーブ)	ブート時のPLLC1クロック設定			DSPBOOTADDR (デフォルト) (1)
BOOTMODE[3:0]	PCIEN			PLL モード(2)	CLKDIV1ドメイン (SYSCLK1分周器)	デバイス周波数 (SYSCLK1)	
0000	0又は1	ブートなし（エミュレーション·ブート)	マスタ	バイパス	/1	CLKIN	0x0010 0000
0001	0	PLL逓倍数×27の HPIブート	スレーブ	×27	/2	CLKIN ×27 /2	0x0010 0000
	1	予約	-	-	-	-	-
0010	0	PLL逓倍数×20の HPIブート	スレーブ	×20	/2	CLKIN ×20 /2	0x0010 0000
	1	予約	-	-	-	-	-
0011	0	PLL逓倍数×15の HPIブート	スレーブ	×15	/2	CLKIN ×15 /2	0x0010 0000
	1	予約	-	-	-	-	-
0100	0又は1	AISあり EMIFA ROM高速ブート	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000
0101	0又は1	I2Cブート [FASTBOOT] (3)	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000
0110	0又は1	16ビットSPIブート [McBSP0]	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000
0111	0又は1	NANDフラッシュ·ブート	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000
1000	0又は1	ハードウェア·フロー制御なしUARTブート [UART0]	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000
1001	0又は1	AISなし EMIFA ROM高速ブート	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000
1010	0又は1	VLYNQブート	スレーブ	×20	/2	CLKIN ×20 /2	0x0010 0000
1011	0又は1	予約	-	-	-	-	-
1100	0又は1	予約	-	-	-	-	-
1101	0又は1	予約	-	-	-	-	-
1110	0又は1	ハードウェア·フロー制御ありUARTブート [UART0]	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000
1111	0又は1	24ビットSPIブート (McBSP0+GP[97]]	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000

(1) DSPBOOTADDR=0x0010 0000 をデフォルトとするブート·モードにおいて、ブートローダはすべてのC64x+キャッシュ(L2、L1P及びL1D)をディスエーブルにします。そのため、ブート·ローダ·コードから抜けるにあたり、すべてのC64x+メモリはRAMに設定されます。もしキャッシュを使う必要があれば、アプリケーション·コードで明示的にキャッシュをイネーブルにする必要があります。

(2) この表に示されるように、固定逓倍高速ブートのPLLモードは固定です。そのため、PLLMS[2:0]設定ピンはPLLモードに影響を与えません。

(3) I2Cブート(BOOTMODE[3:0]=0101b)は、MXI/CLKINの周波数が21MHzから30MHzの時のみ使用可能です。I2CブートはMXI/CLKINの周波数が21MHz未満の時は使用できません。

表 4 ユーザ選択逓倍高速ブート (FASTBOOT = 1, AEM[2:0] = 000b, 011b, 100b, 又は 101b)

デバイス･ブート及び設定ピン		ブート詳細 (1)	DM643x DMP (マスタ/スレーブ)	ブート時のPLLC1クロック設定			DSPBOOTADDR (デフォルト) (1)
BOOTMODE[3:0]	PCIEN			PLL モード(2)	CLKDIV1ドメイン (SYSCLK1分周器)	デバイス周波数 (SYSCLK1)	
0000	0又は1	ブートなし（エミュレーション･ブート)	マスタ	バイパス	/1	CLKIN	0x0010 0000
0001	0	予約	-	-	-	-	-
	1	自動初期化なしPCIブート	スレーブ	表 5	/2	表 5	0x0010 0000
0010	0	HPIブート	スレーブ	表 5	/2	表 5	0x0010 0000
	1	自動初期化ありPCIブート	スレーブ	表 5	/2	表 5	0x0010 0000
0011	0又は1	予約	-	-	-	-	-
0100	0又は1	AISありEMIFA ROM高速ブート	マスタ	表 5	/2	表 5	0x0010 0000
0101	0又は1	I2Cブート [高速モード] (3)	マスタ	表 5	/2	表 5	0x0010 0000
0110	0又は1	16ビットSPIブート [McBSP0]	マスタ	表 5	/2	表 5	0x0010 0000
0111	0又は1	NANDフラッシュ･ブート	マスタ	表 5	/2	表 5	0x0010 0000
1000	0又は1	ハードウェア･フロー制御なしUARTブート [UART0]	マスタ	表 5	/2	表 5	0x0010 0000
1001	0又は1	AISなしEMIFA ROM高速ブート	マスタ	表 5	/2	表 5	-
1010	0又は1	VLYNQブート	スレーブ	×20	/2	CLKIN ×20 /2	0x0010 0000
1011	0又は1	予約	-	-	-	-	-
1100	0又は1	予約	-	-	-	-	-
1101	0又は1	予約	-	-	-	-	-
1110	0又は1	ハードウェア･フロー制御ありUARTブート [UART0]	マスタ	表 5	/2	表 5	0x0010 0000
1111	0又は1	24ビットSPIブート (McBSP0+GP[97])	マスタ	×20	/2	CLKIN ×20 /2	0x0010 0000

(1) DSPBOOTADDR=0x0010 0000 をデフォルトとするブート･モードにおいて、ブートローダはすべてのC64x+キャッシュ(L2、L1P及びL1D)をディスエーブルにします。そのため、ブート･ローダ･コードから抜けるにあたり、すべてのC64x+メモリはRAMに設定されます。もしキャッシュを使う必要があれば、アプリケーション･コードで明示的にキャッシュをイネーブルにする必要があります。
(2) サポートされているすべてのPLLモードが使用可能です。[サポートされているDM643x PLLモードは表 5を参照してください]
(3) I2Cブート(BOOTMODE[3:0]=0101b)は、MXI/CLKINの周波数が21MHzから30MHzの時のみ使用可能です。I2CブートはMXI/CLKINの周波数が21MHz未満の時は使用できません。

表 5 ユーザ選択逓倍高速ブート における逓倍数(PLLMS[2:0])の選択
(FASTBOOT = 1; AEM[2:0] = 000b, 011b, 100b, 又は 101b)

デバイス･ブート及び設定ピン	ブート時のPLLC1クロック設定		
PLLMS[2:0]	PLLモード	CLKDIV1ドメイン (SYSCLK1分周器)	デバイス周波数 (SYSCLK1)
000	×20	/2	CLKIN ×20 / 2
001	×15	/2	CLKIN ×15 / 2
010	×16	/2	CLKIN ×16 / 2
011	×18	/2	CLKIN ×18 / 2
100	×22	/2	CLKIN ×22 / 2
101	×25	/2	CLKIN ×25 / 2
110	×27	/2	CLKIN ×27 / 2
111	×30	/2	CLKIN ×30 / 2

2.1 ブートに関する要求事項、制約及びデフォルトの設定

以下の要求事項に注意してください。

- FASTBOOTはすべてのPCIブートに必要です。
- ブート･ローダはI2C EEPROMについては16ビット･アドレス幅のみサポートします。
- 自動初期化付きPCIブートについてI2C EEPROMは必ずデバイスのI2Cに接続されていなければなりません。
- すべてのブート･タイミングは27MHzの入力クロックに対して最適化されていることに注意してください。
- I2C、SPI、UART、NAND及びEMIFA FASTBOOT (BOOTMODE=[3:0]=0100b) ではブートのためのデータをAISフォーマットで格納する必要があります。AISはTexas Instruments Inc.の独自のブート･イメージのフォーマットです。AISについての詳細は本資料の3 項で述べます。HPIやPCIといったホスト･ブート･モードではフォーマットはユーザが決めることができます。
- FASTBOOTが選択された時、ブートローダはPLLを設定します。PLLの逓倍数はリセット時にラッチされBOOTCFGレジスタに格納されるAEMおよびPLLMS[3:0]の値によって決められます。このドキュメントではデバイスへの入力クロックが27MHzとしてすべてのタイミング計算を行っています。FASTBOOTにおけるより広い範囲の動作周波数の詳細についてはデバイスのデータシートを参照してください。
- ROMブートローダは動作時にチップ･セレクトのトグルを必要とするNANDフラッシュをサポートしていません。
- NANDブートではNANDデバイスはEMIFAのCS2に接続してください。
- ブートローダはブート･モードの選択にかかわらずブート処理中にキャッシュをディスエーブルにします。例外のブート･モードは、ブートローダが動作しないダイレクトROMブートで、キャッシュは電源投入時のデフォルト状態になります。
- ブート･ローダはすべてのSPIブート･モードでデータ長が×8ビットのSPI EEPROMをサポートします。ブートローダは8ビット分のクロックしか供給しないため、データ長が×16ビットのデバイスはサポートできません。

2.2 FASTBOOT モード

エミュレーション･ブート･モード(BOOTMODE[3:0]=0000b)を除き、FASTBOOTオプションが選択された時、ブート･ローダ･ソフトウェアはPLLを設定します。PLLの逓倍数はリセット時にラッチされるAEM及びPLLMS[2:0]の値によって決められます。ブートローダはデバイスの/PORリセット時にラッチされBOOTCFGレジスタに格納されたこれらの値をリードし、表 6に従ってPLLの逓倍数を選択します。これらの設定や関連するタイミングについては各デバイスのデータシートを参照してください。

表 6 AEM 及び PLLMS[2:0] ピンを基にしたPLL逓倍数

FASTBOOT	AEM	PLLMS[2:0]	PLLM
1	001	N/A	BOOTMODE[3:0] == 0001 かつPCIEN==0 であれば、PLLM = 26 (CLKIN × 27)
			BOOTMODE[3:0] == 0011 かつPCIEN==0 であれば、PLLM = 14 (CLKIN × 15)
			その他のBOOTMODE[3:0], PCIEN に対しては、PLLM = 19 (CLKIN ´ 20)
1	!=001	000	BOOTMODE[3:0], PCIENにかかわらず、PLLM = 19 (CLKIN × 20)
		001	BOOTMODE[3:0], PCIENにかかわらず、PLLM = 14 (CLKIN × 15)
		010	BOOTMODE[3:0], PCIENにかかわらず、PLLM = 15 (CLKIN × 16)
		011	BOOTMODE[3:0], PCIENにかかわらず、PLLM = 17 (CLKIN × 18)
		100	BOOTMODE[3:0], PCIENにかかわらず、PLLM = 21 (CLKIN × 22)
		101	BOOTMODE[3:0], PCIENにかかわらず、PLLM = 24 (CLKIN × 25)
		110	BOOTMODE[3:0], PCIENにかかわらず、PLLM = 26 (CLKIN × 27)
		111	BOOTMODE[3:0], PCIENにかかわらず、PLLM = 29 (CLKIN × 30)

ブートローダは無効なPLL逓倍数の選択に対してエラー･コードを生成しないことに注意してください。そのため、選択されたPLLの逓倍数がタイミング制約及びブートペリフェラルまたはPLLの動作周波数を超えていないことを必ず確認してください。これらの要求事項については各デバイスのデータシートを参照してください。

表 7 PLL1 及び PLL2の逓倍数の範囲

PLL逓倍数 (PLLM)	最小値	最大値
PLL1逓倍数	×14	×30
PLL2逓倍数	×14	×32

表 8 PLLC1クロック周波数の範囲

クロック信号名		最小値	最大値	単位
MXI/CLKIN (1)		20	30	MHz
PLLOUT	1.2V CVDDの時	400	600	MHz
	1.05V CVDDの時	400	520	MHz
SYSCLK1 (CLKDIV1ドメイン)	-600デバイス		600	MHz
	-500デバイス		500	MHz
	-400デバイス		400	MHz
	-300デバイス		300	MHz

(1) MXI/CLKIN入力クロックは、双方のPLLコントローラ(PLLC1及びPLLC2)で使用されます。

表 9 PLLC2クロック周波数の範囲

クロック信号名		最小値	最大値	単位
MXI/CLKIN (1)		20	30	MHz
PLLOUT	1.2V CVDDの時	400	900	MHz
	1.05V CVDDの時	400	666	MHz
PLL2_SYSCLK1 (DDR2 PHYへ)			333	MHz
PLL2_SYSCLK2 (VPBEへ)			54	MHz

(1) MXI/CLKIN入力クロックは、双方のPLLコントローラ(PLLC1及びPLLC2)で使用されます。

2.2.1 FASTBOOT オプションの CPU 動作周波数

ブート・ローダ・ソフトウェアはCPUクロックを供給するのにPLL分周器を1の設定(2分周)にします。入力周波数を27MHzとした場合のFASTBOOTオプションにおけるPLLMの値に対するCPU周波数を表 10に示します。

表 10 高速ブート中のCPU周波数

PLLM	CPU周波数
19	270 MHz
14	202 MHz
15	216 MHz
17	243 MHz
21	297 MHz
24	337 MHz
26	364 MHz
27	405 MHz

2.3 エミュレーション･ブート(BOOTMODE[3:0]=0000b、FASTBOOT=0 または 1)

このブート･モードではROMブート･ローダ･ソフトウェアはソフトウェア･ループを実行します。エミュレーション･ソフトウェアはコードをダウンロードし、デバイスを制御する必要があります。このモードではすべてのFASTBOOTオプションが無視されます。PLLをバイパス･モードで動作しCPUに27MHzが供給されます。

2.4 HPI ブート (BOOTMODE[3:0]=0001b または 0010b、または 0011b,PCIEN=0,FASTBOOT=0 または 1)

HPIブート･モードではデバイス･ブート･ローダ･ハードウェア･モジュールはROMブートローダ･ソフトウェアの先頭に分岐します。その次にROMブートローダ･コードは次の順序で処理を行います。

1. FASTBOOT=1の時、ブートローダは表 6 で説明されたようにリセット時にラッチされるPLL逓倍器の設定に基づいてPLLを設定します。
2. 必要とされるHPIレジスタを設定します。
3. DSPBOOTADDR レジスタをクリアします。BOOTCMPLT レジスタのブート･エラー･コード･フィールド(BOOTCMPLT.ERR)および完了ビット(BOOTCMPLT.BC)をクリアします。
4. DSPが起動し、コードのダウンロードの準備が整ったことを知らせるために、ホスト･デバイスにHINTを発行します。
5. BOOTCMPLT.BCレジスタに0以外の値が書き込まれるのを待つために、ソフトウェア･ループに入ります。
6. アプリケーションがダウンロードしたら、ホストはアプリケーションのスタート･アドレスをDSPBOOTADDRレジスタに書き込みます。その次にBOOTCMPLTレジスタのブート完了ビットにセットします。
7. 一旦BOOTCMPLT.BCがホストによりセットされると、ROMブート･ローダ･ソフトウェアはホストによりDSPBOOTADDRにセットされたアドレスに分岐します。

2.5 PCI ブート (BOOTMODE[3:0]＝0001b または 0010b,PCIEN=1,FASTBOOT=1)

DM643xはDSPがPCIスレーブの時のみPCIブートをサポートしています。ブートローダは自動初期化有り、無し双方をサポートしています。自動初期化有りPCIブートが選択された時、ブート･ローダはデバイスのI2Cに接続されたI2C EEPROMに格納された自動初期化データを必要とします。ブートローダはFASTBOOTがイネーブルでない時にもブートを行おうとしますが、これはPCIブートで推奨されたモードでないことに注意してください。PCIタイミング要求にミートするために、どのPCIブートにおいてもFASTBOOTをイネーブルにしてください。
自動初期化無しのPCIブートでは、ROMブートローダは次のステップを実行します。

1. ブートローダは表 6 に従ってAEMおよびPLLMS[2:0]の値を元にPLL逓倍器を設定します。(FASTBOOTが選択されていない場合においてもブートローダはブートを完了しようとすることに注意してください。しかし、PCIの動作周波数は33MHzのPCI要求にミートしません。)
2. DSPBOOTADDRおよびBOOTCMPLTレジスタをクリアします。
3. ブート･モードが0001bの時、ROMブート･ローダはPCIコンフィグレーション･ダン･レジスタ(PCICFGDONE)のPCI CONFIG_DONEビットおよびPCIスレーブ･コントロール･レジスタ(PCISLVCNTL)に1をセットします。ブート･モードが0010Bの時、PCIの自動初期化モードがイネーブルであり、ROMブートローダはPCIラッパー･レジスタにCONFIG_DONE=1になるように設定します。
4. その次にブート･ローダはBOOTCMPLTレジスタをポーリングするためにソフトウェア･ループに入ります。一旦ブート完了が認識されると、ROMブート･ローダ･ソフトウェアはホストによりDSPBOOTADDRにセットされたアドレスに分岐します。

初めに書いたとおりPCIブートに沿ってFASTBOOTが選択された時、ROMブートローダ･ソフトウェアはDSPBOOTADDR及びBOOTCMPLTレジスタをクリアする前にPLLを設定します。

自動初期化有りPCIブートが選択された時、ブート・ローダはI2C EEPROMに格納されたPCI設定データをリードします。I2C EEPROMに格納されたデータはI2C EEPROMのアドレス0x400から始まり、表 11に示すようにフォーマットされている必要があります。

表 11 自動初期化に対する PCI設定データ

バイト・アドレス	内容
0x400	ベンダ ID [15:8]
0x401	ベンダID [7:0]
0x402	デバイスID [15:8]
0x403	デバイスID [7:0]
0x404	クラス・コード [7:0]
0x405	リビジョン ID [7:0]
0x406	クラス・コード [23:16]
0x407	クラス・コード [15:8]
0x408	サブシステム・ベンダ ID [15:8]
0x409	サブシステム・ベンダ ID [7:0]
0x40a	サブシステムID [15:8]
0x40b	サブシステム ID [7:0]
0x40c	Max_Latency
0x40d	Min_Grant
0x40e	予約 (use 0x00)
0x40f	予約(use 0x00)
0x410	予約(use 0x00)
0x411	予約(use 0x00)
0x412	予約(use 0x00)
0x413	予約(use 0x00)
0x414	予約(use 0x00)
0x415	予約(use 0x00)
0x416	予約(use 0x00)
0x417	予約(use 0x00)
0x418	予約(use 0x00)
0x419	予約(use 0x00)
0x41a	チェックサム [15:8]
0x41b	チェックサム [7:0]

PCI初期化データはビッグ・エンディアン・フォーマットで格納されていることが想定されています。

2.6 EMIFA ROM ダイレクト・ブート(BOOTMODE[3:0]=0100b, FASTBOOT=0)

EMIFAダイレクト・ブートはROMブート・ローダを必要としません。DSPハードウェア・ブート・モジュールが直接アドレス0x42000000番地に分岐します。

2.7 AIS 無し EMIFA ROM 高速ブート (BOOTMODE[3:0]=1001b, FASTBOOT=1)

このブート・モードではROMブート・ローダはリセット時にラッチされたAEMおよびPLLMS[2:0]の値を基にPLLを設定します。その次にアドレス0x42000000番地に分岐します。このブート・モードはPLLが設定されることを除いてEMIFAダイレクト・ブート(BOOTMODE[3:0]=0100b, FASTBOOT=0)と事実上同じ動作をします。このモードは外部デバイスからすばやくブートするためにEMIFクロックを高速にします。

2.8 AIS 有り EMIFA ROM 高速ブート (BOOTMODE[3:0]=0100b, FASTBOOT=1)

EMIFA 高速ROMブート・モードではDSPハードウェア・ブート・モジュールはROMブートローダ・ソフトウェアに制御を転送します。このブート・モードはEMIFAダイレクト・ブートと異なる動作をします。ROMブート・ローダはブート・プロセスを制御し、初めにより速いCPUスピードで動作するためにPLLを設定し、その次にEMIFAのアドレス0x42000000番地から始まるコード/データを読みます。FLASH/ROMに格納されているデータはAISフォーマットでなければなりません。AISの詳細は本資料の3項で述べられています。AISブート・イメージはAIS命令及びアプリケーション・コードをDSPメモリにロードするのに必要なデータから成っています。AISフォーマットを使用することにより、コードをロードするためのユーザ独自のセカンダリ・ブート・ローダは必要とされません。ROMブート・ローダはAISのJUMP_CLOSE命令が現われるまでEMIFA ROMからのAISコマンドを処理します。JUMP_CLOSE命令は、アプリケーション・コードのスタート・アドレスを内包しています。この命令は、ブートするにあたってアプリケーションが完全にロードされ、すべてのAISコマンドが実行されたことを示します。ROMブートローダはその内部状態クリアし、アプリケーション・コードの先頭に分岐します。EMIF 高速ブートの順序は以下のとおりです。

1. 表 6に示すようにAEMおよびPLLMS[2:0]ピンの値によりPLL逓倍数を使ってPLLを設定します。
2. ラッチされてBOOTCFGレジスタに格納される8_16ピンをリードし、それに従ってEMIFのデータ幅を設定します。
3. AISデータを外部メモリからフェッチし、JUMP_CLOSE命令が現われるまでAIS命令を実行します。
4. JUMP_CLOSEコマンドで与えられるアプリケーション開始アドレスに分岐します。

2.9 I2C マスタ・モード・ブート(BOOTMODE[3:0] = 0101b, FASTBOOT = 0 または 1)

DM643xはDSPがI2Cマスタの場合のみI2Cブートをサポートします。DSPハードウェア・ブート・モジュールはデバイスリセット時にROMブートローダに制御を送ります。ROMブートローダはI2Cペリフェラル・デバイスを設定し、I2CEEPROMからデータのリードを開始します。I2C EEPROMに格納されたデータはAISフォーマットであることが想定されています。始めの32ビットはブート・ローダにより無視されます。この領域は現在予約です。2番目の32ビット・ワードはAISのマジック・ナンバーを含んでいなければなりません。残りのI2C EEPROM内のデータはAISフォーマットでなければなりません。AISの詳細については3項を参照してください。I2Cブート順序は次のとおりです。

1. FASTBOOT=1のとき、ブートローダは表 6に表に示すようにAEMおよびPLLMS[2:0]ピンの値によりPLL逓倍数を使ってPLLを設定します。
2. I2Cをマスタ・モードに設定し、スレーブ・アドレス・レジスタに0x50を設定します。また、オウン・アドレス・レジスタに0x29をセットします。
3. JUMP_CLOSE命令が現われるまで各AIS命令を処理します。
4. アプリケーションの開始アドレスに分岐します。
5. もしI2Cのブート処理でエラーが発生した場合、ブートローダはBOOTCMPLTレジスタのERRフィールドにエラー状況をライトします。その後、UARTを使ってブートを試みます。

2.9.1 I2C マスタ･ブート･タイミング

ブート･ローダはI2Cクロック･プリスケール及びクロック･ロウ･ホールド/クロック･ハイ･ホールド･レジスタに次の値を設定します。

表 12 I2Cタイミング･レジスタの設定

FASTBOOT	レジスタ	値
0	IPSC	0x2
	ICCLKH	0x2D
	ICLKL	0x2D
1	IPSC	0x2
	ICCLKH	0x8
	ICCLKL	0x8

I2Cマスタ･クロックは次の式から導かれます。

$$\text{I2Cマスタ･クロック周波数} = \frac{\text{I2Cペリフェラル･クロック周波数}}{(\text{IPSC} + 1) * [(\text{ICCLKL} + \text{D}) + (\text{ICCLKH} + \text{D})]}$$

DM643xデバイスでは、I2Cの入力クロックは直接CLKINから入力されます。Dで表される量の値はIPSCの値で決定されます(IPSC > 1, D = 5, IPSC = 1, D = 6, IPSC = 0, D = 7)。ブートに使用されるI2Cマスタ･クロックを決定するために、ブート･ローダが常にIPSCに2を設定するためにD=5になります。27MHzオシレータの入力周波数を想定しているため、表 12に示されるIPSC, ICLKH, ICCLKL は次のI2Cマスタ･クロック周波数を導きます。

FASTBOOT == 1

I2C Master clock ~ 310 Khz

FASTBOOT == 0

I2C Master clock ~87 Khz

I2Cモジュールの入力クロックはPLLをバイパスすることに注意してください。そのため、AEM及びPLLMS[2:0]とそれで導かれるPLLMの選択を使ったFASTBOOTオプションはI2Cマスタ･クロックに影響を与えないことに注意してください。

2.10 SPI 16 ×8 マスタ･モード･ブート (BOOTMODE[3:0] = 0110b, FASTBOOT = 0 または 1)

SPI 16×8ブートはデバイスのMCBSP0をSPIモードに設定することにより実装されています。このモードは16ビットのアドレスを要求し、8ビットのデータをフェッチ/受信するSPI EEPROMをサポートします。16ビットのアドレスにより、64K×8間でのサイズをSPIからブートすることができます。ブート･ローダ･ソフトウェアはMCBSP0を32ビットデータ送信/受信するSPIモードに設定します。SPIリード命令と16ビットのアドレスは32ビットワードの上位3バイトにパックされます。4つ目の空のバイトはSPI EEPROMからの8ビットのデータを受信するのに必要なクロック･サイクルとして使われます。ブートのフローは以下のとおりです。

1. FASTBOOTがイネーブルの場合、表 6 に従って選択されたPLLMの値を使ってPLLを設定します。
2. その次にEEPROMからAISフォーマットにされたブート･イメージをリードします。
3. 最後のAIS命令(JUMP CLOSE命令)が現われた時、ブートローダは命令内で与えられたアプリケーションのエントリ･アドレスに分岐します。

2.10.1 SPI 16×8 マスタ･ブート･タイミング

SPIマスタ･クロック周波数はMcBSP0に供給される内部クロックから生成されます。ペリフェラル･クロックはCPUクロックの1/6固定で分周されることにより生成されます。マスタ･クロックはその後さらにMcBSPのSRGRのCLKGDVフィールド値によって分周されます。ブート･ローダ･ソフトウェアCLKGDVの値を2に固定します。これにより、McBSP入力クロックの1/3に分周されます。表 13に可能なPLLM設定を基にした生成されるマスタ･クロック周波数を示します。

表 13 FASTBOOT=1に対するSPIマスタ･クロックの周波数

PLLM	PLLOUT (MHz)	CPU (MHz)	McBSP (MHz)	SPI マスタ (MHz)
19	540	270	45	15
14	404	202	33.7	11
15	432	216	36	12
17	486	243	40.5	13.5
21	594	297	49.5	16.5
24	675	337	56	18.7
26	729	364	60.7	**20.2**
29	810	405	67.5	**20.5**

表の中で与えられているタイミングは27MHzの入力周波数に基にしていることに注意してください。適切なタイミング及び周波数範囲を決定するために、各デバイスのデータシート及びSPI EEPROMのデータシートを確認してください。

2.10.2 SPI 16×8 マスタ･ブート信号極性

MCBSP0はSPIマスタ･ブートのために、選択された次に示すモードで設定されます。

表 14 SPIマスタ･ブート･モード

レジスタ	値
PCR	フィールド･バリュー･セット
	FSXM=1, FSRP = 1, CLKXM = 0, FSXP = 1
SPCR	Field Values Set
	CLKSTP = 3, (11b)
RCR	Field Values Set
	RDATDLY = 1
XCR	Field Value Set
	XDATDLY = 1

これらのモードを選択することに加え、SPIマスタ･クロックの極性はインアクティブ･ハイになり、McBSPは始めのクロックの立ち上がりエッジの前の半周期のクロックサイクルからデータ転送を開始し、クロックの立ち上がりエッジでデータをサンプルします。この動作はクロックの立ち上がりエッジでサンプルし、立ち下がりエッジでデータを送るSPI EEPROMをサポートします。

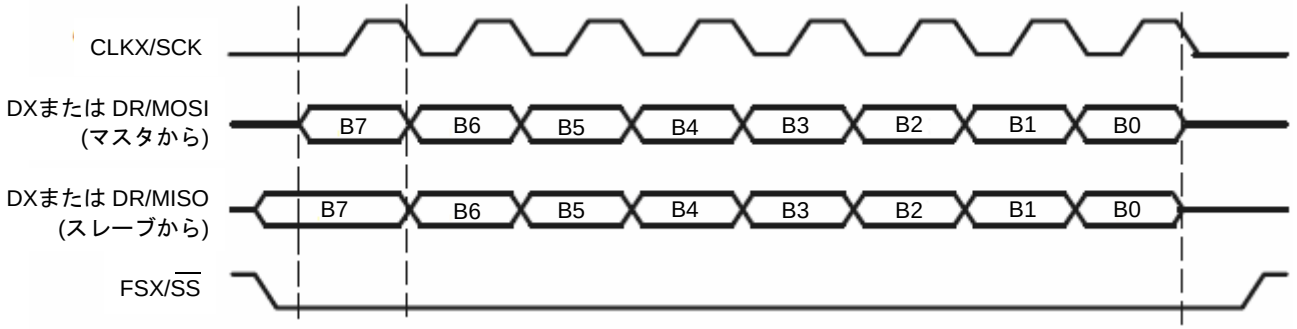

図 1 CLKSTP = 11 かつCLKXP = 0 でのSPI転送

SPIモードにおいてMcBSPがどのように動作するかについての詳細はTMS320C6000 Multi-Channel Buffered Serial Port User's Guide (SPRU580)を参照してください。

2.10.3 SPI 16×8 ブートに対する SPI EEPROM の接続

16×8 SPI EEPROMからのブートを可能にするために、EEPROMを表に示すようにMcBSP0に接続します。

表 15 SPI 16x8 EEPROMとDSP McBSP0 の接続

SPI EEPROM	McBSP0
Sn	FSX0
C	CLKX0
D	DX0
Q	DR0

2.11 NAND フラッシュ・ブート (BOOTMODE[3:0] = 0111b, FASTBOOT = 0 または 1)

NANDフラッシュ・ブートは現在セカンダリ・ブートを通じてサポートされています。Rev 1.10および1.20のデバイスのROMブートローダではNANDレディのポーリング時に競合状態が生じます。そのため、このブート・モードはROMにより完全にサポートされていません。この問題の回避策はI2CやSPIといったその他のサポートされているブート方法でブートし、その次にセカンダリ・ブートローダでNANDフラッシュからブートすることです。サンプルのセカンダリ・ブート・ローダ及びI2CまたはSPIにセカンダリ・コードを書き込むために必要なコードを本資料に付け加えてあります。セカンダリNANDブートによってサポートされるデバイスのリストを表 16 に示します。セカンダリ・ブートローダは動作時にチップ・セレクト信号をトグルさせる必要があるNANDデバイスをサポートしていないことに注意してください。ブートに使用されるNANDデバイスはEMIFAのCS2に接続される必要があります。

表 16 サポートされているNANDデバイスの種別

デバイスID	ページ・サイズ	トータル・メモリ・サイズ
0xE3	512+16	4 MB
0xE5	512+16	4 MB
0xE6	512+16	8 MB
0x39	512+16	8 MB
0x6B	512+16	8 MB
0x73	512+16	16 MB
0x33	512+16	16 MB
0x75	512+16	32 MB
0x35	512+16	32 MB
0x43	512+16	16 MB
0x45	512+16	32 MB
0x53	512+16	16 MB
0x55	512+16	32 MB
0x76	512+16	64 MB
0x36	512+16	64 MB
0x79	512+16	128 MB
0x71	512+16	256 MB
0x46	512+16	64 MB
0x56	512+16	64 MB
0x74	512+16	128 MB
0xF1	2048+64	128 MB
0xA1	2048+64	128 MB
0xAA	2048+64	256 MB
0xDA	2048+64	256 MB
0xAC	2048+64	512 MB
0xDC	2048+64	512 MB
0xB1	2048+64	128 MB
0xC1	2048+64	128 MB

セカンダリ･ブート･ローダはNAND内のデータはAISフォーマットであることを要求します。AISフォーマットはシリアル･ストリームで考えられているので、すべてのAISデータはフラッシュ内の連続したページ･ブロックに格納されている必要があることに注意してください。現在、セカンダリ･ブートローダは不良ブロックをバイパスしようとしません。一旦AISデータの先頭が決定されると、残りのデータはメモリの「良い」ブロック内にあることを想定しています。セカンダリ･ブートローダはECCの1ビットエラーが認識された時、1ビットのエラー訂正を行います。ブートローダはAISマジック･ナンバーを見つけるために、メモリのブロック1からのAISデータを検索し始め、残りのすべてのブロックを検索します。ブロック0はアプリケーションで使用するために予約になっています。

2.12 UART ブート(BOOTMODE[3:0] = 1000b, 1110b, FASTBOOT = 0 または 1)

UARTブートはブートローダ･ソフトウェアがブート中にホストといくつか通信を行う点で他のブートとは異なります。ブートローダはUARTブートが選択された場合次の順序で動作します。

1. FASTBOOT=1の時、表 6 に従って選択されたPLLMの値を使ってPLLを設定します。
2. ブートローダが選択されたモードで要求されているようにUARTレジスタを設定します。
3. ブートローダはシリアル･インターフェースを使ってBOOTMEメッセージをホストに送ります。
4. ブートローダはAISマジック･ナンバーの形式でホストから応答が来るのを待ちます。
5. ホストからの応答を受信した時、ブートローダはJUMP_CLOSEコマンドが現われるまでシリアル･インターフェースからリードしながらAIS命令を処理します。
6. JUMP_CLOSE命令がリードされた時、ブートローダはホストにDONEメッセージを送り、アプリケーション開始アドレスに分岐します。

AIS命令はASCII表示であること要求されていることに注意してください。そのため、AISマジック･ワード0x41504954を送るために、ブートローダで受信できるように「41」、「50」、「49」、「54」のキャラクタ順に送る必要があります。UARTブートのためのサンプルのAISストリームは3 項をご覧ください。

2.12.1 UART ブート･タイミング

動作上、BOOTMODE[3:0]=1000b及び1110bによるUARTブートは原則的に同じです。違いはデータ･フローの管理にあります。BOOTMODE[3:0]=1000bの時、UARTブートはハードウェア･フロー制御なしになります。BOOTMODE[3:0]=1110bが選択された場合、UARTはハードウェア･フロー制御モジュールが使用されるように設定されます。両方のモードでUART FIFOが使用可能で、最大FIFOサイズは14に設定されます。

ブートローダ･ソフトウェアは自動ボー認識を使いません。UARTクロック分周のレジスタはトータルで15になるように設定されています。27MHzの入力クロックに対し、約115kbps(キロビット/秒)のボー･レートになります。UARTに供給されるクロックはPLLをバイパスしているので、ボー･レートはPLLの設定やよりCPUをより高速にするFASTBOOTモードの恩恵を受けることはありません。UARTブートに必要な接続を表 17 に示します。

表 17 ブートのためのUART接続属性

属性	値
ボー･レート	115 kbps
データ･ビット	8
ストップ･ビット	1
パリティ	無し
フロー制御	ハードウェア･フロー制御 (BOOTMODE[3:0]==1110b) 又は無し (BOOTMODE[3:0]==1000b)

2.13 VLYNQ ブート

ROMブートローダはDSPがVLYNQスレーブでのVLYNQ経由でのブートをサポートしています。ブートローダは、VLYNQがイネーブルであることを確認し、VLYNQホストによるアプリケーションのダウンロードが完了したことを示すBOOTCMPLTレジスタのBOOTCMPLT.CMPLTフラグをポーリングします。ブートローダはその次にVLYNQホストによって書かれたDSPBOOTADDR内のスタート･アドレスに分岐します。VLYNQによるブート処理は次のとおりです。

1. FASTBOOTがイネーブルの場合適切なPLL逓倍数に従ってPLLを設定します。
2. ブートローダはVLYNQがイネーブルになっていることを確認します。
3. ブートローダはBOOTCMPLT.CMPLTフラグをポーリングするために空のループに入ります。
4. VLYNQホストはアプリケーションをDSPにダウンロードします。
5. VLYNQホストはアプリケーションの開始アドレスをDSPBOOTADDRにライトします。
6. VLYNQホストはBOOTCPLT.CMPLTレジスタ･フラグに1をライトします。
7. ブート･ローダはBOOTCMPLT.CMPLTを認識し、DSPBOOTADDR内のスタート･アドレスに分岐します。

2.13.1 VLYNQ ブート･タイミング

DSPをVLYNQからブートするに当たって考慮すべき独立した2つのクロックがあります。一つはVLYNQクロック(データ･クロック)で、2つ目は内部のVLYNQモジュールのクロックです。DSPはブートのためにVLYNQスレーブ･デバイスとして動作しているため、VLYNQクロック(データ･クロック)はVLYNQマスタにより外部から供給されます。このクロックにより、VLYNQのデータ･レートが決まります。VLYNQクロック周波数はシステム要求により決定されるべきであり、内部のVLYNQモジュールのクロック周波数とは独立しています。

内部VLYNQモジュール･クロックはDSPのクロック･モジュールから供給され、PLLモードに応じて、CLKIN/PLLOUTの影響を受けます。PLLがバイパス･モードの場合、内部VLYNQモジュール･クロックはCLKINから供給されCLKIN/6になっています。FASTBOOTオプションのどれかが選択された場合、PLLはバイパス･モードではなく、内部VLYNCモジュール･クロックはPLLOUTより供給され、SYSCLK1/6(SYSCLK1はCPUクロック)の値になります。内部VLYNQモジュール･クロックは99MHzを超えないべきです。そのため、FASTBOOTオプションのPLL逓倍数を選ぶ際に、内部VLYNQモジュール･クロックを99MHzの動作周波数かそれ以下に保つためにSYSCLK1の値が594MHzを超えないように注意するべきです。再度書きますが、99MHz制限は内部VLYNQモジュール･クロックのみであり、VLYNQマスタから供給される外部VLYNQクロック(データ･クロック)には影響を与えません。

2.14 SPI 24 ×8 マスタ･モード･ブート (BOOTMODE[3:0]=1111b, FASTBOOT = 0 または 1)

このブート･モードは24ビットのアドレスを要求し、リード/ライトサイクル毎に8ビット転送するSPI EEPROMをサポートします。アドレスの24ビットは16M×8までのサイズのSPI EEPROMをサポートします。SPI 16ビット･モードのようにSPI EEPROMに格納されたデータは有効なAISイメージにフォーマットされていることが要求され、同じブートの流れになります。

1. FASTBOOTがイネーブルの場合、ブートローダはPLLを設定します。
2. その次にブートローダはAISフォーマットされたブート･イメージをEEPROMからリードします。
3. 最後のAIS命令が現われた時(JUMP_CLOSE命令)、ブートローダは命令の中で与えられたアプリケーション･エントリ･アドレスに分岐します。

24ビットのアドレスを要求するSPI EEPROMと通信を行うために、McBSPは24ビットSPIマスタ･ブートのために、16×8マスタ･モードとは違う設定になります。McBSPは元々SPIモードで動作する時送信/受信を32ビットで行わなければならない制限があります。24ビット･アドレスを要求するSPI EEPROMに正しくアドレスを与えるには、実際には40ビット必要になります。コマンドの8ビット+アドレスの24ビット+EEPROMのデータ･イン/アウトです。そのため、SPI 24×8マスタ･ブートでは、通常ではチップ･セレクトとしてFSX0ピンが使用されますが、この代わりにGPIOピンが使用されます。このブート･モードを使用する場合、FSX0品は接続されるべきではなく、使用されません。それから、McBSP0は8ビット･データを送信/受信するように設定されます。このモードではチップ･セレクトはインアクティブ･ハイが想定されていることに注意してください。

SPIデバイスとの通信に必要な40ビットは、分割されたバイトで送信されます。DSPからEEPROMへの転送順序は次のとおりです。

1. GPIOをアサートする(早過ぎるチップ･セレクトのトグルを行わないようにするために、McBSP0をリリースする前に始めにGPIOをハイにドライブします)。
2. GPIOをディアサートします(チップ･セレクトをローにドライブします)。
3. 8ビットのSPI命令を転送します。
4. SPIアドレス･ビット[23:16]を転送します。
5. SPIアドレス･ビット[15:8]を転送します。
6. SPIアドレス･ビット[7:0]を転送します。
7. EEPROMからデータをもらうためのクロックを出すために8ビットのダミーを送ります。
8. ブートローダはDR0から8ビットリードします。
9. GPIOをアサートします(チップ･セレクトをハイにドライブします)。

2.14.1 SPI 24×8 マスタ･ブート･タイミング

SPI 24×8マスタ･ブートのタイミングはシステムPLLの設定及び分周されたMcBSPクロックにより決定されます。これは2.10.1項で詳細について述べられています。

2.14.2 SPI 24×8 ブート信号極性

24×8モードのためのMcBSP SPIモードの設定は、16×8 SPIマスタ･モードと同じで、CLKSTP=11b及びCLKXP=0です。このモードではデータはDSPからクロックの立ち上がりエッジの前の半周期のクロックサイクルからデータ転送され、クロックの立ち上がりエッジで受信されます。この動作モードはクロックの立ち上がりエッジでサンプルし、立ち下がりエッジでデータを送るSPI EEPROMとコンパチブルです。
図はDM643x 24×8ビットSPIブートがどのように動作するかを示します。図 2はDM643xと24×8 SPI EEPROMのピン接続を示します。

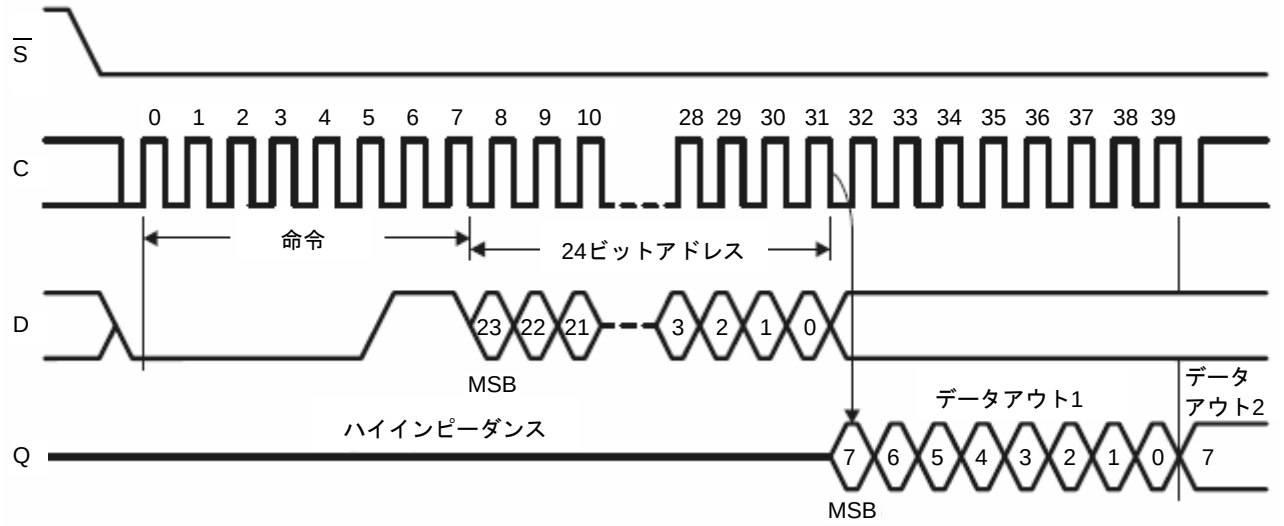

図 2 24×8 ビットSPI EEPROM リード･タイミング

図 3 にどのようにDM643x 24×8ビットSPIブートが動作するかを示します。表 18 にDM643xと24×8 SPI EEPROMとのピン接続を示します。

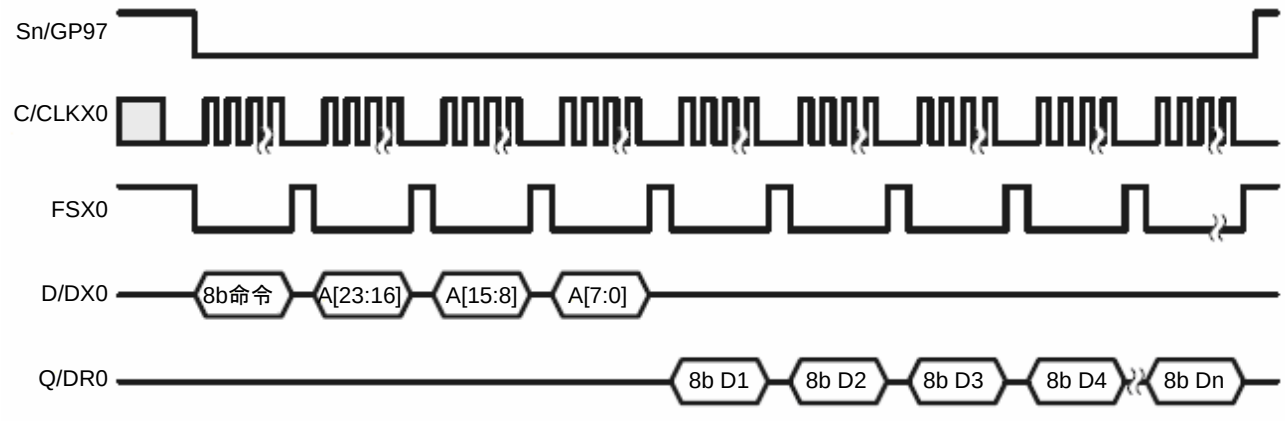

図 3 DM643x 24×8 ビット･アドレスSPI ブート

2.14.3 SPI 24×8 ブートに対する SPI EEPROM の接続

SPI 24×8ブートの場合、McBSP0ピンは表 18 に従ってSPI EEPROMと接続されなければなりません。

表 18 24ビットSPIモードでのSPI EEPROMとDSP のピン接続

SPI EEPROM	DSP	備考
Sn	GPIO97	GPIO97をSPI EEPROMのチップ･セレクトに接続
C	CLKX0	CLKX0をSPI EEPROMのクロックに接続
D	DX0	DX0をSPI EEPROMのデータ入力に接続
Q	DR0	DR0をSPI EEPROMのデータ出力に接続
-	FSX0	この信号は使用しないので、接続無しです

3 アプリケーション･イメージ･スクリプト

ブートローダはアプリケーション･イメージ･スクリプト (AIS)と呼ばれるスクリプト形式のブート情報を取り扱うことができます。アプリケーション･イメージ･スクリプトはTexas Instruments Inc.独自のアプリケーション･イメージの転送フォーマットです。このスクリプトはスクリプト･ヘッダを持っており、これに続いてブートローダによって意味を解釈し実行できるさまざまな命令があります。各命令はオペコードから成っており、オペコードの後には実行に必要な任意の追加データがあります。ブートローダは現在AISバージョン1.99をサポートしており、すべての命令およびデータは32ビット幅であることが想定されています。

AISはマジック・ワード(0x41504954)からなるヘッダで始まります。ヘッダの後は、図 4 に示すように命令が続きます。それぞれの命令はオペコードからなり、その後ろには任意の追加データがあります。すべてのAIS命令のストリームは、ロードされたアプリケーションに制御を渡し、ROMブートローダの実行を終了させるJUMP_CLOSE命令で終わります。

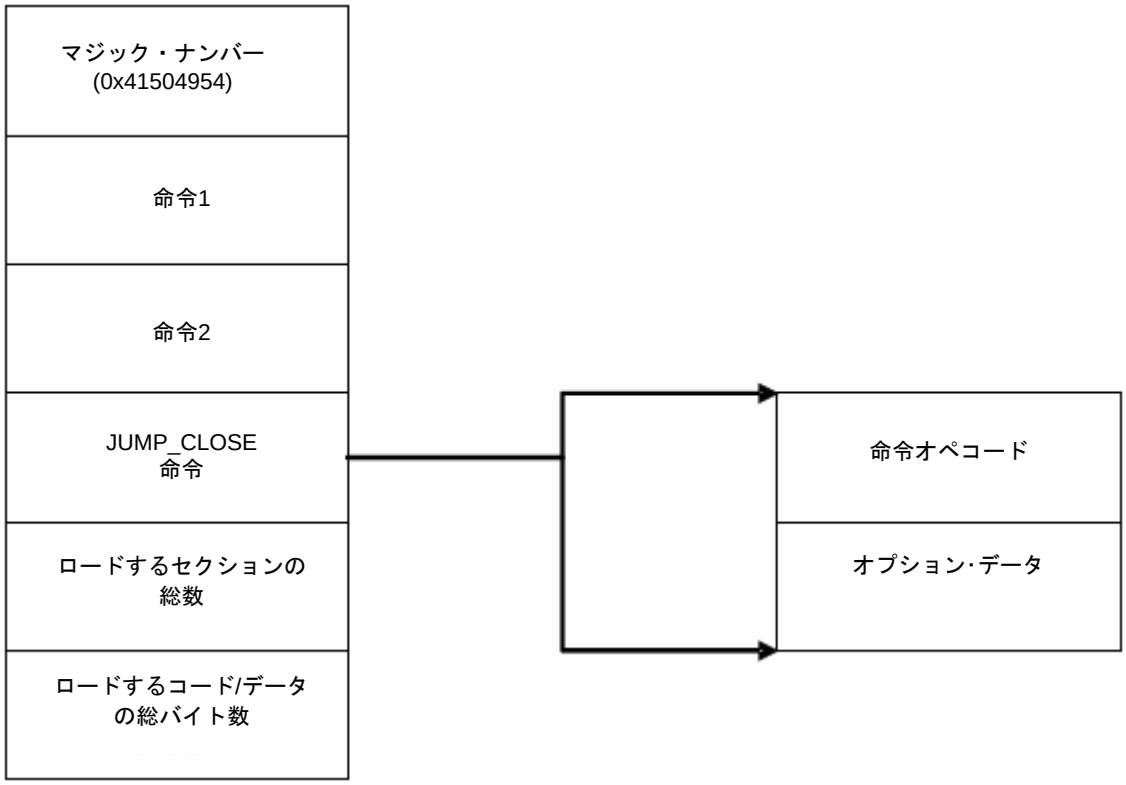

図 4 アプリケーション･イメージ･スクリプトの基本構造

ブートローダはHPI及びPCIブートを除いたすべてのモードでAISフォーマット扱うことができます。次の章以降で、AISの各命令のオペコード、構造、及び配置について述べます。表 19 にAIS 1.0でサポートされる命令を示します。

表 19 AIS 2.0がサポートしているオペコード

オペコード	値
セクション･ロード	0x58535901
リクエストCRC	0x58535902
イネーブルCRC	0x58535903
ディスエーブルCRC	0x58535904
ジャンプ	0x58535905
ジャンプ･クローズ	0x58535906
セット	0x58535907
スタート･オアーバー	0x58535908
予約	0x58535909
セクション･フィル	0x5853590A
ゲット	0x5853590C
関数実行	0x5853590D

3.1 SET 命令

SET命令はDSPのアドレス空間内のすべてのアドレスに8ビット、16ビットまたは32ビットデータをライトすることができる簡単な手段です。この命令の引数のひとつにメモリ・ライトを行った後に入れるディレイがあります。この命令はメモリ・マップドレジスタに対しても使うことができます。SET命令はDSPのさまざまなペリフェラルを設定するために使うことができます。これは少なくともPLLおよびEMIFを含んでおり、さらに必要に応じてより多くのペリフェラルを設定することができます。DSPがリセット後に立ち上がったとき、PLLはバイパス・モードです。そのため、CPUのクロックは接続されたMXIN/CLKINと同じ速度であり、それらは一般的に低速です。この結果、通信速度が遅くなり、ブート時間も長くなります。FASTBOOTを選択することはわずかに早い逓倍数0xCをPLLに設定することによりこの問題を緩和しますが、デフォルトのEMIFのウエート・ステート等を変えることができません。ブート時間を短縮するために、SETコマンドを複数発行することによってブート処理のごく最初の部分でPLL及びEMIFのレジスタを設定しなおすことができます。このため、図 5 に示すようにEMIF及びPLLを設定するすべてのSET命令はAISブート・イメージの先頭に配置するべきです。

図 5 SET命令の構造

各SET命令はSET (0x58535907)オペコードから成っており、この後ろに図に示されるように4ワードの追加のデータが続きます。AISのSET命令エントリは次の表現を使って表されます。

```
<Address> =
<Data＞＜Type>::<Sleep>
```

この命令は、ブートローダにDSPのアドレス空間の<Address> 番地に<Data> をライトさせ、<Sleep> 分のCPUクロックの間スリープします。データ・タイプ・フィールド<Type> は、<Data> が8ビット(B)、16ビット(S) または32ビット(I) のどの大きさでライトされるべきかを決定します。他のすべてのフィールドは表 20に示すように色々な数値フォーマットにすることができます。

表 20 SET命令で使われる数値フォーマット

名前	フォーマット	例1	例2	例3
16進	0[xX][0-9a-fA-F]+	0x1234abCD	0x1000	0x5a
8進	0[0-7]+	02215125715	010000	0132

データ・タイプ・フィールド<Type> は、データのサイズが8ビット(B)、16ビット(S)または32ビット(I)かを決定します。データ・タイプをfieldまたはbitsにすることができます。これにより、あるアドレスのデータの特定のビット範囲を指定することができます。*Field* 及び*bits* データ・タイプに <Type> フィールドは変更されるべき場所を決めるために*start* 及び*stop* ビットにエンコードすることもできます。表 21 に使用できるデータ・タイプ一覧を示します。

表 21 有効なSET命令のデータ・タイプ

データ・タイプ	値
8ビット	1
16ビット	2
32ビット	3
フィールド (1-32ビット)	4
ビット (1-32ビット)	5

Field 及び*bits* は同様にブートローダによって処理されます。これらのタイプ間の違いは、ブートローダは*field* 指定子を使って与えられたアドレスをリード/修正書き込みします。*bits* データ・タイプはアドレスをリードし、新しいアドレスをアドレスに書き込みます。<Type> の指定は*start bit* 、*stop bit* という(上に示した) データ・タイプのためのフィールドを含んだ32ビット・ワードです。*start bit* 及び*stop bit* フィールドは*field(3)* または*bits(4)* のデータ・タイプが使用されている場合のみ必要となります。これらのフィールドは命令で処理されるビットの数を決めます。表 19 に32ビット<Type> の符号化を示します。

3.1.1 有効な SET 命令のデータ・タイプ

図 6 有効なSET命令のデータ・タイプ

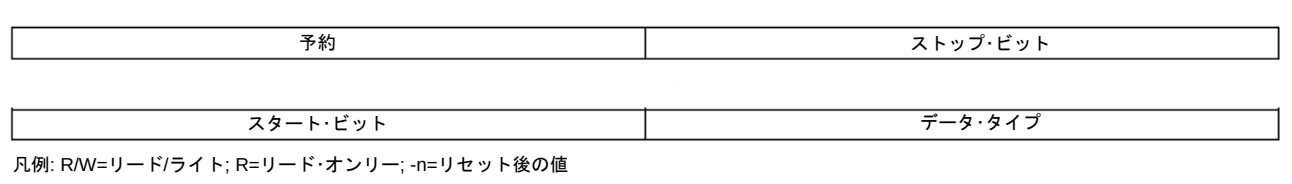

表 22 有効なSET命令のデータ・タイプのフィールド詳細

ビット	フィールド	値	詳細
31-24	予約	0	予約
23-16	ストップ・ビット		ストップ・ビット (*bits* 及び *fields* データ・タイプ) ワード内のフィールドを区切る最後のビット位置
15-8	スタート・ビット		ストップ・ビット (*bits* 及び *fields* データ・タイプ) ワード内のフィールドの開始である最初のビット位置
7-0	データ・タイプ		データ・タイプ (1,2,3,4,5)、ライトするデータのタイプを指定

3.2 GET 命令

GET命令により、リード可能なDSPメモリに格納されている値をフェッチすることができます。GET命令は3.1項で説明されているSET命令とディレイがない点を除き同じフォーマットです。SET命令で説明されたすべてのフォーマットはGETコマンドで使用可能です。GETコマンドは、データが8ビットや16ビット幅であっても常に32ビットの転送を行います。データは0埋めされ右詰めされます(たとえば、すべての32ビット長未満のデータはMSBが0です)。図にGET命令の構造を示します。

図 7 GET命令の構造

3.3 セクション･ロード命令

セクション･ロード命令はコード/データの塊をDSPメモリにロードします。アプリケーションのすべての初期化セクションはセクション･ロード命令を使ってDSPメモリにロードされます。この命令は、AIS内ですべてのSETコマンドの後に配置されます。図 8 にセクション･ロード命令の構造を示します。

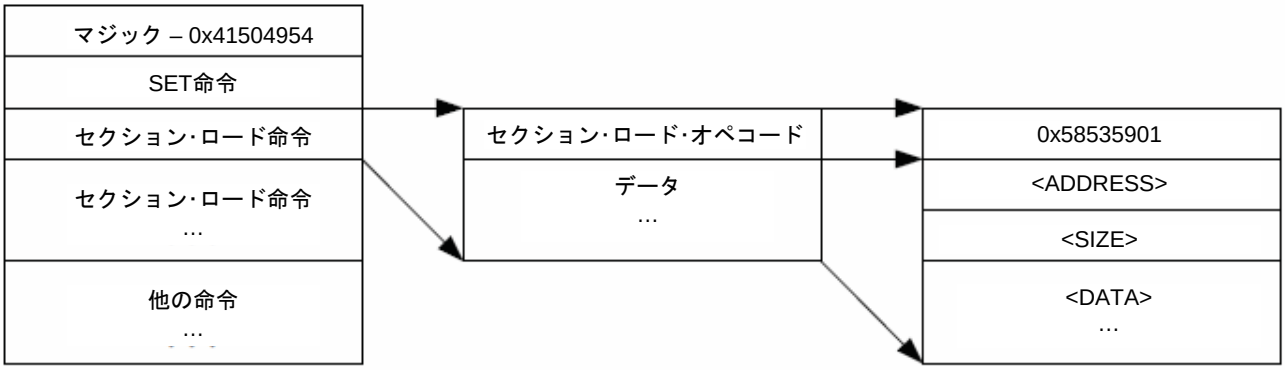

図 8 セクション･ロード命令の構造

各セクション･ロード･コマンドはSECTION_LOAD(0x58535901)オペコードから成り、この後ろにセクション開始アドレス、サイズと内容が続きます。

3.4 セクション･フィル命令

セクション･フィル命令は特定のセクションをあるパターンで埋めるときに使われます。たとえば、すべてが0であるセクションはセクション･フィル命令を使って初期化することができます。この命令は、通常のセクション･ロード命令が配置できる場所であればどこでも配置できます。図 9 にセクション･フィル命令の構造を示します。

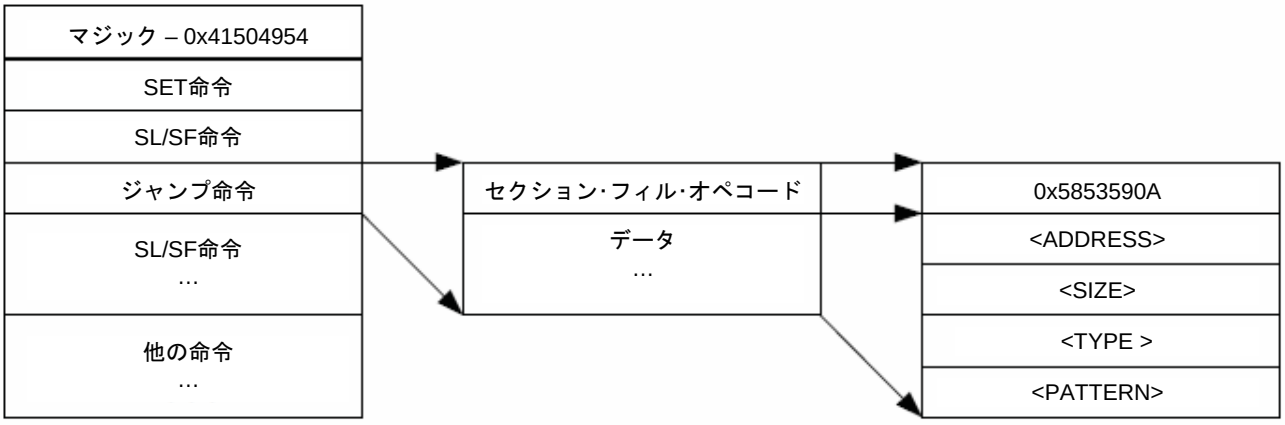

図 9 セクション･フィル命令の構造

各セクション。フィルコマンドはSECTION_FILL(0x5853590A)オペコードから成り、この後ろに開始アドレス、サイズ、パターン･タイプ(8/16/32ビット)及び埋めたいパターンが続きます。

3.5 ジャンプ命令

この命令はDSPをロードされたアプリケーションの開始アドレスに飛ばします。この命令はJUMP(0x58535905)オペコードから成り、この後ろに飛び先のアドレスがあります。図 10 にジャンプ命令の構造を示します。

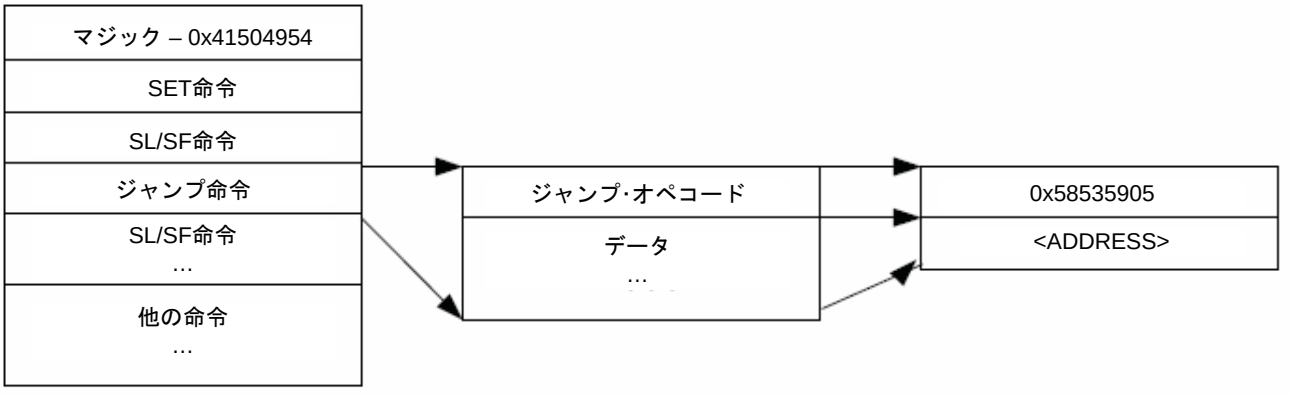

図 10 ジャンプ命令の構造

この命令はブートローダ2を実装するために使われます。これを行うために、セクション·ロード及びセクション·フィル命令を使ってブートローダ2をロードします。そして一旦ブートローダ2の開始アドレスから実行するために、ジャンプ命令を実行します。一旦ブートローダ2の実行が終わると、引き続き、通常のAISの解釈及び実行が行われます。

3.6 ジャンプ·クローズ命令

この命令はロードされたアプリケーションを開始するために、ブート処理の最後で使われます。この命令はDSPのブート処理を終了させ、ロードされたアプリケーションの開始アドレスに飛びます。図 11 にジャンプ·クローズ命令の構造を示します。

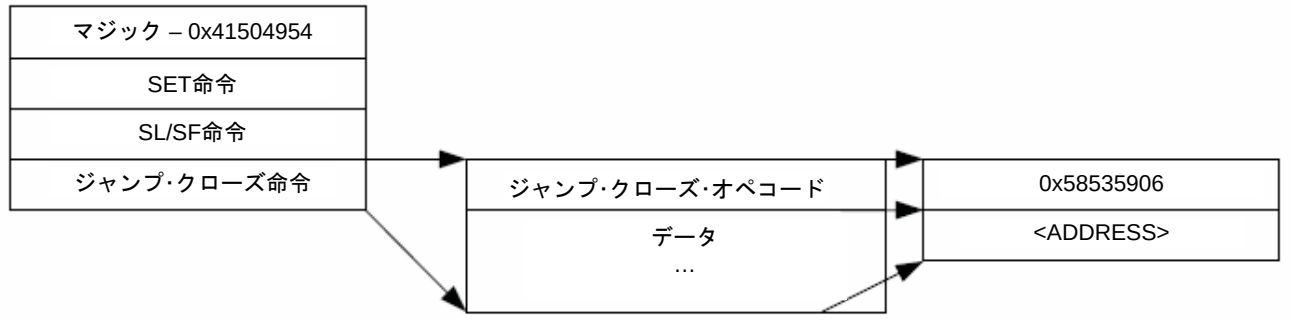

図 11 JUMP_CLOSE命令の構造

この命令はAISの他のすべての命令の後、最後尾に配置します。この命令はJUMP_CLOSE(0x58535906)オペコードから成り、この後ろにブートローダが飛ぶべきアプリケーションの開始アドレスが続きます。アプリケーションのエントリ·ポイントのアドレスに加え、2ワードが続きます。1)ブート中に転送されるべきセクション数、及び2)ブート中にロードされるべき総バイト数。この2ワードがイメージの最後尾の2ワードに配置されます。

3.7 CRC オプション

DSPがブート中に通信エラーが発生する可能性があります。不正なアプリケーション・イメージを実行することは、不安定さや機能不全をもたらします。このような問題を回避するために、AISはセクション・ロード/セクション・フィル命令でロードされたデータの有効性を検証するオペコードをサポートしています。独自の32ビットCRC算出アルゴリズムが検証に使われます。CRCオプションはAIS生成ツールのオプションで指定することにより、実装されています。ツールは次のオプションを指定することにより、CRCイネーブル及びCRCリクエスト命令を埋め込みます。

CRC無し – CRC算出はディスエーブルで、エラーを検出したり、訂正することはできません。

シングルCRC - シングルCRCはすべてのセクションにわたって算出します。検証は最後尾のジャンプNクローズコマンドの直前に行われます。エラーが発生した場合、すべてのセクションは最ロードされます。CRCは最後に再度算出され、再度検証されます。

セクション・ワイズCRC - CRCは各セクションに対して算出されます。検証は各セクションの最後に行われ、エラーが起こった場合はセクションのリロードを試みます。

3.7.1 イネーブル/ディスエーブル CRC 命令

この命令は、セクション・ロード/セクション・フィル命令でセクションをロードする際にCRCの算出をイネーブル/ディスエーブルするために使われます。図 12 にイネーブルCRC/ディスエーブルCRC命令の構造を示します。

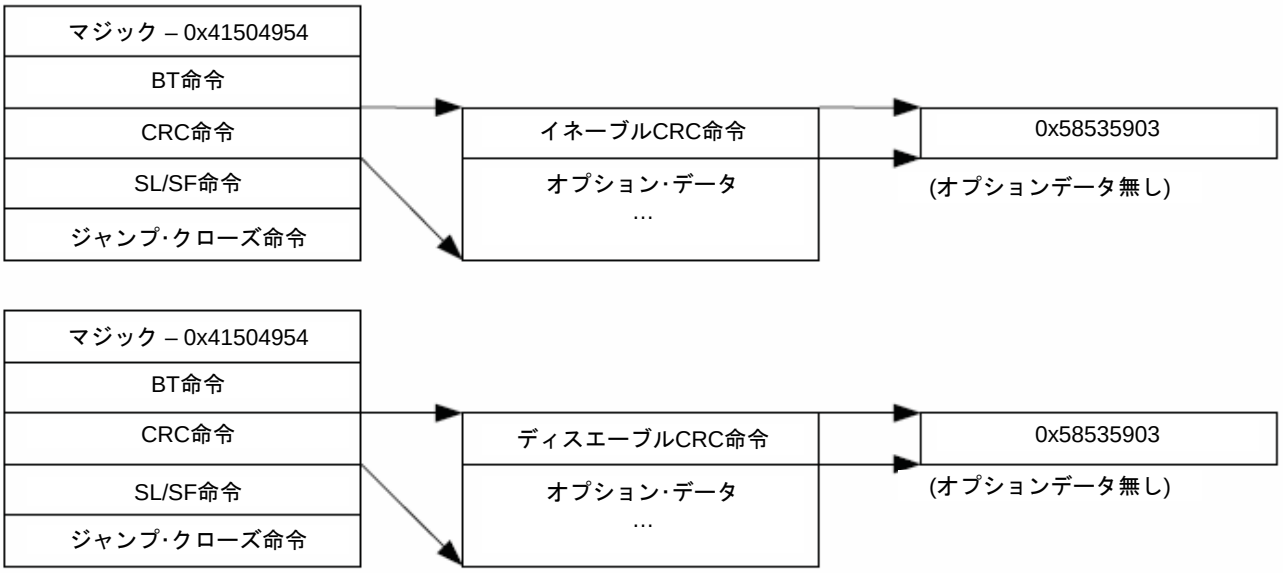

図 12 イネーブルCRC/ディスエーブルCRC 命令の構造

これらのコマンドは、シングルENABLE_CRC(0x58535903)又はDISABLE_CRC(0x58535904)オペコードのみからなります。追加のデータはありません。

3.7.2 リクエスト CRC 命令

この命令はDSPにより算出された現在のCRC値をリクエストし、検証するために使われます。この命令を使用するには、AIS内で先にイネーブルCRC命令を発行しておく必要があります。この命令はREQUEST_CRC(0x58535902)オペコードから成り、この後ろにCRCの期待値及びシーク値が入ります。セクションをロード/フィルした際のCRCは期待値と照合されます。CRCが正しければ、シーク値は無視され、引き続き次の命令が実行されます。

CRCが合わないということは、セクション･ロード/セクション･フィル命令でDSPメモリにロードされたデータが不正であるということです。AISはエラーが起こってないことがわかっている場所の最後から再度実行されます。このポイントを指し示すために、リクエストCRC命令の一部としてシーク値を利用することできます。この値は負の数として解釈され、AISの現在のアドレスに加算されるべきです。これを行うにあたって、アドレスはAIS内の最後にエラーが起こっていないポイントを指します。この更新されたアドレスから引き続き通常通り実行されるべきです。

CRCエラーが起こった場合、ホストは3.7.3項で説明するスタート･オーバー命令を使ってDSPに同じことを通知する必要があります。これを行った後、ホストはシーク値をAISアドレス･ポインタに加え、AISをこのポイントから実行開始します。

スタート･オーバー命令を受信した際、DSPはCRCエラーが発生したことを知ります。DSPはCRC算出をリセットし、ホストからの命令を受け取る準備ができます。

図 13 にリクエストCRCコマンドの構造を示します。

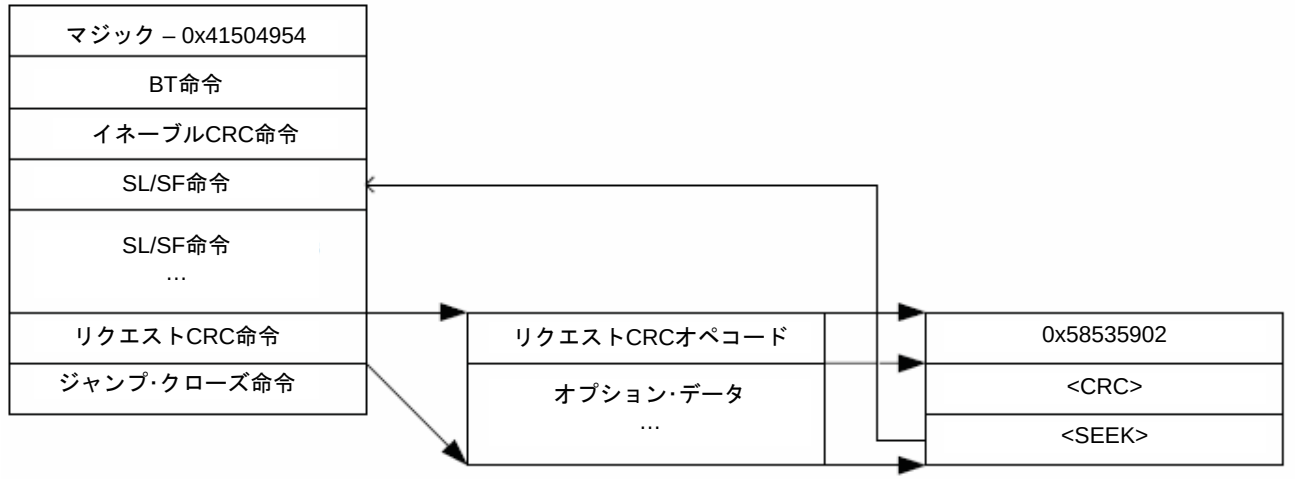

図 13 リクエストCRC命令の構造

シングルCRCオプションの場合、この命令はAIS内に最後のセクション･ロード/セクション･フィル命令の後ろに一度だけ現れます。シーク値は負の数として解釈され、AISの現在のオフセットに関された際、図 13 に示すように初めのセクション･ロード/セクション･フィル命令へのオフセットを生成します。

セクション･ワイズCRCオプションの場合、この命令は各々のセクション･ロード/セクション･フィル命令の後ろに現れます。シーク値は負の数として解釈され、AISの現在のオフセットに関された際、図 13に示すように一つ前のセクション･ロード/セクション･フィル命令へのオフセットを生成します。

3.7.3 スタート･オーバー命令

スタート･オーバー命令はSTARTOVER(0x58535908)オペコードからなり、追加のデータはありません。この命令はブートローダに算出されたCRC値を0にリセットさせます。

この命令は通常、スレーブ･モードでCRC不一致が検出されたときにホストより発行されます。マスタ･モードではブートローダのステート･マシーンによって取り扱われます。

3.8 関数実行命令

図 14に関数実行命令の構造を示します。

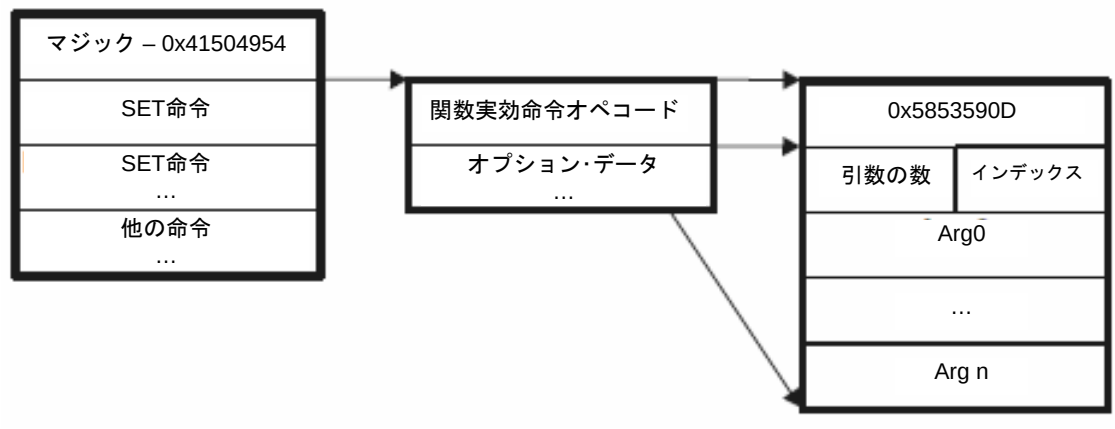

図 14 関数実行命令の構造

関数実行命令により、ブートローダROMコード内に存在する定義済みの関数を実行することができます。DM643xでは、次の関数がブート処理中にPLL、DDRメモリ･コントローラ及びEMIFAを設定できるように定義されています。このコマンドを使ってPLLを設定すると、FASTBOOTオプションを選択してブートローダが行った設定を上書きすることに注意してください。表 23 に定義済みのROM関数の例を示します。

表 23 定義済みROM関数

関数	インデックス	引数の数	詳細
PLL Config	0	3	PLLを設定
EMIFA Config	1	5	EMIFコントロール･レジスタを設定
DDR Config	2	9	DDRコントロール･レジスタをセットして、DDR PLL及びDDRメモリ･コントローラを設定

関数実行命令で命令シーケンスを作成した際、命令オペコードの次のワードの上位16ビットに関数で必要な引数の数が入り、下位16ビットに関数インデックスが入っていなければなりません。

3.8.1 PLL Config 関数

PLL設定関数を使用することにより、FASTBOOTオプションで選択された設定を上書きして設定しなおすことができます。PLL設定関数は3つの引数が必要で、それらは次の順番で与える必要があります。

1. PLL逓倍数
2. PLL分周期1 (CPU/システム･クロックに分周)
3. オシレータ･ソース(0-内部、1-外部)

表 24 にPLL Configの関数実行命令の例を示します。

表 24 関数実行命令のサンプル

AIS データ	詳細
0x5853890D	関数実行オペコード
0x00030000	3つの引数、関数インデックス=0
0x00000019	PLLM=0x19
0x00000001	PLLDIV1 (CPU/Sysclk) = 1, 2 分周
0x00000000	内部オシレータ

3.8.2 EMIFA Config 関数

EMIFA設定関数は、次に示す順序で与えられる5つの引数を使います。

1. AB1CRコントロール･レジスタの値
2. AB2CRコントロール･レジスタの値
3. AB3CRコントロール･レジスタの値
4. AB4CRコントロール･レジスタの値
5. NANDFCRコントロール･レジスタの値

この関数はデバイスのリセット時にラッチされるAEM及びAEAWピンの設定を上書きできないことに注意してください。

3.8.3 DDR Config 関数

DDR2メモリ･コントローラ設定関数は9個の引数が必要です。引数は次の順序で与えられる必要があります。

1. DDR PLLM値
2. DDR CLK分周
3. ビデオプロセッシング･バック･エンド(VPBE)CLK分周
4. DDRクロック･ソース(0-内部、1-外部)
5. DDRコントロール･レジスタの値
6. シンクロナス･ダイナミック･ランダム･アクセス･メモリ(SDRAM)コンフィグ･レジスタの値
7. SDRAMタイマ0レジスタの値
8. SDRAMタイマ1レジスタの値
9. SDRAMリフレッシュ･コントロール･レジスタの値

DDRメモリ･コントローラ設定関数は、関数実行命令で与えられたレジスタ設定を使ってDDR PLL及びDDRメモリ･コントローラを設定します。この関数はDDRメモリ･コントローラの動作を確認するために、DDRメモリ･コントローラ空間にシングル･ライト/リードを行います。

4 オペレーティングシステムのブート(Linux®/DSP/BIOS™等)

ROMブートローダはオペレーティングシステムにより供給されるブート･モードにかかわらず、動作します。どのようなオペレーティングシステムのブート･スタートアップ･コードも先の章で述べたROMブート･モードに準拠したフォーマットになっていなければなりません。ROMブートローダはオペレーティングシステムのスタートアップ･コードと他のアプリケーション･コードの区別をしません。そのため、オペレーティングシステムがそのコードをブートするのに特別なフォーマットを必要とする場合、セカンダリ･ブートによってこれを行わなければなりません。オペレーティングシステムのためのセカンダリ･ブートローダは、そのコードを正しくロードするために、ROMブートローダが扱える適切なフォーマットに成っていなければなりません。オペレーティングシステムのブート･コード(必要であればセカンダリ･ブート)をロードした後、ROMブートローダはオペレーティングシステムのスタート･アップ/ブート･アップに分岐します。もしセカンダリ･ブートローダが必要な場合、セカンダリ･ブートローダはその後に残りのオペレーティングシステムをダウンロードし、実行を開始します。

このシナリオではセカンダリ･ブートローダのみが、選択したブート･モードの適切なROMブートローダのプロトコルに従わなければならないことに注意してください。残りのオペレーティングシステムのコード/データはシステムのロードをするセカンダリ･ブートで必要とされるいかなるフォーマットも使用できます。

例えば、uCLinuxオペレーティングシステムのユニバーサル･ブートを使用する場合、SPI/I2C、高速EMIF等からブートするなら、u-bootのコード自身のみがAISである必要があります。uCLinuxのための残りのコード/データは、u-bootで使用される圧縮されたフォーマットで構いません。u-bootはその後に解凍しDSPメモリに残りをブートします。

5 ROM ブートローダの RAM 要求及びコード/データの配置

ROMブートローダはスタック及びテンポラリのバッファ/データの置き場所としてデバイスの内部メモリ空間内の少量のRAMを使用します。この目的で使用されるメモリはL1D(データ)キャッシュのはじめの20Kバイトにアロケートされます。アプリケーションは初期化済みコード/データ・セクションをこの範囲のメモリに配置してはいけません。これを行うと、ブートローダがブートを行うために使用している不可欠なデータを上書きしてしまい、ブートに失敗します。コンパイラが生成する.bssや.farといった未初期化のセクションは、これらはブート処理が終わり、アプリケーションが実行されるまで存在しないので、この範囲にアロケートすることができます。また、ブートローダはダウンロードされたコード/データをメモリにコピーするのにCPUのライトを使うことにも注意してください。このため、ブートローダは直接L1Pのプログラム領域にコードをロードすることができません。アプリケーションはダウンロードされた後にこの領域を明示的に使用しなければなりません。

6 ROM ブートローダのキャッシュに関する考察

ROMブートローダはブート処理中にL2 RAM及びL1 RAM(L1データ及びL1プログラム双方)のすべてキャッシュをディスエーブルにします。もし、ブート処理中にAIS命令によってキャッシュがイネーブルにされた場合、ブートローダ・コードはアプリケーション・コードが完全にロードされアプリケーションを開始するために分岐する前にもう一度キャッシュをディスエーブルにすることに注意してください。このため、アプリケーション・コードはキャッシュを使う必要があれば、明示的にキャッシュをイネーブルにする必要があります。特に、キャッシュがブート中にイネーブルにされた場合、アプリケーションはキャッシュがデフォルトの電源投入時の状態にあることを想定できません。ブートローダは電源投入時のデフォルトのキャッシュ・レジスタの値を復元せず、終了時にただ単純にディスエーブルにするだけです。

7 AIS 生成ツール、DM643x

DM643xはリンクされたDM643xの実行形式を、ブート・モード及びデータ/メモリ幅に適切なフォーマットに変換するPerlスクリプトです。DM643xはより大きなスクリプトやMakefileの一部として呼び出されるコマンド・ライン・ツールです。DM643xの現在のバージョンはActive Perl V5.8.6を使って開発されています。
DM643xを簡単に実行するには、アプリケーションの実行ファイルの名前、AIS出力ファイルの名前、出力ファイルの形式、ブート・モードおよびイメージが格納されるデバイスのデータ又はアドレス/メモリの幅を指定します。
例えば、

DM643x -i MyApplication.out -o MyApplication.ais -bootmode spi -otype ascii -addrsz 16

これを実行すると、SPIブート用のASCII AIS ファイルに変換されます。AIS 生成ツールはASCII、バイナリ、プレーン・テキスト又はasm 出力ファイルを生成することができます。asm出力ファイルでは、アセンブリの.word 指定子の形でAIS イメージが出力されます。アセンブリ・ファイルはEEPROM の書き込みツールで使用するために、、アセンブル/リンクされHEX 変換ユーティリティを通すかもしれません。DM643x ツールの使用可能なオプションを表 25 に示します。

警告

genAIS ツールは、TMS320C6000 Code Generation Tools Installation Instructions (SPRU237) に伴って供給されるOFDユーティリティ(ofd6x, ofd6x.exe) に依存しています。genAISユーティリティは現在ofd6x v6.1.0.A06333またはそれ以上のバージョンを必要とします。genAIS ユーティリティは、OFD ユーティリティを呼び出すのにPerl のsystem() 関数を使っています。いくつかのバージョンのWindows®NT およびWindows95 オペレーティングシステムは、system() 関数の仕様をサポートしていない可能性があります。この場合、OFD ユーティリティをgenAISの前に実行し、出力されたXMLファイルをgenAISのコマンドラインで入力に指定する必要があります。以下に示すOFDユーティリティを呼び出すオプションは、genAIS を使うにあたって最適なものになっています。そのため、OFD を呼び出すに当たり次のオプションの組み合わせで使用することを推奨します。
例を以下に挙げます。

ofd6x -x --obj_display=none,header,optheader,sections,symbols myApp.out -o myApp.xml
genAIS -i myApp.out -x myApp.xml -o myAIS.txt -bootmode uart -otype txt

(OFDに関するより詳細な情報は TMS320C6000 Assembly Language Tools User's Guide, SPRU186 を参照してください)

表 25 DM643xプログラムのオプション

オプション	詳細
-i filename	入力実行ファイルの名前を指定
-o filename	AIS出力ファイルの名前を指定
-x filename	OFD ツール (ofd6x) からのXML出力の名前を指定
-crc N (N=0,1,2)	CRCの生成方法を指定:
	N=0 - CRC生成無し
	N=1 - 各セクション･ロードでCRCを生成
	N=2 - ロード全体で1つのCRCを生成
-bootmode N (N=i2c, spi, uart. nand, raw)	生成すべきブート･モードを指定
	raw はモード独自のAISイメージを生成することに注意してください
-otype N (N=ascii, binary, txt, asm)	AIS出力のフォーマットを指定
-memwidth N (N=8,16,24)	I2C及びSPIブート･モードで使われる外部メモリのメモリ/アドレス幅を指定。 16ビット幅のメモリは、DM643x及びC6424/C6421デバイスのI2Cに足してのみ、有効なメモリ･タイプであることに注意してください。
-datawidth N (N=8,16)	EMIF 高速ブート･オプションで使用するNORフラッシュのデータ･アクセス幅を指定。このオプションをしても、EMIFからブートした際に、デバイスのEMIF 8_16ビット･ピンを適切に設定することにはならないことに注意してください。
-cfg	AIS出力ファイルの先頭に配置されるセット及び関数実行命令のシーケンスを含んだ、オプションの設定ファイルの名前を指定
-addrsz	SPI EEPROMのアドレス幅をビットで指定。つまり16,24。

8 AIS ブート･イメージのサンプル

AISデータ･ストリームは高速EMIFA、SPI、I2C、NANDフラッシュ及びUARTブート･モードで必要とされます。これらの各モードのサンプルのAISストリームをこの項で示します。この項で扱うAISブート･イメージはDM643xと呼ばれる1つのツールを使って生成されました。DM643xは実行形式にリンクされたアプリケーションを選択されたブート･モードに応じたAISブート･イメージに変換するPerlスクリプトです。DM643xは次項で議論されています。この項で生成されたすべてのブート･イメージは、例1に示すアセンブリ･ソースを使っています。

例1 AIS例のサンプル･ソース･コード

```
;======================================
; Sample Assembly Source File
; a = 6;
; while(1) {
;   b = a + 1;
;   c = b + 2;
; }
;
;======================================
                .global         _a,_b,_c
                .sect           "myData"
_a              .word 0xA
_b              .word 0xB
_c              .word 0xC

                .text
                .global Start
Start:
                MVKL            .S1             _a,A3
                MVKL            .S1             _c,A5
                MVKL            .S1             _b,A4
                MVKH            .S1             _a,A3
||              MVK             .S2             6,B4
                STW             .D1T2           B4,*A3
```

例1 AIS例のサンプル・ソース・コード(続き)

```
||              MVKH            .S1             _c,A5
                MV              .L2X            A3,B5
||              MVKH            .S1             _b,A4
loop:
                LDW             .D2T2           *B5,B4
                NOP                             4
                ADD             .L2             1,B4,B4
                STW             .D1T2           B4,*A4
                NOP                             2
                LDW             .D1T1           *A4,A3
                NOP                             4
                ADD             .L1             2,A3,A3
                STW             .D1T1           A3,*A5
                NOP                             2
                B               .S1             loop
                NOP                             5
```

8.1 EMIF ROM ブートのための AIS ブート･イメージ

EMIFAによりアクセスされるフラッシュ/ROMの最初の8ビット･バイトはEEPROMのサイズになります。有効な数値は、0x00→8ビット、0x01→16ビットになります。その次の3バイトは予約です。最初の有効なAISワードは次の32ビット･ワード境界から始まります。このワードはAISのマジック･ワードである0x41504954でなければなりません。すべての有効なAIS命令はこのマジック･ワードの後ろに格納されます。表 26 にこの項の始めに掲載したサンプル･コードを使った16ビット･フラッシュ用のサンプル･データ･ストリームを示します。

表 26 EMIFA ROM 高速ブートのAIS ブート･イメージ例

データ	説明
0x00000001	ワードの最初のバイトで外部メモリのデータ幅を指定
0x41504954	AIS マジック･ナンバー
0x58535903	イネーブルCRC命令
0x58535901	セクション･ロード命令
0x10800000	セクション･ロード･アドレス
0x00000040	バイト単位でのセクション･サイズ
0x01802028	生セクション･データの先頭
0x02802428	
0x02002228	
0x01884069	
0x0200032A	
0x020C0277	
0x02884068	
0x028C1FDB	
0x02084068	
0x6C6E10CD	
0x10442641	
0x003C2C6E	
0x45B06C6E	
0x2C6E00B4	
0x8C6E008A	
0xEFC08000	生セクション･データの終端
0x58535902	リクエストCRC命令
0x0E85A97B	CRC期待値
0xFFFFFFA8	ストリーム内の最後の有効な命令への負のポインタ
0x58535901	セクション･ロード命令
0x10800040	セクション･ロード･アドレス
0x0000000C	バイト単位でのセクション･サイズ
0x0000000A	生セクション･データの先頭
0x0000000B	
0x0000000C	生セクション･データの終端
0x58535902	リクエストCRC命令
0x8434A250	CRC期待値
0xFFFFFFDC	ストリーム内の最後の有効な命令への負のポインタ
0x58535906	ジャンプ･クローズ命令
0x10800000	アプリケーション･エントリ･ポイント･アドレス
0x00000002	ロードされるべきセクションの総数
0x0000004C	ロードされるべき総バイト数

8.2 I2C ブートのための AIS ブート･イメージ

I2Cブート用のAISヘッダの最初の32ビット･ワードは予約で、ブートローダによって無視されます。2番目の32ビット･ワードはAISマジック･ワードでなければなりません。I2Cブート用のサンプルAISイメージを表 27 に示します。

表 27 I2C AIS ブート･イメージ例

データ	説明
0x00000002	DM643xにおいて予約 – ブートローダは無視する
0x41504954	AIS マジック･ナンバー
0x58535903	イネーブルCRC命令
0x58535901	セクション･ロード命令
0x10800000	セクション･ロード･アドレス
0x00000040	バイト単位でのセクション･サイズ
0x01802028	生セクション･データの先頭
0x02802428	
0x02002228	
0x01884069	
0x02884068	
0x028C1FDB	
0x02084068	
0x6C6E10CD	
0x10442641	
0x003C2C6E	
0x45B06C6E	
0x2C6E00B4	
0x8C6E008A	
0xEFC08000	生セクション･データの終端
0x58535902	リクエストCRC命令
0x0E85A97B	CRC期待値
0xFFFFFFA8	ストリーム内の最後の有効な命令への負のポインタ
0x58535901	セクション･ロード命令
0x10800040	セクション･ロード･アドレス
0x0000000C	バイト単位でのセクション･サイズ
0x0000000A	生セクション･データの先頭
0x0000000B	
0x0000000C	生セクション･データの終端
0x58535902	リクエストCRC命令
0x8434A250	CRC期待値
0xFFFFFFDC	ストリーム内の最後の有効な命令への負のポインタ
0x58535906	ジャンプ･クローズ命令
0x10800000	アプリケーション･エントリ･ポイント･アドレス
0x00000002	ロードされるべきセクションの総数
0x0000004C	ロードされるべき総バイト数

表 28 はI2C EEPROM内のAISブート･イメージの期待されるバイトの並び方を示します。

表 28 I2C EEPROM内のAISイメージ

バイト･アドレス	バイト0	バイト1	バイト2	バイト3	32ビットAISデータ	説明
0x0000	0x02	0x00	0x00	0x00	0x00000002	初めのバイトにアドレスサイズを格納します - このデバイスではブートローダにより無視されます
0x0004	0x54	0x49	0x50	0x41	0x41504954	AISマジック･ワード
0x0008	0x03	0x59	0x53	0x58	0x58535903	イネーブルCRC命令
0x000C	0x01	0x59	0x53	0x58	0x58535901	セクション･ロード命令
0x0010	0x00	0x00	0x80	0x10	0x10800000	セクション･ロード･アドレス
0x0014	0x40	0x00	0x00	0x00	0x00000040	バイト表記でのセクション･サイズ
0x001C	0x28	0x20	0x80	0x01	0x01802028	セクションの生データの先頭
0x0020	0x28	0x24	0x80	0x02	0x02802428	
0x0024	0x28	0x22	0x00	0x02	0x02002228	
0x0028	0x69	0x40	0x88	0x01	0x01884069	
0x008C					0x58535906	ジャンプ･クローズ命令
0x0090					0x10800000	アプリケーション･エントリ･アドレス
0x0094					0x00000002	総セクション数
0x0098					0x0000004C	総バイト数

8.3 SPI ブートのための AIS ブート･イメージ

SPI用のAISブート･イメージは最初の32ビット･ワードにバイト表現されたSPI EEPROMのアドレス幅を含んでいなければならない点を除き、I2Cのイメージとまったく同じです。アドレス幅を含んだバイトはEEPROMのアドレス0に格納されていなければなりません。このアドレス幅のバイトはブートローダが内部的に使用するために存在しています。

表 29 SPI AIS ブート･イメージ例

データ	説明
0x00000002	バイト表記でのEEPROMのアドレス幅 – この値は24ビットSPIでは0x00000003になることに注意して下さい
0x41504954	AIS マジック･ナンバー
0x58535903	イネーブルCRC命令
0x58535901	セクション･ロード命令
0x10800000	セクション･ロード･アドレス
0x00000040	バイト単位でのセクション･サイズ
0x01802028	生セクション･データの先頭
0x02802428	
0x02002228	
0x01884069	
0x0200032A	
0x020C0277	
0x02884068	
0x028C1FDB	
0x02084068	
0x6C6E10CD	
0x10442641	
0x003C2C6E	
0x45B06C6E	
0x2C6E00B4	
0x8C6E008A	
0xEFC08000	生セクション･データの終端
0x58535902	リクエストCRC命令
0x0E85A97B	CRC期待値
0xFFFFFFA8	ストリーム内の最後の有効な命令への負のポインタ
0x58535901	セクション･ロード命令 「myData セクション」

0x10800040	セクション･ロード･アドレス
0x0000000C	バイト単位でのセクション･サイズ
0x0000000A	生セクション･データの先頭
0x0000000B	
0x0000000C	生セクション･データの終端
0x58535902	リクエストCRC命令
0x8434A250	CRC期待値
0xFFFFFFDC	ストリーム内の最後の有効な命令への負のポインタ
0x58535906	ジャンプ･クローズ命令
0x10800000	アプリケーション･エントリ･ポイント･アドレス
0x00000002	ロードされるべきセクションの総数
0x0000004C	ロードされるべき総バイト数

EEPROMに格納されるデータのバイト順は、例として表 30 に示されるAISデータに従うべきであることに注意してください。

表 30 SPI EEPROM内のAIS イメージ

バイト･アドレス	バイト0	バイト1	バイト2	バイト3	32ビットAISデータ	説明
0x0000	0x02	0x00	0x00	0x00	0x00000002	初めのバイトにアドレスサイズを格納します - このデバイスではブートローダにより無視されます
0x0004	0x54	0x49	0x50	0x41	0x41504954	AISマジック･ワード
0x0008	0x03	0x59	0x53	0x58	0x58535903	イネーブルCRC命令
0x000C	0x01	0x59	0x53	0x58	0x58535901	セクション･ロード命令
0x0010	0x00	0x00	0x80	0x10	0x10800000	セクション･ロード･アドレス
0x0014	0x40	0x00	0x00	0x00	0x00000040	バイト表記でのセクション･サイズ
0x001C	0x28	0x20	0x80	0x01	0x01802028	セクションの生データの先頭
0x0020	0x28	0x24	0x80	0x02	0x02802428	
0x0024	0x28	0x22	0x00	0x02	0x02002228	
0x0028	0x69	0x40	0x88	0x01	0x01884069	
0x008C					0x58535906	ジャンプ･クローズ命令
0x0090					0x10800000	アプリケーション･エントリ･アドレス
0x0094					0x00000002	総セクション数
0x0098					0x0000004C	総バイト数

8.4 UART ブートのための AIS ブート･イメージ

UARTブート･モードは、AIS命令の連想に加え、DSPとホスト間でいくつか通信を行う点で、先に説明したモードと異なります。DSPのUARTはブート処理中にスレーブとして動作します。しかし、DSPが起動して、受信する準備ができたことをホストに知らせるために、DSPは初期メッセージであるBOOT MEをホストに送ります。アクナレッジメントとして、ホストはAISマジック･ナンバーで始まるAISブート･イメージを送り始めます。AISデータはASCIIテキストとして送られます。ブートローダ･ソフトウェアは等価な16進定数に変換します。

ブートローダはJUMP CLOSE命令が現れるまでホストから転送されたAIS命令を処理し続けます。JUMP CLOSEコマンドを受信した後、ブートローダはホストにメッセージDONEを送ります。これにより、ブートが正常に完了したことをホストに通知します。

DSP				HOST
送信	→	“BOOT ME”	→	
	←	“41”	←	AISマジック･ナンバーの1番目のバイトを送信
	←	“50”	←	AISマジック･ナンバーの2番目のバイトを送信
	←	“49”	←	AISマジック･ナンバーの3番目のバイトを送信
	←	“54”	←	AISマジック･ナンバーの4番目のバイトを送信
	←	“58”	←	AIS命令の1番目のバイトを送信
	←	“53”	←	AIS命令の2番目のバイトを送信
	←	“59”	←	AIS命令の3番目のバイトを送信
	←	“03”	←	AIS命令の4番目のバイトを送信
			←	ホストはジャンプ･クローズ命令を送信するまで、命令とデータを送信し続ける
	←	“58”	←	ジャンプ･クローズ命令の1番目のバイトを送信
	←	“53”	←	ジャンプ･クローズ命令の2番目のバイトを送信
	←	“59”	←	ジャンプ･クローズ命令の3番目のバイトを送信
	←	“06”	←	ジャンプ･クローズ命令の4番目のバイトを送信
	←	“10”	←	エントリ･ポイント･アドレスの1番目のバイトを送信
	←	“80”	←	エントリ･ポイント･アドレスの2番目のバイトを送信
	←	“00”	←	エントリ･ポイント･アドレスの3番目のバイトを送信
	←	“00”	←	エントリ･ポイント･アドレスの4番目のバイトを送信
	←	“00”	←	セクション数の1番目のバイトを送信
	←	“00”	←	セクション数の2番目のバイトを送信
	←	“00”	←	セクション数の3番目のバイトを送信
	←	“02”	←	セクション数の4番目のバイトを送信
	←	“00”	←	バイト数の1番目のバイトを送信
	←	“00”	←	バイト数の2番目のバイトを送信
	←	“00”	←	バイト数の3番目のバイトを送信
	←	“4C”	←	バイト数の4番目のバイトを送信
送信	→	“DONE”	→	

この時点で、ブート処理が完了し、ブートローダはアプリケーション開始アドレスに分岐します。エラーが発生した場合、例えばCRCエラーが起きた場合、ブートローダはメッセージ CORRUPTをホストに送出し、BOOTCMPLTレジスタのERRフィールドにエラー状態を格納します。その次に再度ブートを試みます。
UART用のAISブート･イメージはエレメント間にスペース、キャリッジ･リターンなしのASCII文字列です。(図 15 参照)

```
41504954585359035853590110800000000000400180202802802428020022280188406902
0032A020C027702884068028C1FDB020840686C6E10CD10442641003C2C6E45B06C6E2
C6E00B48C6E008AEFC08000585359020E85A97BFFFFFFA858535901108000400000000
C0000000A0000000B0000000C585359028434A250FFFFFFDC585359061080000000000000
20000004C
```

図 15 UART AISブート･イメージ

8.5 NAND ブートのための AIS ブート･イメージ

NANDブート用のAISブート･イメージはAISイメージが格納された開始ブロック及びページ数を定義する3ワードを除き、これまでに説明したほかのすべてのイメージととても似ています。これらの情報は、実際にデータがNANDデバイスに書き込まれるまでにわからないため、ユーザがこれら3つのフィールドをAISイメージに埋めなければなりません。DM643xツールは表 31 NANDブートのAIS ブート･イメージ例のように、後で、つまりイメージが最終的にNANDにかかれたときに、実際の値を書き込むための代替として、空きを残しておきます。

表 31 NANDブートのAIS ブート･イメージ例

データ	説明
0x41504954	AIS マジック･ナンバー
0x00000000	イメージが格納されるページ数を格納するための代替
0x00000000	イメージが開始されるブロックを格納するための代替
0x00000000	イメージが開始されるページを格納するための代替
0x58535903	イネーブルCRC命令
0x58535901	セクション･ロード命令
0x10800000	セクション･ロード･アドレス
0x00000040	バイト単位でのセクション･サイズ
0x01802028	生セクション･データの先頭
0x02802428	
0x02002228	
0x01884069	
0x0200032A	
0x020C0277	
0x02884068	
0x028C1FDB	
0x02084068	
0x6C6E10CD	
0x10442641	
0x003C2C6E	
0x45B06C6E	
0x2C6E00B4	
0x2C6E00B4	
0xEFC08000	生セクション･データの終端
0x58535902	リクエストCRC命令
0x0E85A97B	CRC期待値
0xFFFFFFA8	ストリーム内の最後の有効な命令への負のポインタ
0x58535901	セクション･ロード命令
0x10800040	セクション･ロード･アドレス
0x0000000C	バイト単位でのセクション･サイズ

0x0000000A	生セクション･データの先頭
0x0000000B	
0x0000000C	生セクション･データの終端
0x58535902	リクエストCRC命令
0x8434A250	CRC期待値
0xFFFFFFDC	ストリーム内の最後の有効な命令への負のポインタ
0x58535906	ジャンプ･クローズ命令
0x10800000	アプリケーション･エントリ･ポイント･アドレス
0x00000002	ロードされるべきセクションの総数
0x0000004C	ロードされるべき総バイト数

8.6 データ･ファイルの設定

-cfgオプションを使用することにより、セットまたは関数実行命令のシーケンスをAIS出力データ･ファイルの先頭に埋め込むことができます。これにより、外部メモリから/へ適切にブートするためのDDRメモリ･コントローラ、EMIF、PLLを設定することができます。このファイルの命令は、生成されたAISデータよりも前に配置されます。設定ファイル内のデータは、genAISツールによって構文解析されないことに注意してください。単純に直接出力ファイルに書き込まれます。ファイル内に正しいデータ･シーケンスがおかれていることに十分注意してください。PLL、EMIF及びDDRメモリ･コントローラのROM化された設定関数を呼び出すサンプルの設定を以下に示します。

```
0x5853590D # Function Execute Command
0x00030000 # Selects PLL configuration function, with 3 arguments
0x00000015 # PLLM value
0x00000000 # PLLDIV 0
0x00000000 # Clock source
0x5853590D # Function Execute Command
0x00050001 # Selects EMIFA configuration, with 5 arguments
0x3FFFFFFC # AB1CR control register mask
0x3FFFFFFC # AB2CR control register mask
0x3FFFFFFC # AB3CR control register mask
0x3FFFFFFC # AB4CR control register mask
0x00000000 # NANDFCR control register mask
0x5853590D # Function Execute Command
0x00090002 # Selects DDR memory configuration, with 9 arguments
0x00000017 # DDR PLLM
0x00000001 # PLL SRC
0x0000000B # DDR CLLK DIV
0x00000000 # VPBE CLK DIV
0x50006405 # DDR Control register mask
0x00138822 # SDRAM Config register mask
0x16492148 # SDRAM Timer 0 register mask
0x000CC702 # SDRAM Timer 1 register mask
0x000004EF # SDRAM Refresh control register mask
```

9 オン･チップ･ブートローダのバージョンの確認方法

ブートローダのバージョンは、ROMの0x0101A00番地を読むことにより確認することができます。1つ以上のROMバージョンが現在存在しています。ROMバージョン*0x27B2A120* は、EMIFAダイレクトROMブートのみサポートしています。このバージョンを使用しているとき、他のブート･モードを選択すべきではありません。FASTBOOTオプションもこのバージョンでサポートされていません。ROMバージョン*0x00010200* 及び*0x0010300* はFASTBOOTを含め本資料で説明したすべての特徴をサポートしています。

10 CRC の計算

オンチップ･ブートローダは32ビットCRCを使います。CRCを計算するコードを付録 A に示します。オンチップ･ブートローダのために計算されたCRCはBL_updateCrc関数を3回呼び出します。最初の呼び出しは、データ･ワードのセクション･ロード･アドレスを送るために行われます。2番目の呼び出しは、データ･ワードのバイト単位でのセクション･サイズのために使われます。3回目の呼び出しは、実際のセクション･データ、つまりセクション内のすべてのデータ･エレメントのCRCを計算するために、使われます。そのため、最後のCRCは、計算されたセクション･アドレス、セクション･サイズ、及びセクション･データのCRCの合計になります。以下に、CRCの期待値を生成するために関数を呼び出すサンプルを示します。

```
unsigned int crc;
unsigned int sectionAddr;
unsigned int sectionSize;
unsigned int *sectionData;
  crc = BL_updateCRC(&sectionAddr, 4, 0);
  crc = BL_updateCRC(&sectionSize, 4, crc);
  crc = BL_updateCRC(sectionData, sectionSize, crc);
```

最後に計算されたCRCが、REQUSET_CRC命令のCRC期待値であるべきです。もし、アプリケーションのロード全体でシングルCRCを計算したならば、単純に各々のCRC値に対して、BL_updateCRCを継続的に呼び出してください。

```
typedef struct {
  unsigned int sectionAddr;
  unsigned int sectionSize;
  unsigned int *sectionData;
} SectionDatObj;
SectionDataObj mySections[10];

unsigned int crc;
  crc = 0;

  for(i=0;i<10;i++) {
    crc = BL_updateCRC(&(mySections[i].sectionAddr), 4, crc);
    crc = BL_updateCRC(&(mySections[i].sectionSize), 4, crc);
    crc = BL_updateCRC(mySections[i].sectionData, mySections[i].sectionSize, crc);
  }
```

付録 A. CRC の計算

REQUST_CRC命令を処理するために計算されたCRCは、次のアルゴリズムに基づいて計算されています。data_ptrは現在のセクションの最初のデータ・エレメントを指し、section_sizeは8ビット・バイトで表されるセクション・サイズで、crcは現在のcrc値です。

```
unsigned int updateCRC(unsigned int *data_ptr, unsigned int section_size, unsigned int crc)
{
        unsigned int n, crc_poly = 0x04C11DB7; /* CRC - 32 */
        unsigned int msb_bit;
        unsigned int residue_value;
        int bits;

        for( n = 0; n < (section_size>>2); n++ )
        {
                bits = 32;
                while( --bits >= 0 )
                {
                        msb_bit = crc & 0x80000000;
                        crc = (crc << 1) ^ ( (*data_ptr >> bits) & 1 );
                        if ( msb_bit ) crc = crc ^ crc_poly;
                }
                data_ptr ++;
        }

        switch(section_size & 3)
        {
                case 0:
                        break;
                case 1:
                        residue_value = (*data_ptr & 0xFF) ;
                        bits = 8;
                        break;
                case 2:
                        residue_value = (*data_ptr & 0xFFFF) ;
                        bits = 16;
                        break;
                case 3:
                        residue_value = (*data_ptr & 0xFFFFFF) ;
                        bits = 24;
                        break;
        }

        if(section_size & 3)
        {
                while( --bits >= 0 )
                {
                        msb_bit = crc & 0x80000000;
                        crc = (crc << 1) ^ ( (residue_value >> bits) & 1 );
                        if ( msb_bit ) crc = crc ^ crc_poly;
                }
        }
        return( crc );
}
```

ご注意

日本テキサス・インスツルメンツ株式会社（以下TIJといいます）及びTexas Instruments Incorporated（TIJの親会社、以下TIJないしTexas Instruments Incorporatedを総称してTIといいます）は、その製品及びサービスを任意に修正し、改善、改良、その他の変更をし、もしくは製品の製造中止またはサービスの提供を中止する権利を留保します。従いまして、お客様は、発注される前に、関連する最新の情報を取得して頂き、その情報が現在有効かつ完全なものであるかどうかご確認下さい。全ての製品は、お客様とTIJとの間に取引契約が締結されている場合は、当該契約条件に基づき、また当該取引契約が締結されていない場合は、ご注文の受諾の際に提示されるTIJの標準販売契約約款に従って販売されます。

TIは、そのハードウェア製品が、TIの標準保証条件に従い販売時の仕様に対応した性能を有していること、またはお客様とTIJとの間で合意された保証条件に従い合意された仕様に対応した性能を有していることを保証します。検査およびその他の品質管理技法は、TIが当該保証を支援するのに必要とみなす範囲で行なわれております。各デバイスの全てのパラメーターに関する固有の検査は、政府がそれ等の実行を義務づけている場合を除き、必ずしも行なわれておりません。

TIは、製品のアプリケーションに関する支援もしくはお客様の製品の設計について責任を負うことはありません。TI製部品を使用しているお客様の製品及びそのアプリケーションについての責任はお客様にあります。TI製部品を使用したお客様の製品及びアプリケーションについて想定されうる危険を最小のものとするため、適切な設計上および操作上の安全対策は、必ずお客様にてお取り下さい。

TIは、TIの製品もしくはサービスが使用されている組み合せ、機械装置、もしくは方法に関連しているTIの特許権、著作権、回路配置利用権、その他のTIの知的財産権に基づいて何らかのライセンスを許諾するということは明示的にも黙示的にも保証も表明もしておりません。TIが第三者の製品もしくはサービスについて情報を提供することは、TIが当該製品もしくはサービスを使用することについてライセンスを与えるとか、保証もしくは是認するということを意味しません。そのような情報を使用するには第三者の特許その他の知的財産権に基づき当該第三者からライセンスを得なければならない場合もあり、またTIの特許その他の知的財産権に基づきTI からライセンスを得て頂かなければならない場合もあります。

TIのデータ・ブックもしくはデータ・シートの中にある情報を複製することは、その情報に一切の変更を加えること無く、かつその情報と結び付られた全ての保証、条件、制限及び通知と共に複製がなされる限りにおいて許されるものとします。当該情報に変更を加えて複製することは不公正で誤認を生じさせる行為です。TIは、そのような変更された情報や複製については何の義務も責任も負いません。

TIの製品もしくはサービスについてTIにより示された数値、特性、条件その他のパラメーターと異なる、あるいは、それを超えてなされた説明で当該TI製品もしくはサービスを再販売することは、当該TI製品もしくはサービスに対する全ての明示的保証、及び何らかの黙示的保証を無効にし、かつ不公正で誤認を生じさせる行為です。TIは、そのような説明については何の義務も責任もありません。

TIは、TIの製品が、安全でないことが致命的となる用途ないしアプリケーション（例えば、生命維持装置のように、TI製品に不良があった場合に、その不良により相当な確率で死傷等の重篤な事故が発生するようなもの）に使用されることを認めておりません。但し、お客様とTIの双方の権限有る役員が書面でそのような使用について明確に合意した場合は除きます。たとえTIがアプリケーションに関連した情報やサポートを提供したとしても、お客様は、そのようなアプリケーションの安全面及び規制面から見た諸問題を解決するために必要とされる専門的知識及び技術を持ち、かつ、お客様の製品について、またTI製品をそのような安全でないことが致命的となる用途に使用することについて、お客様が全ての法的責任、規制を遵守する責任、及び安全に関する要求事項を満足させる責任を負っていることを認め、かつそのことに同意します。さらに、もし万一、TIの製品がそのような安全でないことが致命的となる用途に使用されたことによって損害が発生し、TIないしその代表者がその損害を賠償した場合は、お客様がTIないしその代表者にその全額の補償をするものとします。

TI製品は、軍事的用途もしくは宇宙航空アプリケーションないし軍事的環境、航空宇宙環境にて使用されるようには設計もされていませんし、使用されることを意図されておりません。但し、当該TI製品が、軍需対応グレード品、若しくは「強化プラスティック」製品としてTIが特別に指定した製品である場合は除きます。TIが軍需対応グレード品として指定した製品のみが軍需品の仕様書に合致いたします。お客様は、TIが軍需対応グレード品として指定していない製品を、軍事的用途もしくは軍事的環境下で使用することは、もっぱらお客様の危険負担においてなされるということ、及び、お客様がもっぱら責任をもって、そのような使用に関して必要とされる全ての法的要求事項及び規制上の要求事項を満足させなければならないことを認め、かつ同意します。

TI製品は、自動車用アプリケーションないし自動車の環境において使用されるようには設計されていませんし、また使用されることを意図されておりません。但し、TIがISO/TS 16949の要求事項を満たしていると特別に指定したTI製品は除きます。お客様は、お客様が当該TI指定品以外のTI製品を自動車用アプリケーションに使用しても、TIは当該要求事項を満たしていなかったことについて、いかなる責任も負わないことを認め、かつ同意します。

弊社半導体製品の取り扱い・保管について

半導体製品は、取り扱い、保管・輸送環境、基板実装条件によっては、お客様での実装前後に破壊/劣化、または故障を起こすことがあります。

弊社半導体製品のお取り扱い、ご使用にあたっては下記の点を遵守して下さい。

1. 静電気
   - 素手で半導体製品単体を触らないこと。どうしても触る必要がある場合は、リストストラップ等で人体からアースをとり、導電性手袋等をして取り扱うこと。
   - 弊社出荷梱包単位（外装から取り出された内装及び個装）又は製品単品で取り扱いを行う場合は、接地された導電性のテーブル上で（導電性マットにアースをとったもの等）、アースをした作業者が行うこと。また、コンテナ等も、導電性のものを使うこと。
   - マウンタやはんだ付け設備等、半導体の実装に関わる全ての装置類は、静電気の帯電を防止する措置を施すこと。
   - 前記のリストストラップ・導電性手袋・テーブル表面及び実装装置類の接地等の静電気帯電防止措置は、常に管理されその機能が確認されていること。
2. 温・湿度環境
   - 温度：0～40℃、相対湿度：40～85%で保管・輸送及び取り扱いを行うこと。（但し、結露しないこと。）
   - 直射日光があたる状態で保管・輸送しないこと。
3. 防湿梱包
   - 防湿梱包品は、開封後は個別推奨保管環境及び期間に従い基板実装すること。
4. 機械的衝撃
   - 梱包品（外装、内装、個装）及び製品単品を落下させたり、衝撃を与えないこと。
5. 熱衝撃
   - はんだ付け時は、最低限260℃以上の高温状態に、10秒以上さらさないこと。（個別推奨条件がある時はそれに従うこと。）
6. 汚染
   - はんだ付け性を損なう、又はアルミ配線腐食の原因となるような汚染物質（硫黄、塩素等ハロゲン）のある環境で保管・輸送しないこと。
   - はんだ付け後は十分にフラックスの洗浄を行うこと。（不純物含有率が一定以下に保証された無洗浄タイプのフラックスは除く。）

以上