参 考 資 料

JAJA177– 2008年12月

TMS320DM647/DM648 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、TMS320DM647/DM648 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、不適切なブート･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。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(AIS)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2 章で説明されています。

本資料はhttp://www-s.ti.com/sc/techlit/spraaj1.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。

- エミュレーション･ブート
- HPI
- PCI (DSPはスレーブ)
- EMIFA CS2 ROMダイレクト･ブート
- AIS有り EMIFA CS2 ROM高速ブート
- I2C (DSPはマスタ) (ROMバージョン3.70以降でサポート)
- SPI (DSPはマスタ)
- フロー制御無し UART (DSPはスレーブ) (ROMバージョン3.70以降でサポート)
- フロー制御有り UART (DSPはスレーブ) (ROMバージョン3.70以降でサポート)
- イーサネット(ROMバージョン3.70以降でサポート)

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、ブート中はウォッチドッグ･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ォッチドッグ･タイマの設定を行ってはいけません(AISコマンドやコードにてブート中に設定を行なうべきではありません)。

表1. 用語と略語

用語	説明
ブートローダ	ROM DM647/DM648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)
DMP	デジタル･メディア･プロセッサ(本資料ではDM647/8を指します)
I2C	インター･インテグレーティッド･サーキット

SPRAAJ1B翻訳版

最新の英語版
http://www.ti.com/lit/spraaj1

OS	オペ-コード･シンクロナイゼイション
POS	ピング･オペ-コード･シンクロナイゼイション
ROM	リード･オンリー･メモリ
SPI	シリアル･ペリフェラル･インターフェイス
SWS	スタート-ワード･シンクロナイゼイション
EMIF	外部メモリ･インターフェイス
PLL	フェイズ･ロック ト･ループ
EEPROM	エレクトリカリ･イレーザブル･プログラマブル･リード-オンリー･メモリ
McBSP	マルチチャネル･バッファード･シリアル･ポート
FIFO	ファースト･イン･ファースト･アウト
HPI	ホスト･ポート･インターフェイス
PCI	ペリフェラル･コンポーネント･インターコネクト
UDP	ユーザ･ダイアグラム･プロトコル

目次

図

表

1 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･ローダは、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。

ブートローダは、デバイスの/PORリセットでサンプルされるFASTBOOTピンの状態によって決められるFASTBOOTオプションをサポートしています。FASTBOOTがイネーブルされていると、ブートローダは、決められたPLLの逓倍値でPLLを設定します。DM647/DM648のブートローダは、イーサネット･ブートを除く全てのブート･モードにおいて、このPLLの逓倍値を19に設定します。これは入力クロックに対して、x20のPLL逓倍となります。イーサネット･ブート･モードが選択されていると、ブートローダは常にPLLを設定します。FASTBOOTオプションがアサートされていると、ブートローダは入力クロックの25倍を供給する逓倍値24をPLLにプログラムします。イーサネット･ブートでFASTBOOTがアサートされていない場合は、ブートローダはデフォルトの逓倍値19を使用します。

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。各有効なモードの概要については、次章にて説明されます。イーサネット･ブート、I2CブートおよびUARTブート･モードは、ROMバージョン3.70にてサポートされます。他の全てのモードは、全ての現存しているROMバージョンにてサポートされます。

表2および表3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. DM648ブート･モード

デバイス･ブートおよびコンフィグレーション･ピン			ブート詳細	genAIS DMP	DSPBOOTADDR
BOOTMODE[3:0]	UHPIEN	FASTBOOT		(マスタ/スレーブ)	(デフォルト)
0000	0 or 1	0 or 1	ブートなし(エミュレーション･ブート)	-	0x0080 0000
0001	0	1	自動初期化無し PCIブート	スレーブ	0x0080 0000
	1	0 or 1	HPIブート	スレーブ	0x0080 0000
0010	0	1	自動初期化有り PCIブート	スレーブ	0x0080 0000
	1	0 or 1	HPIブート	スレーブ	0x0080 0000
0011	0 or 1	0	ハードウェア･フロー制御無しUART ブート	スレーブ	0x0080 0000
0100	0 or 1	0	EMIFA ROMダイレクト･ブート (PLLバイパス･モード)	マスタ	0xA000 0000
		1	AIS有りEMIFA ROMブート	マスタ	0x0080 0000
0101	0 or 1	0 or 1	I2Cブート(標準モード)	マスタ	0x0080 0000
0110	0 or 1	0 or 1	SPIブート	マスタ	0x0080 0000
0111	0 or 1	0 or 1	予約	-	0x0080 0000
1000	0 or 1	0 or 1	SGMII0 – ブート･ポート パケット･フォワードなし	スレーブ	0x0080 0000
1001	0 or 1	0 or 1	SGMII0 – ブート･ポート SGMII1パケット･フォワード	スレーブ	0x0080 0000
1010	0 or 1	0 or 1	SGMII1 – ブート･ポート SGMII0パケット･フォワード	スレーブ	0x0080 0000
1011	0 or 1	0 or 1	予約	-	0x0080 0000
1100	0 or 1	0 or 1	予約	-	0x0080 0000
1101	0 or 1	0 or 1	予約	-	0x0080 0000
1110	0 or 1	0	ハードウェア･フロー制御有りUART ブート [UART0]	スレーブ	0x0080 0000
1111	0 or 1	0 or 1	予約	-	0x0080 0000

表3. DM647ブート･モード

デバイス･ブートおよびコンフィグレーション･ピン			ブート詳細	genAIS DMP (マスタ/スレーブ)	DSPBOOTADDR (デフォルト)
BOOTMODE[3:0]	UHPIEN	FASTBOOT			
0000	0 or 1	0 or 1	ブートなし(エミュレーション･ブート)	-	0x0080 0000
0001	0	1	自動初期化無し PCIブート	スレーブ	0x0080 0000
	1	0 or 1	HPIブート	スレーブ	0x0080 0000
0010	0	1	自動初期化有り PCIブート	スレーブ	0x0080 0000
	1	0 or 1	HPIブート	スレーブ	0x0080 0000
0011	0 or 1	0 or 1	ハードウェア･フロー制御無しUART ブート	スレーブ	0x0080 0000
0100	0 or 1	0	EMIFA ROMダイレクト･ブート (PLLバイパス･モード)	マスタ	0xA000 0000
		1	AIS有りEMIFA ROMブート	マスタ	0x0080 0000
0101	0 or 1	0 or 1	I2Cブート(標準モード)	マスタ	0x0080 0000
0110	0 or 1	0 or 1	SPIブート	マスタ	0x0080 0000
0111	0 or 1	0 or 1	予約	-	0x0080 0000
1000	0 or 1	0 or 1	SGMII0 – ブート･ポート パケット･フォワードなし	スレーブ	0x0080 0000
1001	0 or 1	0 or 1	予約	-	0x0080 0000
1010	0 or 1	0 or 1	予約	-	0x0080 0000
1011	0 or 1	0 or 1	予約	-	0x0080 0000
1100	0 or 1	0 or 1	予約	-	0x0080 0000
1101	0 or 1	0 or 1	予約	-	0x0080 0000
1110	0 or 1	0 or 1	ハードウェア･フロー制御有りUART ブート [UART0]	スレーブ	0x0080 0000
1111	0 or 1	0 or 1	予約	-	0x0080 0000

1.1 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。

1. FASTBOOTは全てのPCIブートモードに必要で、SGMIIブート･モードではデフォルトです。
2. ブートローダは、I2C EEPROMにおいて16ビット･アドレス幅のみサポートします。
3. 自動初期化付きPCIブートにおいて、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。
4. 全てのブート･タイミングは、27MHzの入力クロック周波数に最適化されています。
5. I2C、SPI、UART及びEMIFA CS2 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されている必要があります。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1.7項で述べます。HPIやPCIといったホスト･ブート･モードでは、フォーマットをユーザが決めることができます。
6. FASTBOOTが選択された時、ブートローダはPLLを設定します。PLLの逓倍値は、SYSCLK値としてCLKINx20となる19に固定されています。本資料では、デバイスへの入力クロックが27MHzとして全てのタイミングが計算されています。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。
7. デフォルトで、DMPがリセットから解除されるとキャッシュはイネーブルされます。ROMブートローダが動作しているとき、ブート中全てのL1及びL2キャッシュはディスエーブルです。これはEMIFAダイレクト･ブートを除いて、全てのブート･モードに適用されます。EMIFAダイレクト･ブートでは、ROMブートローダ･コードは実行されません。そのため、キャッシュはイネーブルされ、デフォルトのパワーオン設定に従って処理されます。
8. UARTブート･モードにおいて、27MHz入力クロック周波数が必要となります。この要求が必要とされる唯一のブート･モードです。

1.2 FASTBOOT モード

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。ブートローダは、決められたPLLの逓倍値でPLLを設定します。これは、エミュレーション･ブートを除いた全てのブート･モードに適用されます。FASTBOOTはエミュレーション･ブートの場合、無視されます。全てのイーサネット/SGMIIブート･モードにおいて、FASTBOOTピンの状態に関わらず、PLLはプログラムされます。イーサネット/SGMIIブート･モードが選択されると、FASTBOOTピンの状態によってPLLの逓倍値が決定されます。FASTBOOT=1かつBOOTMODE[3:0]=1000b, 1001b, 1011bのとき、PLLの逓倍値はSYSCLK値としてCLKIN x 25となる24が設定されます。FASTBOOT=0かつイーサネット/SGMIIブート･モードが選択されていると、PLLの逓倍値は通常のFASTBOOT PLL逓倍ちである19に設定されます。その他のブート･モードでは、PLL逓倍値はSYSCLK値としてCLKIN x 20となる19に固定されます。

1.2.1 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、エミュレーション･ブートとSGMIIブートを除く全てのブート･モードにおいてFASTBOOTオプションに対するPLLを設定する際、固定されたPLL逓倍値である19を用います。SGMIIブートにおいて、FASTBOOTのPLL逓倍率は24に固定されています。本資料における全てのタイミングは、DMP/CPUに対して27MHz入力クロック、イーサネット･ポートのコンフィグレーション用のSERDESに対しては62.5MHzに基づいています。各デバイスのデータシートを参照し、デバイスやアプリケーションに最も適したCLKIN周波数を決定してください。仮に設定している入力クロック周波数とFASTBOOTで用いられるPLLの逓倍率では、デバイスの最大CPU周波数が得られない場合があります。FASTBOOTモードの目的は、CPUのクロッキングやブート･ペリフェラル、速いスピードでブート時間の短縮を有効にするためです。ブート時に最大CPU周波数にセットされていない場合、アプリケーションにてブート完了後に最大動作周波数になるようにPLLの設定を変更してください。

1.3 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。このモードでは全てのFASTBOOTオプションが無視されます。PLLはバイパス･モードで動作し、CPUに27MHzが供給されます。

1.4 HPI ブート (BOOTMODE[3:0]=0001b または 0010b,または 0011b,UHPIEN=1,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。それから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PLL逓倍値19となるように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. DMP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。

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

DM647/DM648はDMP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. PLL逓倍値19となるようにPLL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します。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. 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ブート及びFASTBOOTが選択される時、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 表 4 に示すようにフォーマットされ、データはEEPROMのバイトアドレス0x400から始まる必要があります。

表4. 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します。

1.6 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。DMPハードウェア･ブート･モジュールが直接アドレス0xA0000000番地に分岐します。コード実行はこのアドレスから開始されます。

1.7 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、DMP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0xA0000000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2章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DMP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･ローダの順序は以下のとおりです：

1. PLL逓倍値19となるように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。

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

DM647/DM648は、DMPがI2Cマスタの場合のみI2Cブートをサポートします。DMP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領域です。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2章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PLL逓倍値19となるようにPLLを設定します(CLKINx20のSYSCLKを生成)。
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。

1.8.1 I2C マスタ･ブート･タイミング

ブート･ローダは、FASTBOOTコンフィグレーション･ピンの値によって、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5. I2Cタイミング･レジスタの設定

FASTBOOT	システムクロック	I2Cペリフェラル･クロック周波数	レジスタ	値	I2Cマスタ･クロック周波数
0	27MHz	4.5MHz	IPSC	0x1	140KHz
			ICCLKH	0x2	
			ICLKL	0x2	
1	540MHz	90MHz	IPSC	0xB	187KHz
			ICCLKH	0xF	
			ICCLKL	0xF	

I2Cマスタ･クロックは次の式から導かれます：

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

DM647/DM648デバイスでは、I2Cのペリフェラル･クロックはI2Cに供給される内部クロックから生成されます。そのペリフェラル･クロックはSYSCLKの1/6固定で分周されることにより生成されます。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11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 5にIPSC, ICLKH, ICCLKL値に対するI2Cクロック周波数を示します。

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

SPIマスタとしたDMPがブート･ローダによってサポートされる唯一のモードであるので、ブート･ローダはSPIマスタとして動作するようにSPIを設定し、それから接続されているSPI EEPROMからリードされたデータにて初期化します。ブートのフローは以下のとおりです：

1. FASTBOOTがイネーブルされている場合、PLL逓倍値19となるようにPLLを設定します。
2. その次に、ブート･ローダはバイト数で表されるアドレス幅を取り出すためにEEPROMから際そのバイトデータをリードします。
3. 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。
4. 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。

1.9.1 SPI マスタ･ブート･タイミング

SPIマスタ･クロック周波数はSPI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。さらに、SPIマスタ･クロック周波数はプレスケール値によって決定されます。FASTBOOTがイネーブルされる時、ブート･ローダは固定のプリスケール値 7 を使用します。FASTBOOTがディスエーブルされている時、クロック･プリスケール値は 0 にセットされます。表 6にFASTBOOTコンフィグレーションを基に生成されるマスタ･クロック周波数を示します。

表6. SPIマスタ･クロックの周波数(1)

FASTBOOT	PRESCALE	SYSCLK	SPIペリフェラル･クロック	SPIマスタ
0	0	27MHz	4.5MHz	4.5MHz
1	7	540MHz	90MHz	11.25MHz

(1) 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ます。SPI EEPROMに関する情報や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を確認してください。

1.9.2 SPI マスタ･ブート信号極性

SPI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7. SPIマスタ･ブート･モード

レジスタ	値
SPIFMT	フィールド･バリュー･セット
	DISCTIMERS = 1, POLARITY = 0, PHASE = 1, PRESCALE = 0 or 7, CHARLEN = 8
SPIFUN	フィールド･バリュー･セット
	SOMIFUN = 1, SIMOFUN = 1, CLKFUN = 1
SPIDIR	フィールド･バリュー･セット
	SCSDIR0 = 1
XCR	フィールド･バリュー･セット
	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、最初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が発生します。

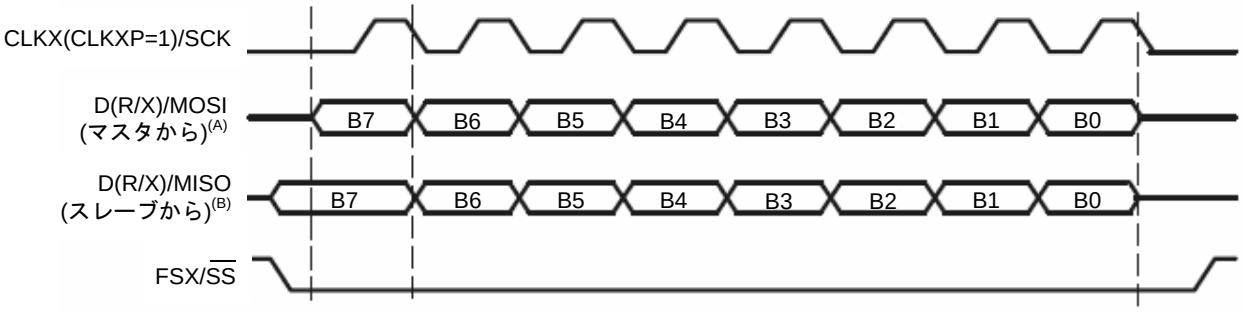


図1. Polarity=1 かつPhase=1 でのSPI転送

1.10 UART ブート(BOOTMODE[3:0] = 0011b, 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。UARTブートが選択された場合、ブートローダは次の順序で動作します。

1. FASTBOOT=1の時、PLL逓倍値19 となるように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･インターフェイスを使って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･インターフェイ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表記を要求す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」のキャラクタを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 2章 をご覧ください。

1.10.1 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8 に示します。

表8. ブートのためのUART接続属性

属性	値
ボー･レート	11Kbps
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]==0011b)

1.11 イーサネット･ブート･モード - SGMII0 のみ, SGMII0 及びSGMII1 (BOOTMODE[3:0] = 1000b, 1001b, 1010b)

イーサネット･ブート･モードにおいて、ROMブート･ローダはイーサネット経由でのブートをイネーブルするために、コミュニケーション･プロセッサ･ギガビット･イーサネット･スイッチ(CPSW_3G)を設定します。CPSW_3Gは3ポートのギガビット･イーサネット･スイッチです。そのポートのうちポート0とポート1の2つは、ブートするために設定が行なわれます(ポート2はDMPへのポートとして割り当てられています)。デバイスのブート･ポートは選択されたブート･モードに依存するので、どちらかのポートが動作します。DM647では、ポート0のみが使用可能なので、SGMII0のみがブートを行うBOOTMODE[3:0] = 1000bのモードのみがサポートされます。

SGMII0ブート･モード(BOOTMODE[3:0] = 1001b)が設定されていると、3-ポート･スイッチのポート0はブート･ポートとして設定されます。スイッチのポート1はパケット-フォワーディング･モードにセットされます。この設定は複数のDMPがアプリケーション･ボード上でデイジーチェインされている場合、パケットを適切に各デバイスへフォワードするためのものです。SGMII1ブート設定(BOOTMODE[3:0] = 1010b)では、ポート1がブート･ポートとして設定され、ポート0がパケット･フォワーディング･モードにセットされます。ポートが異なるだけで、これら2つのブート･モード(SGMII0/SGMII1)は同じ動作です。イーサネット･ブートのブート･フローは次の通りです：

- シリアライザ/デシリアライザ(SerDes)モジュールを設定
- 次の手順に従って、CPSW_3Gを設定
  - ポート0, 1 及び 2 用のフロー制御をイネーブルします。
  - (デバイスの$\overline{\text{POR}}$リセット時にEFUSEからリードされた)DMP MACアドレス･レジスタにラッチされたMACアドレスをリードします。
  - CPGMAC_SL0(ポート0)、CPGMAC_SL1(ポート1)ソース･アドレス･レジスタを設定します。
  - CPDMA送信及び受信をイネーブルします。
  - ALE(アドレス･ルックアップ)をイネーブルします。
  - PORリセット時にMACアドレス･レジスタにラッチされるデバイスMACアドレスの単一エントリに挿入するALEルックアップ･テーブルをクリア及び初期化します。
  - 全てのポート(0,1,2)をパケット･フォワーディング･モードに設定します。
  - ポート0とポート1をGMIIIに設定します。
  - SGMII0ブートが選択されている場合、ポート0をスレーブ･ポートとして、ポート1をオート･ネゴシエーションがイネーブルされたマスタに設定します。
  - SGMII1ブートが選択されている場合、ポート1をスレーブ･ポートとして、ポート0をオート･ネゴシエーションがイネーブルされたマスタに設定します。
- オート･ネゴシエーションが完了する時間を保証するために、CPSW_3Gが完全に設定された後、おおよそ25msのウェイトが挿入されます(この時間はプロセッサの動作速度によって変わります。25msの遅延はCPU速度が500MHzとして計算されています)。

- スイッチが設定されると、ブート･ローダは送信及び受信ディスクリプタ･バッファを初期化し、ブート･ポートからBOOTPリクエスト･パケット(イーサネット･レディ･アナウンスメント･フレーム)を送信します。

Note: BOOTPリクエスト･パケットは、BOOP仕様に準拠していなく、TI固有のものです。通常、このパケットは標準のサーバーにて破棄もしくは拒否するでしょう。

- BOOTPリクエスト･パケットは一度のみ送信されます。その後、ブート･ローダはホスト/サーバーからのBOOTテーブル･パケットを受信するまで待機します。
- 全てのBOOTパケットを受信すると、ブート･ローダはダウンロードされたアプリケーション･コードの先頭に分岐します。

この設定を用いたシステムのパケット･フローを図2に示します。この図はポート0がブート･ポート、ポート1がパケット･フォワーディング用に設定されいると仮定したSGMII0ブート･モードを示しています。SGMII1ブート･モードでは、ポートの役割が逆になります。BOOTテーブル･フレームはポート1で受信され、自分のデバイスに対するパケットでない場合はポート0を経由してフォワードされます。

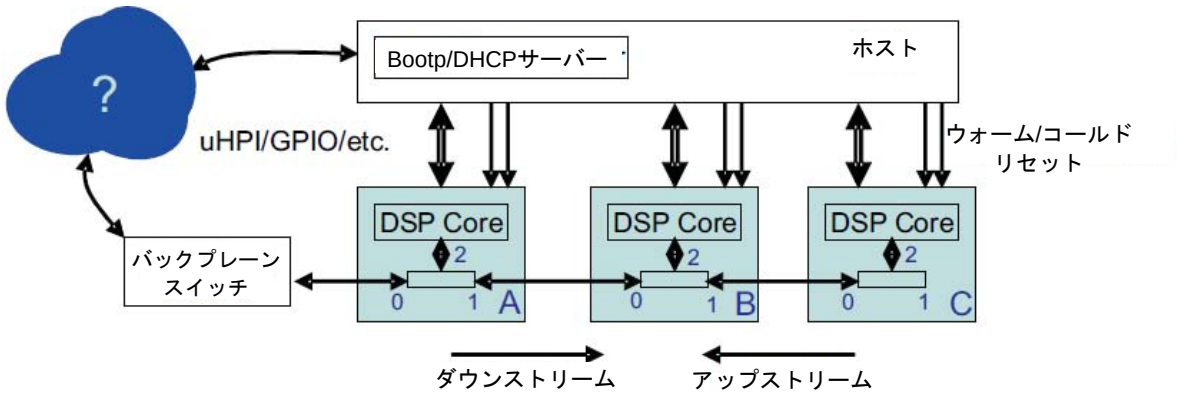


図2. SGMII0ブートにおけるパケット･フロー

表9. イーサネット 3 ポート･スイッチの設定

サブ･モジュール	レジスタ	値	説明
SerDes	CFGPLL	0x00000013	(デフォルト)PLLイネーブル、MPY=x20 (入力が62.5MHzであることを仮定)
	CFGTX0/1	0x00000B21	TXイネーブル、10-bitバス幅, SWING = 5
	CFGRX0/1	0x00089121	RXイネーブル、ハーフレート、10-bitバス幅、信号損失検出イネーブル
CPSW	CONTROL	0x000000E0	ポート0/1 TXフロー制御、ポート2 RXフロー制御
	SL0_SA_LO, SL0_SA_HI, SL1_SA_LO, SL1_SA_HI	デバイスMACアドレス	POR時にEFUSEからラッチされたMACアドレス

1.11.1 イーサネット･ブート-モード･タイミング

SerDes PLLモジュールはデフォルトの設定を想定しています。そのため、デバイス$\overline{\text{POR}}$リセット時に、PLLはデフォルトの倍値 x20 にイネーブルされます。さらに、ブート･ローダは、500-625Mbpsの有効ライン･レートを提供するTX/RXで1/4のレート(4つのPLL出力サイクル毎に1データサンプル)になるようにSerDesモジュールを設定します。

1.11.2 イーサネットのデータ･フォーマット

ブート･ローダは3ポート･スイッチを設定すると、ブート･ポートを介してBOOTPリクエスト･パケットを送信します。表10にBOOTPリクエスト･パケットのフォーマットを示します。ホスト/サーバーはブート･テーブル･フォーマットを用いてアプリケーション･コードを送信することで応答します。そのアプリケーションのブート･テーブルは、ホスト/サーバーからDMPにブート･テーブル･フレームとして送信されます。これは複数のブート･テーブル･フレームが転送される必要があります。各ブート･テーブル･フレームは連続する必要があり、そのフォーマットを表11に示します。

表10. BOOTPリクエスト･パケットのフォーマット

内容	バイト･オフセット	フィールド説明	フィールド値	備考
DIXイーサネット･ヘッダ	5-0	DST MACアドレス	0xFF, 0xFF, 0xFF, 0xFF, 0xFF	
	11-6	SRC MACアドレス	MSB-DMP MACアドレス - LSB-DMP MACアドレス	DMPのMACアドレス
	13-12	タイプ	0x08, 0x00	
IPV4ヘッダ	14	バージョン/長さ	0x45	
	15	タイプ･オブ･サービス(TOS)	0x00	
	17-16	全体の長さ	0x01, 0x48	
	19-18	ID	0x00, 0x01	
	21-20	フラグ/フラグメント･オフセット	0x0000	
	22	Time to LIve	0x10	
	23	プロトコル	0x11	UDPプロトコル
	25-24	ヘッダ･チェックサム	0xA9, 0xA5	
	29-26	SRC IPアドレス	0x000000	
	33-30	DST IPアドレス	0x000000	
UDPヘッダ	35-34	SRCポート	68 (0x44)	
	37-36	DSPポート	9 (0x09)	ROMブート･コードは9もしくは'discard'のパケット･タイプを期待
	39-38	長さ	308 (0x134)	
	41-40	チェックサム	0x00, 0x00	
BOOTPペイロード	42	オペコード	0x01	リクエスト
	43	HWタイプ	0x01	イーサネット
	44	HWアドレス長	0x06	
	45	ホップ･カウント	0x00	
	49-46	トランザクションID	0x12345678	
	51-50	SECONDS数	0x0001	
	53-52	フラグ	0x0000	
	57-54	クライアントIP	0x00000000	
	61-58	DMP IP	0x00000000	クライアントIP, DMP IP, サーバーIP及びゲートウェイIPは全てゼロ
	65-62	サーバー IP	0x00000000	
	69-66	ゲートウェイ IP	0x00000000	
	85-70	DMP HW (MAC) アドレス	MSB-DMP MACアドレス - LSB-DMP MACアドレス	DMPのMACアドレス
	149-86	サーバー･ホスト名	ti-boot-table-svr	
	278-150	ブート･ファイル名	ti-boot-table-0000	
	341-279	ベンダ仕様	デバイスID	DMP デバイスID

表11. イーサネット･ブート･テーブル･フレームのフォーマット

イーサネット･ヘッダ (下記のタイプのうち１つ):	・DIXイーサネット (DMAC, SMAC, type: 14バイト) ・802.3 w/SNAP/LLC (DMAC, SMAC len LLC, SNAP: 22バイト) ・DIXイーサネット w/VLAN (18バイト) ・802.3 w/VLAN and SNAP LLC (26バイト)
IVP4ヘッダ	IVP4 (20~84バイト)
UDPヘッダ	UDP (8バイト)
ブート･テーブル･フレーム･ヘッダ	ブート･テーブル･フレーム･ヘッダ (4バイト)
ブート･テーブル･フレーム･ペイロード	(最小4バイト、最大イーサネット･フレームの最大に制限 – 前のヘッダ)

表12. ブート･テーブル･フレーム･ヘッダ

バイト･アドレス	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	マジック･ナンバー (0x544B)															
0x2	オペコード (0x01)							シーケンス･ナンバー								

ブート･テーブル･フレーム･ヘッダの各エントリは16ビットです。最初のエントリはマジック･ナンバーです。2つ目のエントリは2つの8ビット･フィールドから構成されます。8ビットのオペコード･フィールドは常に0x01にセットされます。8ビットのシーケンス･ナンバーはストリームのパケット順序を示します。シーケンス･ナンバーはブートストリームの最初のパケットにおける数として、常に0で開始すべきです。

ブート･パケットのペイロードはブート･テーブル･データ･ストリームです。ブート･テーブルは表13に示されるフォーマットでなければなりません。ブート･テーブル･データは、アプリケーションのためにブート･テーブル全体を複数のパケット/フレームに分割して送信する必要があります。それぞれのブート･テーブル･エントリは32ビット幅です。

ブート･テーブル･フォーマットは、IPV4及びUDPヘッダを含んだイーサネット･フレームでカプセル化されます。以下の節は受理可能なイーサネット･フレームについて説明します。ブートローダが下記で指定された基準に従っていないフレームに遭遇した場合、それらは何事も無かったように破棄されます。続くフレームは通常通り処理されます。DIXや802.3 MACヘッダ･フォーマットの両方を用いているフレームは、VLANタグの有無に関わらずフレームとして受理されます。どのソースMACアドレスも受理されます。はじめにブート･ローダはデバイスのEFUSEから読み出したMACアドレスとなるようにディスティネーションMACアドレスをセットします。これは、ブートが完了するまで使用されるディスティネーション･アドレスです。VLANフィールド(type/len以外の)は無視されます。802.3 MACフォーマットが使用される場合、SNAP/LLCヘッダは照合され、スキップされます。typeフィールドはIPV4タイプ(0x0800)を選択します。IPV4ヘッダはVersion(4)やフラグ及びフラグメント･フィード、プロトコル(UDP)フィールドを有効にします。ヘッダ長フィールドはヘッダ･オプション･ワードを適切にスキップするために解析されます。いかなるソース及びディスティネーションIPアドレスも受理されます。UDPヘッダはソース及びディスティネーション･ポート番号がブート･パラメータで指定されたIPアドレスに一致することを認証します。ブート･パラメータ･ソース･ポート･フィールドが0の場合、いかなるソース･ポートでも受理されます。UDPヘッダ長は、適切に調整されたフレーム長が正しいかどうかの判断をテストされます。UDP長がフレームに対して長すぎる、もしくは２の倍数でない場合、フレームは破棄されます。UDPチェックサムが検証され、UDPチェックサム･フィールドがゼロ(0)ならば、不正なUDPチェックサムを持つフレームは破棄されます。

表13. ブート･テーブルのフォーマット

バイト･オフセット	エントリ
0x0000 0000	エントリ･ポイント･バイト･アドレス (コードがロードされた後に分岐する開始アドレス)
0x0000 0004	セクション1のサイズ(バイトで)
0x0000 0008	セクション1のロード･アドレス(バイトで)
0x0000 000C	セクション1のデータ
....	....
....	....

....	セクション2のサイズ(バイトで)
....	セクション2のロード･アドレス(バイトで)
....	セクション2のデータ
....	...
....	セクションNのサイズ(バイトで)
....	セクションNのロード･アドレス(バイトで)
....	セクションNのデータ
....	....

2 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ヘッダはマジック･ワード(0x41504954)から構成されています。ヘッダの後は、図3 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。

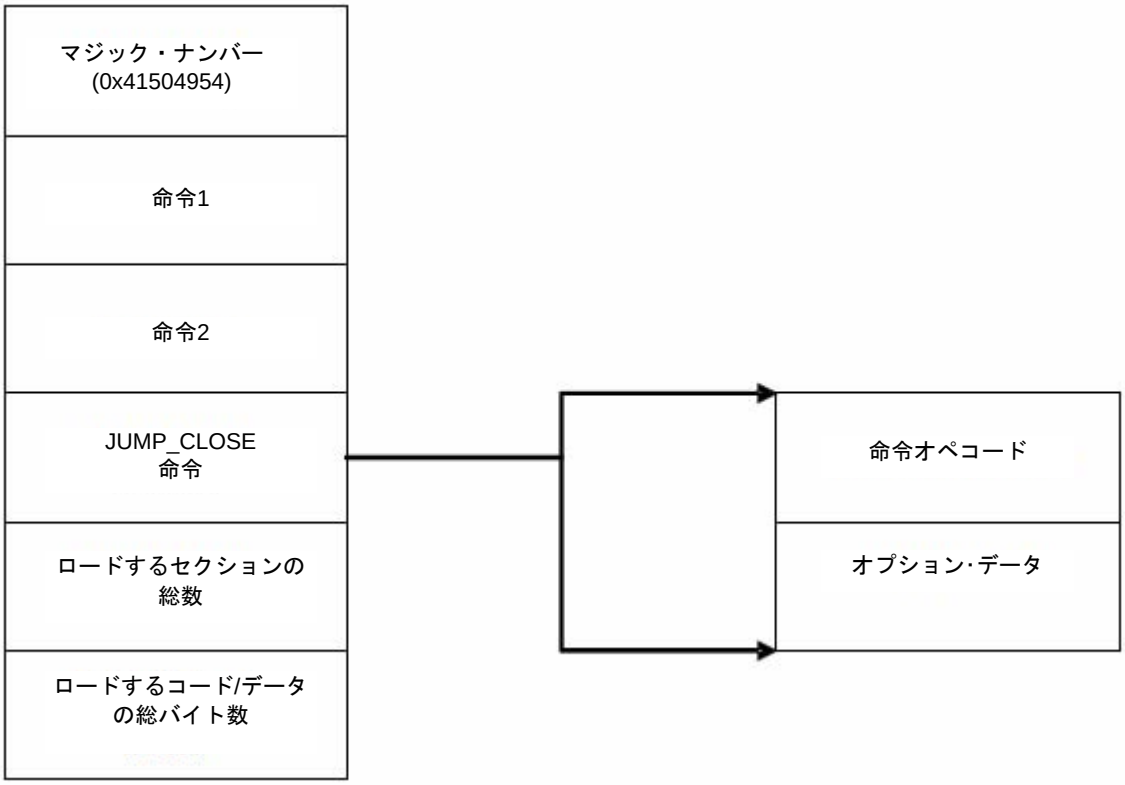


図3. アプリケーション･イメージ･スクリプトの基本構造

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14 にAIS 1.0でサポートされる命令を示します。

表14. 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･オーバー	0x58535908
予約	0x58535909
セクション･フィル	0x5853590A
ゲット	0x5853590D
関数実行	0x5853590D

2.1 SET 命令

SET命令はDMP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DMP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。
DMP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4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4. GET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DMP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15 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15. 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16 に使用できるデータ･タイプ一覧を示します。

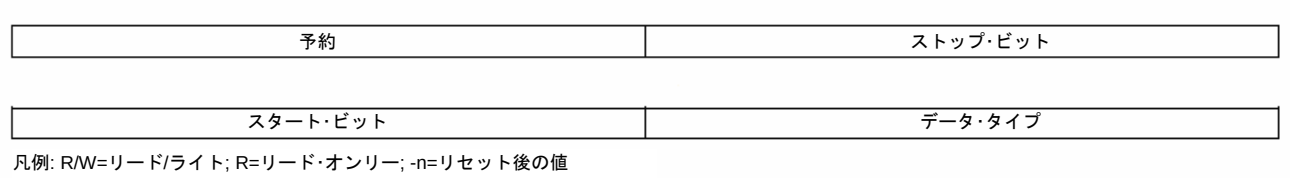
表16. 有効なSET命令のデータ･タイプ

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(表16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14 に32ビット<Type> の符号化を示します。

2.1.1 有効な SET 命令のデータ･タイプ

図5. 有効なSET命令のデータ･タイプ



凡例: R/W=リード/ライト; R=リード･オンリー; -n=リセット後の値

表17. 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 (0,1,2,3,4)、ライトするデータのタイプを指定

2.2 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2.1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6 にGET命令の構造を示します。

図6. GET命令の構造

各ブート･テーブル･コマンドはSET(0x58535907)オペコードから成り、この後ろに以下に示される追加データの3つのワードが続きます。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>

2.3 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7 にセクション･ロード命令の構造を示します。

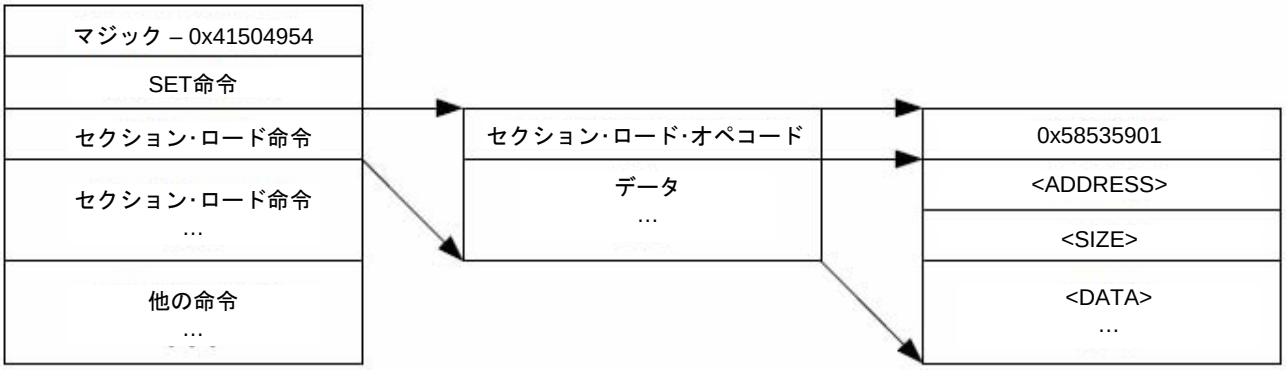


図7. セクション･ロード命令の構造

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。

2.4 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8 にセクション･フィル命令の構造を示します。

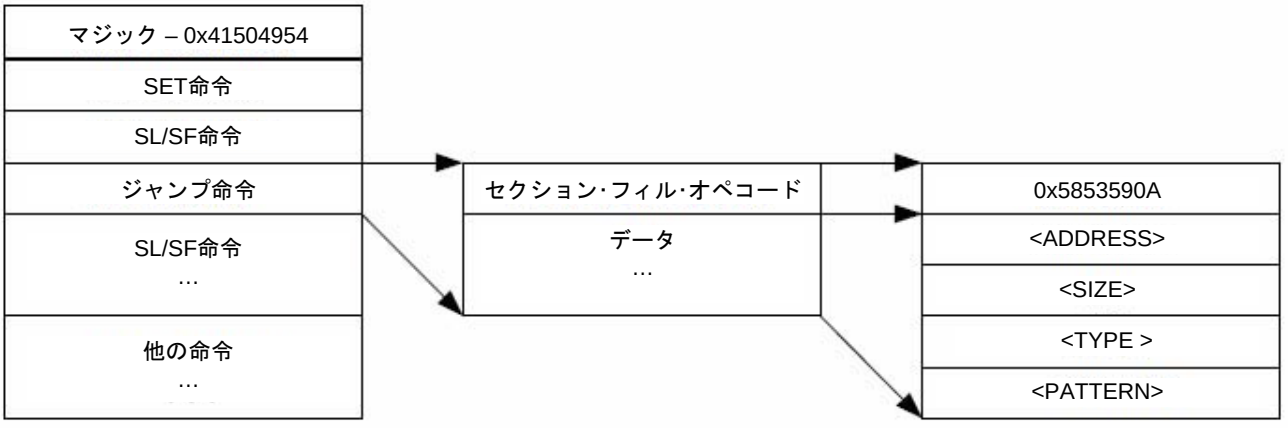


図8. セクション･フィル命令の構造

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。

2.5 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9 にジャンプ命令の構造を示します。

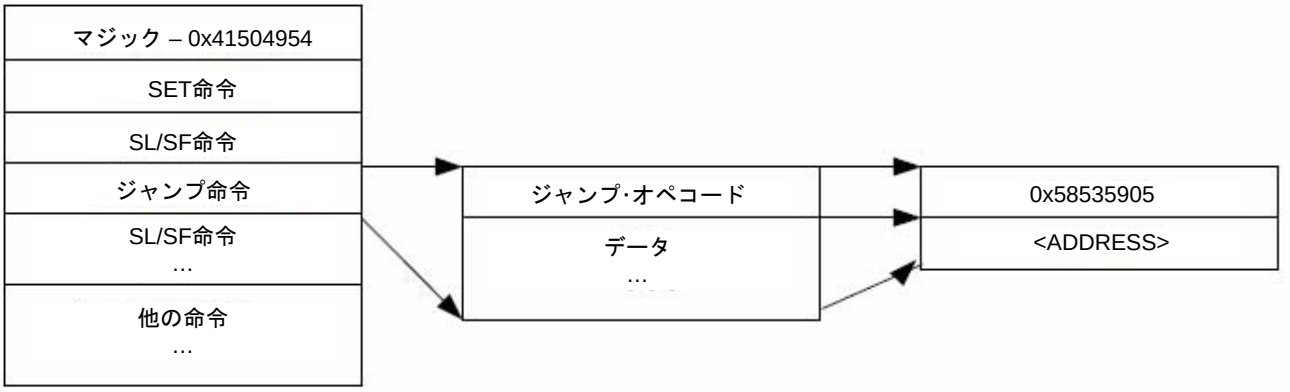


図9. ジャンプ命令の構造

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。

2.6 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DMP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10 にジャンプ･クローズ命令の構造を示します。

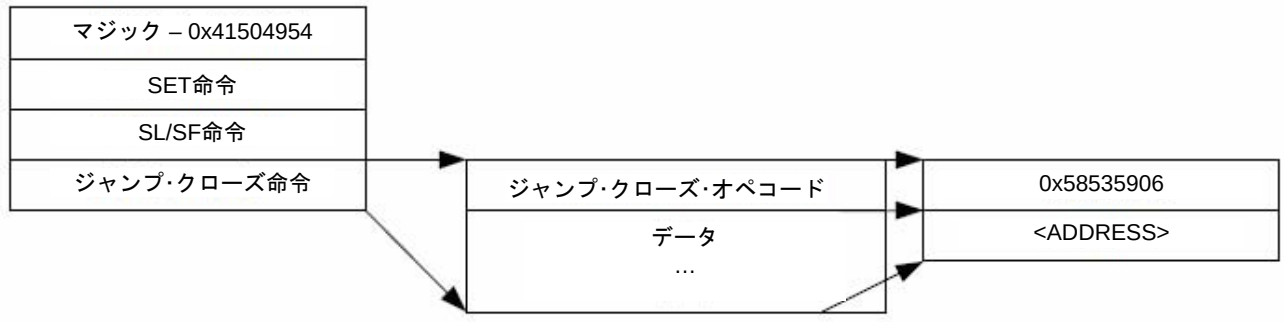


図10. JUMP_CLOSE命令の構造

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。

2.7 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、エラーを検出または訂正することはできません。

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。

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。

2.7.1 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11 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。

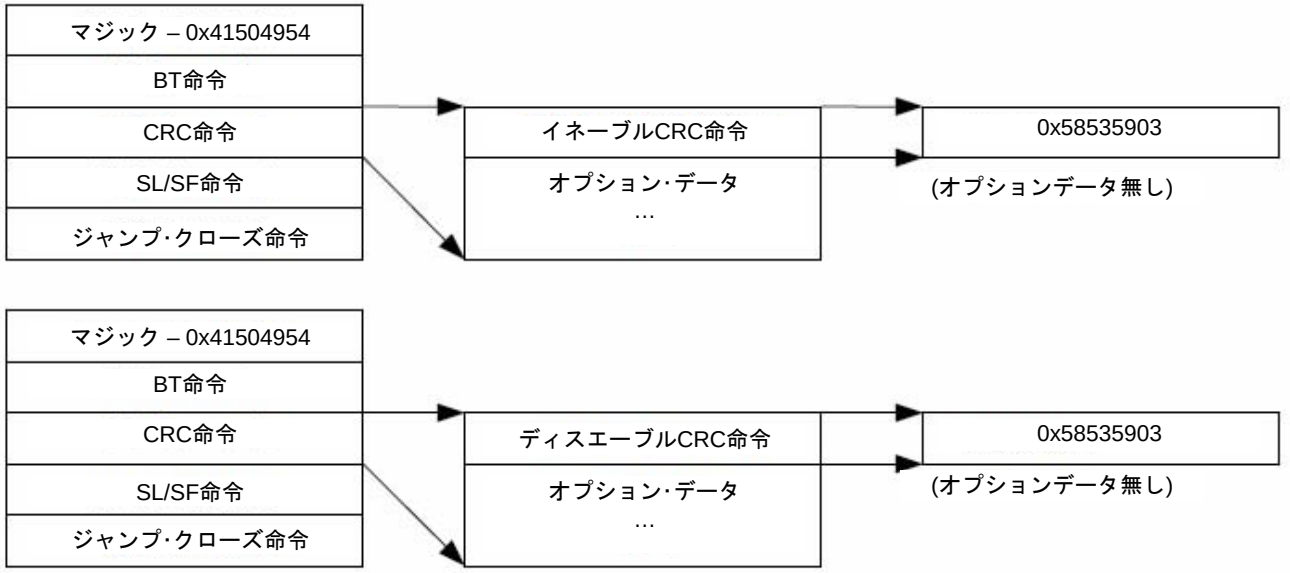


図11. イネーブルCRC/ディスエーブルCRC 命令の構造

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。

2.7.2 リクエスト CRC 命令

この命令はDMP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DMP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、ホストは以下で説明するスタート･オーバー命令を使ってDMP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、DMP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12 にリクエストCRCコマンドの構造を示します。

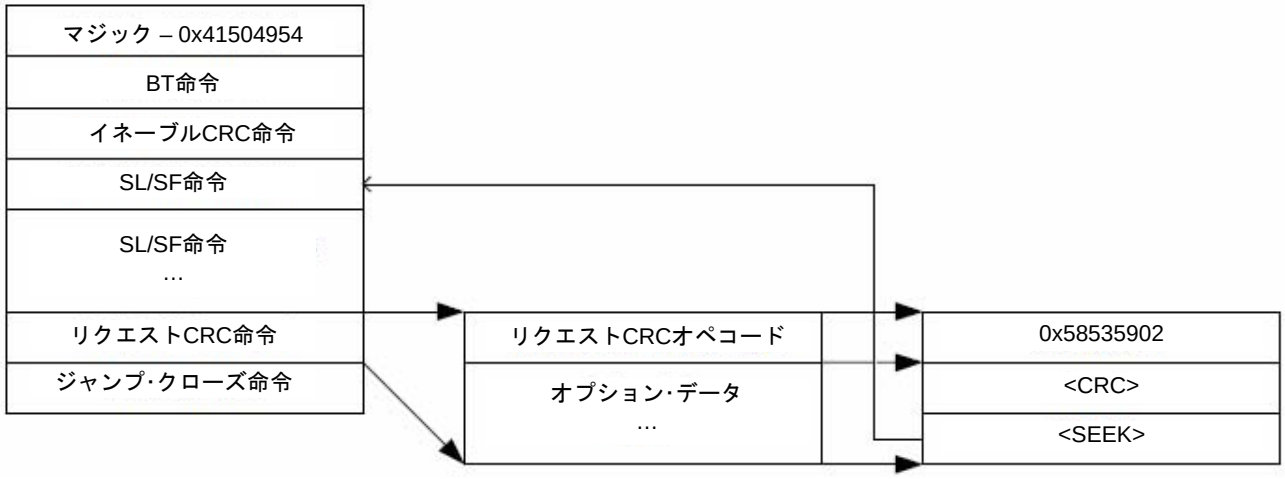


図12. リクエストCRC命令の構造

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12　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12 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。

2.7.3　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(とオペコード)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ることに注意してください。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　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DMPメモリに残りをブートします。

4 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0x01BFB000-0x01BFFFFF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。

5 AIS 生成ツール、genAIS

genAIS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。

genAIS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、

```
genAIS -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。genAIS ツールの使用可能なオプションを 表18 に示します。

表18. genAISプログラムのオプション

オプション	詳細
-I	入力実行ファイルの名前を指定
-o	AIS出力ファイルの名前を指定
-opath	出力ファイルの出力先のパスを指定. デフォルト入力ファイルと同じパス
-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, enet, raw)	生成されるブート･モードを指定: • Raw はモード独自のAISイメージを生成します. • このユーティリティでサポートされるブート･モードがデバイスではサポートしていない場合があります. サポートされているブート･モードの詳細は、デバイスのデータシートもしくはブート･ローダの資料を参照してください
-otype N (N=ascii, binary, asm, cfile, txt)	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、DM647及びC6423/C6421デバイスのI2C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
-srcipaddr	000.000.000.000形式による転送元IPアドレスの指定. 全ての数字は10進数フォーマット
-dstipaddr	000.000.000.000形式による転送先IPアドレスの指定. 全ての数字は10進数フォーマット
-srcmacaddr	000.000.000.000.000.000形式による転送元MACアドレスの指定. 全ての数字は10進数フォーマット
-dstmacaddr	000.000.000.000.000.000形式による転送先MACアドレスの指定. 全ての数字は10進数フォーマット
-htype N (N=dix, snap, vlan, vsnap)	イーサネット･ヘッダ･タイプ
-packetsize	(ヘッダを含まない)イーサネット･ブート･パケット･ペイロードのバイト･サイズ 4で割り切れる均等な値である必要があり、最大値は1400です

6 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ストリームをこの章で示します。この章で扱うAISブート･イメージはgenAISを使って生成されました。genAISは5章で議論されています。この章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
||              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
```

6.1 EMIFAROM 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19にこの章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19. 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

6.2 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20に示します。

表20. 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21 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21. 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

6.3 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。

表22. 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
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23 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23. 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

6.4 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、DMP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。DMP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、DMP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、DMP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。

DMP				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13 参照)

```
41504954585359035853590110800000000000400180202802802428020022280188406902000320A020C027702884068028C1FDB020840686C6E10CD10442641003C2C6E45B06C6E2C6E00B48C6E008AEFC08000585359020E85A97BFFFFFFA858535901108000040000000C0000000A0000000B0000000C585359028434A250FFFFFFDC5853590610800000000000000020000004C
```

図13. UART AISブート･イメージ

6.5 イーサネット･ブート･モードのためのブート･パケットの生成

いくつかのオプションがイーサネット･ブート･モード用にフォーマットされたブート･パケットを生成するために必要です。そのリストを表24に示します。genAISツールは、生成された各パケット用のデータを含む別々のCソース･ファイルを作成します。また、各パケットに対する外部宣言やサイズ情報を含んだCヘッダ･ファイルも作成されます。これらのファイルは、DM647/8のブートにおいてブート･パケットを送信するホスト/サーバー･アプリケーションのビルドに用いることができます。

表24. イーサネット･ブート･パケット生成オプション

オプション	説明
-srcipaddr	000.000.000.000形式による転送元IPアドレスの指定. 全ての数字は10進数フォーマット
-srcmacaddr	000.000.000.000.000.000形式による転送元MACアドレスの指定. 全ての数字は10進数フォーマット
-dstipaddr	000.000.000.000形式による転送先IPアドレスの指定. 全ての数字は10進数フォーマット
-dstmacaddr	000.000.000.000.000.000形式による転送先MACアドレスの指定. 全ての数字は10進数フォーマット
-htype	イーサネット･ヘッダ･タイプ (dix, vlan, snap, vsnap)
-packetsize	パケットに対するペイロードのバイト･サイズ. ヘッダを含まず、最大1400バイトまで許容

AISツールを用いた起動サンプルを以下に示します:

genAIS -otype cfile -l my.out -bootmode enet -packetsize 1400 -srcipaddr 11.22.33.44 -dstipaddr 55.66.77.88 -srcmacaddr 123.156.158.218.23.12 -dstmacaddr 123.156.158.218.23.14 -opath myOutputPath

例2. パケットを伴うCヘッダ･ファイルのサンプル

```
//-------------------------------------------------------------
// This is an auto-generated file:
//   Creation Time Stamp:
//     Nov, 13, 2007, 13:47:10
//   Created By User: a0321848
//   Created from .out file : docExample.out
//-------------------------------------------------------------
#ifdef __PACKETHDR__
            #define __PACKETHDR__
            #include <tistdtypes.h>
            #ifndef NULL
            #define NULL (0L)
            #endif


//-------------------------------------------------------------
// Define Total Number of BOOTP Packets
//-------------------------------------------------------------
            #define TOTAL_NUM_BOOTP_PACKETS      1


//-------------------------------------------------------------
// External Packet Array Declarations
//-------------------------------------------------------------
            #define SIZE_PACKET0      144
            Extern Uint32 packet0[];


//-------------------------------------------------------------
// Initialize Packet Size Array
//-------------------------------------------------------------
            Uint32 packetSize[TOTAL_NUM_BOOTP_PACKETS] = {
                                    144
            };
#endif
```

ユーティリティは、作成される全てのデータ･パケットに対して外部宣言及びサイズを含んだCヘッダ･ファイルを出力として作成します。各パケットのデータは、拡張子が".cpp"の1つのCファイル内で定義された2次元配列に格納されます。最初のパケットは、DMPにダウンロードされるとアプリケーション･コードに分岐するアドレスを含んだpacket 0になります。genAISで*-opath*オプションが使用した場合、ヘッダとパケットファイルは指定されたディレクトリに配置されます。指定しなければ、これらのファイルは入力ファイルと同じディレクトリに配置されます。

その他に-otype txt を指定する方法があります。AIS生成ツールは、全てのパケット･データの線形リストを含む単一ASCII".txt"ファイルを作成します。生成されたパケットの長さや数を示したCヘッダ･ファイルは常に作成されます。この情報は、テキストファイルを解析し、適切なパケットとしてDMPに送信されます。

例3. AISユーティリティで作成されるパケットのサンプル

```
Uint32 packet[][] = {
                0xDA9E9C7B,
                0x9C7B0E17,
                0x0C17DA9E,
                0x00450008,
                0x00008400,
                0x11100000,
                0x160B7206,
                0x42372C21,
                0XA8AB584D,
                0x70000000,
                0x544B7C34,
                0x40000100,
                0x004010F0,
                0x40000000,
                0x202810F0,
                0x242801A0,
                0x78690220,
                0x032A0188,
                0x02770200,
                0x7868020C,
                0x1FDB0288,
                0x7868028C,
                0x10CD0208,
                0x26416C6E,
                0x2C6E1044,
                0x6C6E003C,
                0x00B445B0,
                0x008A2C6E,
                0x80008C6E,
                0x000CEFC0,
                0x40400000,
                0x000A10F0,
                0x000B0000,
                0x000C0000,
                0x00000000,
                0x00000000
     };
```

6.6 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。

7 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0x0080ff00番地を読むことにより確認することができます。バージョン 0x00003060とバージョン 0x00003070の2つのバージョンが現在存在しています。バージョン0x00003070は、I2C(DMPマスタ)やUART(DMPスレーブ)、イーサネット(DMPスレーブ)のブート･モード全てが組み込まれています。これらのモードはバージョン0x00003060では全てを使用できません。

8 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