C641x/DM64x DSP ブート用プロジェクトの作成方法

江成 広樹　　DSP/MSP430 プラットフォームグループ

アブストラクト

DSPは様々なアプリケーションを処理するためにプログラムを必要とします。しかし、TMS320C641x/DM64x DSPはROMを内蔵していないため、プログラムを最初から保持しているわけではありません。システムを動作させるためには、事前に実行するプログラムを何らかの方法でDSPの持つ内部/外部のメモリー空間に書き込む必要があります。

本資料では、テキサス・インスツルメンツ社(以下: TI)のDSP の TMS320C6000 シリーズである TMS320C641x/DM64x DSPにおける外部メモリー空間に接続されたROMからプログラムをブートする仕組みと、開発ツールである Code Composer Studio(以下: CCS) 及び プラグインツールであるFlashBurn を用いたROMからブートするためのプログラミング方法をご紹介します。

目次

図

1　ブートの概要

C6000 DSPは、リセット信号解除の際、特定のピンに入力する信号の組み合わせにより、DSPのブート･コンフィグレーションとデバイス･コンフィグレーションが決定され、動作を開始します。

ブート･コンフィグレーションとは、リセット信号解除後、DSPコア(CPU)が実行を開始する前にEDMAによって処理されるデータ/プログラムのコピー動作の選択です。デバイス･コンフィグレーションとは、エンディアンやクロック･モード、ペリフェラルの選択のことを示します。

本資料では、ブート･コンフィグレーションにより決定されたブートに対応するプログラムの作成方法についてご紹介します。

※ブート･コンフィグレーションとデバイス･コンフィグレーションの詳細は、各デバイスのデータシートをご参照下さい。

1.1　ブート･コンフィグレーションの種類

C6000 DSPではリセット解除後、指定されたブート･コンフィグレーションによって、システムを動作させるためのプログラムやデータをしかるべき場所にコピーするブートという作業を行ないます。TMS320C641x/DM64x DSPのブートコンフィグレーションでは、1. ブートなし、2. ホスト･ブート、3. ROMブートの3種類から選択できるようになっています。

1.1.1　ブートなし

ブート動作は行われません。したがって、DSPにリセットがかかる前に、データ/プログラムを実行前にあらかじめしかるべきメモリー空間に配置しておく必要があります。

1.1.2　ホスト･ブート

このブート･モードが選択されていると、CPUコアはリセット状態を保ちます。この状態で外部ホスト接続用ペリフェラル(HPI/PCI)に接続したホスト･プロセッサから、内部/外部問わずC6000 DSPの持つ全てのメモリー空間へのデータ/プログラムの転送が可能となります。

ホスト･プロセッサによって、必要とされる、もしく全てのデータの転送が完了したら、DSPに対してDSPINTというホストからDSPへの割り込み信号をセットし、リセットを解除する必要があります。

1.1.3　ROM ブート

C6000 DSPの外部メモリー･インターフェイス(以下:EMIF)のCE1空間に接続されている不揮発性メモリーの先頭からDSPの内部メモリー空間のゼロ番地に固定サイズ(1KB)分のデータ/プログラムのEDMA転送が自動的に行われます。

※ 外部メモリー･インターフェイス(EMIF)を2基搭載しているデバイスの場合には、2基目の外部メモリー･インターフェイス(EMIF B)のCE1(BCE1)に接続されているメモリーからブートされます。

いずれの場合にもC641x/DM64x DSPのCPUコアは、ブート完了後、ゼロ番地から実行を開始するため、ゼロ番地には必ずプログラムが必要になります。しかし、図1のメモリー･マップに示されるように、C6000 DSPのゼロ番地には内部SRAMが割り当てられています。そのため、リセット直後ではゼロ番地にプログラムが存在しない(不定値である)ので、1. ブートなしの場合は、正常に動作を開始することができません。通常、電源投入時のブートは2, 3のどちらかの方法を用いる必要があります。そして、しかるべきタイミングでブートピンの極性を変えることでブートなしに変更することが可能です。しかし、この方法は、2回目以降のリセットが電源の再投入ではない外部からのリセット時に限定されます。

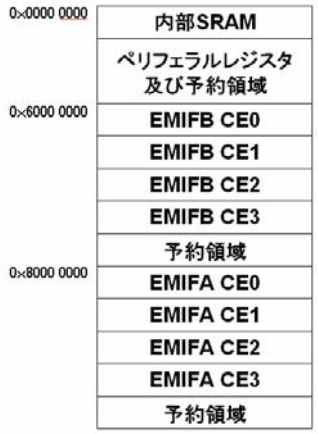


図 1.　C6414(T)/15(T)/16(T) DSPのメモリー･マップ

2　ROM ブート用プログラムの作成方法

ホスト･プロセッサからデータやプログラムを送るホスト･ブートでは、ホスト･プロセッサからC6000 DSPのメモリー空間に必要なデータやプログラムを全て送ることが可能です。

ROMからデータやプログラムを送るROMブートでは、アドレス生成ハードウェア(EDMA)により、自動的に特定ROM領域の先頭から1KBが内部SRAMのゼロ番地から1KBの領域にコピーされます。

本資料では、後者のROMブートを行なうための方法についてご紹介します。

2.1 ROM ブートするために必要なもの

既存のプロジェクトをROMブートから起動するプロジェクトに修正(変更)するためには、少なくとも既存のプロジェクトの他に、ブート・コード(例: boot.asm)とこのブート・コードをROMの先頭に配置するためのリンカー・コマンド・ファイル(例: link.cmd)、実行ファイルを変換するhex変換用コマンド・ファイル(例: hex_6416.cmd)を用意する必要があります。この3つのファイルについて詳細を説明します。

実際のCCS上での手順やファイルの詳細は、第3章とサンプル・コード(SPRAAF7.zip) をご覧ください。

2.1.1 ROM ブート・コードの作成

ROMブートは、ハードウェアにより指定された外部ROMの先頭から1KBが内部SRAMのゼロ番地に自動でコピーされます。しかし、多くの場合、ユーザのプログラムやデータが1KBを超えるため、このハードウェアによるブートだけでは全てのプログラムやデータをメモリー空間にコピー(展開)することができません。このような場合、ハードウェアによってコピーされる1KBのプログラムには、まだROM上に残っているその他のプログラムやデータを適切なメモリー(内部SRAMやその他の外部メモリー)空間にコピーするルーチン、つまりソフトウェアによるブート・コード(セカンダリ・ブートローダーと呼ぶこともあります)を記述する必要があります。このブート・コードで記述すべき基本的な処理を以下に示します。

1. EMIFの設定
2. ROMからRAMへのデータコピー
3. C環境初期化ルーチン(_c_int00)へ分岐

ハードウェアによるブート直後、つまりリセット直後の外部メモリー・インターフェイス(EMIF)は非同期式かつタイミングが一番遅い設定になっています。これからROM上に残っているプログラムやデータをコピーする必要があるため、システムに合ったメモリー設定を行なっていなければ、データのコピーが正しく行なわれず、最悪の場合にはシステムが動作しないということが起こり得ます。そのため、EMIF設定をブート直後に行なわなければなりません。その設定後に初めて、しかるべきセクション(データ/プログラム)をコピーすることができ、C環境を初期化するルーチン(_c_int00)へ分岐する必要があります。

_c_int00は、TIが提供するC環境の初期化ルーチンであり、グローバル/スタティック変数の初期化やスタック・ポインタのセット、DSP/BIOSの初期化などを行ないます。初期化後はユーザのmain関数を自動的にコールします。この動作はデフォルトでイネーブルされていますが、ビルド・オプションのリンカー・オプション “-c” もしくは、”-cr” で制御されます。

このブートコードは、このコード内でC環境の初期化ルーチンに分岐することからわかるように、アセンブリ言語で記述する必要があることに注意してください。

2.1.2 リンカー・コマンド・ファイルの作成

ブート・コードはハードウェアにより内部メモリーの先頭にコピーされます。しかし、この領域に他のセクションが割り当てられてしまっていると、ブート・コードのセクションコピー実行中に内容が上書きされ動作しなくなってしまいます。そのため、ゼロ番地から1KBの領域をブートコード用に保護かつ配置するために、作成したブート・コードにユーザ定義のコード・セクション(例: .boot_load)を割り当て、そのセクションを内部メモリーのゼロ番地に配置するためのユーザ定義のリンカー・コマンド・ファイルを作成する必要があります。

これはコンフィグレーション・ファイルが自動生成するリンカー・コマンド・ファイルとは別に必要となりますので、注意してください。

2.1.3 Hex 変換用コマンド・ファイルの作成

作成したブート・コード(boot.asm)とリンカー・コマンド・ファイル(link.cmd)を既存プロジェクトに追加することで、ROMブートを行なうために必要なファイルが全て揃った最終的なプロジェクトになります。しかし、このプロジェクトをビルドして生成される実行ファイル(*.out)をROMに直接書き込むことはできません。そこで、この実行ファイルを書き込み可能なHEXファイル(*.hex)に変換する作業が必要となります。

CCSはこの変換を行なうhex変換ユーティリティ(hex6x.exe)を付属しており、使用するROMの仕様やプログラムを書き込むホスト・プロセッサ等に合わせて、適切なフォーマットに実行ファイルをhex変換することができます。その作業にHex変換用コマンド・ファイル(リンカー・コマンド・ファイルとは異なります)が必要となります。図2にhexコマンド・ファイルの記述例を示します。

```
.¥Debug¥example.out                          /*hex変換する実行ファイルの指定*/
-a                                           /*ASCII フォーマット変換の指定*/
-image                                       /*メモリー空間全体のイメージを作成*/
-zero                                        /*先頭アドレスを 0 にリセット*/
-memwidth 8                                  /*ROMのデータ幅の指定*/
-map .¥Debug¥examplehex.map                  /*マップファイルと出力先の指定*/
-boot                                        /*全初期化セクションのhex変換を指定*/
-bootorg 0x64000400                          /*セクションテーブルの配置アドレスの指定*/
-bootsection .boot_load 0x64000000           /*ブートセクションと配置アドレスの指定*/

ROMS                                         /*書き込むROMの情報とファイルの指定*/
{
    FLASH: o=0x64000000, l=0x80000, romwidth=8, file={.¥Debug¥example.hex}
}
```

図 2. hexコマンド・ファイルの記述例

ここでは、3つの -boot, -bootorg, -bootsection オプションについて詳細を説明します。その他のオプションの詳細は、[TMS320C6000 Assembly Language Tools User's Guide (SPRU186)]を参照してください。

Option 1: -boot

全ての初期化されたセクション(.text, .cinit, .sysinit等)のHex変換を行ないます。また、-bootsectionで指定されたユーザ・ブートコード・セクション(つまり、boot.asm)を除いたその他の各セクションを1つにまとめたセクション・テーブルを作成します。その際、セクション・テーブルの先頭には、ユーザのブート・コードによってセクションのコピーが終了した後に分岐するC環境初期化ルーチンのアドレス(デフォルト: _c_int00)が挿入され、さらに、各セクションの先頭にはブート・コードでコピーするために必要なセクションのサイズと転送先アドレスがヘッダ情報として付加されています。

Hex変換する対象をセクション単位で指定したい、もしくはROM上に配置されるセクションの順番を制御したい場合は、-bootオプションの代わりに、ROMSキーワードの後にSECTIONSキーワードを用いてセクションを指定してください。しかし、このキーワードを使用した場合は、各セクションの先頭にヘッダ情報が付加されません。そのため、ブート・コード内に必要な初期化セクションのセクション・サイズと転送先アドレスのテーブルを適当な値で用意します。それから、ビルド時に出力されるマップファイルから正しいロード・アドレスやラン・アドレス、セクション・サイズを読み取り、テーブルを適切な値に変更してリビルドすることで最終的な実行ファイルおよびHexファイルを生成しなければなりません。

Option 2: -bootsection [addr]

指定したアドレス[addr]にエントリー・ポイント※(デフォルト: _c_int00)を挿入し、次(4バイト後)のアドレスから -bootオプションで作成されたセクションテーブルを配置します。基本的にこのアドレスは、EMIF CE1空間の先頭から1KB後の 0x6400 0400 もしくは 0x9000 0400 になります。また、ここで指定したアドレスは、セクションのコピーを行なうソース・アドレスとしてブート・コード内で使用するので、変更する際はブート・コードにも修正が必要になります。

※ エントリー・ポイントとは、プログラムの実行が始まるアドレスのことを指します。このエントリー・ポイントは -eオプションで即値アドレスやグローバル・シンボルを使用したアドレスの指定が可能です。

Option 3: -bootsection [sect] [addr]

このオプションは、ブート・コードのセクション[sect]を指定したROM上のアドレス[addr]に正しく配置するためのものです。デバイスによって異なりますが、アドレスはEMIF CE1空間の先頭アドレスである 0x6400 0000 もしくは 0x9000 0000になります。

これらの3つのオプションを含む図2のコマンド・ファイルを使用して生成されるhexファイルのROM上におけるイメージを図3に示します。

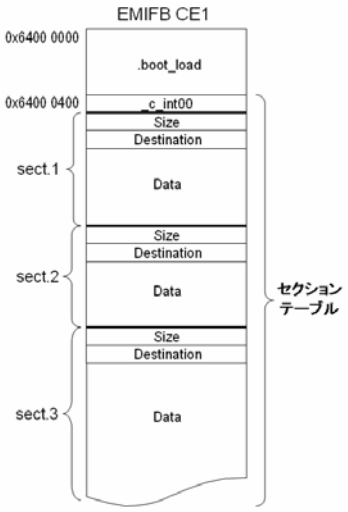


図 3. ROMイメージ

2.2 ブートコードと Hex 変換コマンドファイルとの関係

ブート・コード内で記述するセクションのコピー・ルーチンは、-bootオプションによって挿入される各セクションのヘッダ情報と、-bootorgオプションで指定した転送元のセクション・テーブルの先頭アドレスを用いることで、容易に記述することができます。ブート・コードのフローチャートを図4に示します。

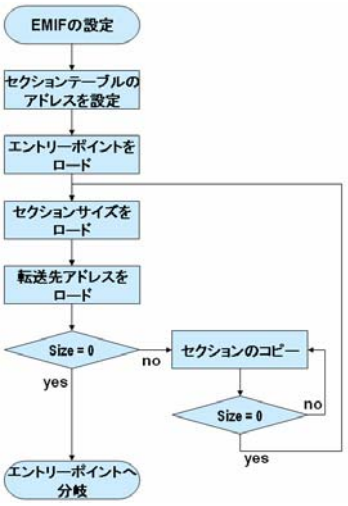


図 4. ブート・コードのフローチャート

3 フラッシュメモリーへのプログラミング方法

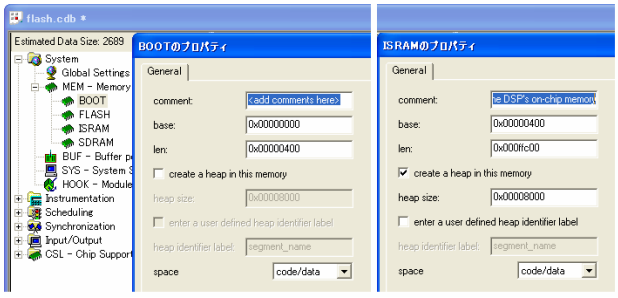
ROMには、EPROMやEEPROM、フラッシュメモリー等といった様々なメモリーが存在します。ここではTIが提供しているC6416 DSPが搭載されたDSP スタータ・キット(以下:DSK)を例に、DSK上のフラッシュメモリーへのプログラミング方法について説明します。この例で使用するプロジェクトは、McBSPの割り込み信号を用いたオーディオ・スルーのプログラム(flash.pjt)になります。

これから作成手順を紹介するにあたって、用いる開発環境について説明します。C6416DSK上のフラッシュメモリーへのプログラミングを行なうために、TI DSPの開発ツールとして、Code Composer Studio version3.1 とCCSのプラグインツールであるFlashBurn2.80を使用しています。また、使用しているファイルは特に断りがない限り、プロジェクトファイル(*.pjt)のあるディレクトリ下に全て保存されているものとします。開発ツールにおきましては、CCS ver2.20以降、FlashBurn2.7xでも同様の手順でプログラミング可能です。その際は、CCSのインストール・ディレクトリが異なることに注意してください。

詳細はサンプルコード(SPRAAF7.zip) をご覧ください。

3.1 Code Composer Studio での手順

1. 既存のプロジェクト(flash.pjt)を開きます。
2. boot_6416.asmをプロジェクトに追加します。
3. コンフィグレーション・ファイル(flash.cdb)を修正します。
   a. [MEM – Memory Section Manager]でブート用のメモリー・セグメント"BOOT"を作成します。
   b. BOOTのプロパティを開き、baseとlengthを0x0, 0x400 に設定し、[create a heap in this memory]のチェックボックスのチェックを外します。
   c. 内部メモリーの先頭1KBをBOOT領域として作成したので、元の内部メモリー・セグメント "ISRAM"のプロパティを開き、先頭から1KB分削除します。



4. boot_6416.asm(".boot_load"というユーザ・セクションを定義)が正しくBOOT領域に配置されるようにユーザ定義のリンカー・コマンド・ファイル(link.cmd)を作成します。

```
SECTIONS
{
      .boot_load   :>  BOOT
}
```

5. ビルド・オプションを開き、[General]タブにある[Final Build Steps]に次のオプションを追加します。

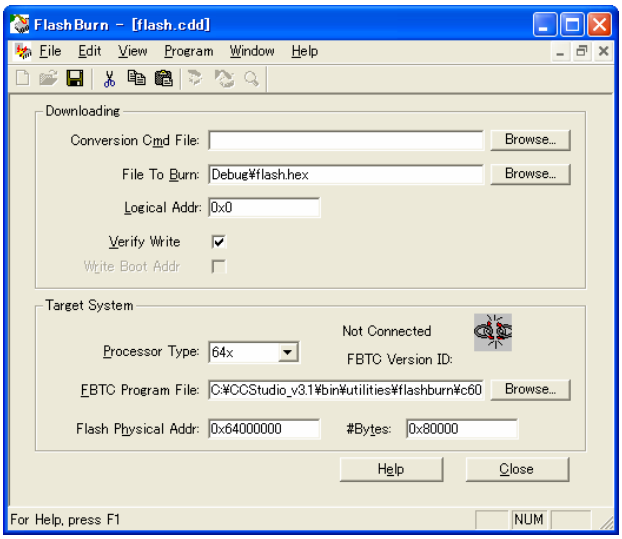
```
hex6x   $(Proj_dir)\hex_6416.cmd
```

6. メニュー: Options → Customizeを選択し、[Program Load Option]タブにある[Load Program After Build]のチェックを外します。
7. メニュー: Debug → Rebuild All でプロジェクトをリビルドし、Debugフォルダにflash.hexが生成されていることを確認します。

CCSウインドウはそのままの状態にしておきます。

3.2 FlashBurn での手順

8. CCSウインドウのメニュー: Tools → FlashBurnを選択し、FlashBurnウインドウを開きます。
9. ファイルを新規作成し、以下の項目を設定します。
   File To Burn : Debug\flash.hex
   FBTC Program File :
   <ccs_install_dir>\bin\utilities\flashburn\c6000\dsk6416(v3)※\FBTC6416.out
   ※ FlashBurnのバージョンによって異なります。
   Flash Physical Addr : 0x64000000
   #Bytes : 0x80000



10. 適切な名前(flash.cdd)でプロジェクトと同じディレクトリにファイルを保存します。
11. メニュー: Program → Download FBTCを選択し、指定した実行ファイルがCCSを経由してDSPにダウンロードし、FlashBurnとターゲットデバイスが接続することを確認します。もし接続しない場合は、CCSウインドウに戻り、一度プログラムを実行してください。自動的にプログラムが停止し、FlashBurnウインドウに戻ると、接続表示に変わっています。
12. メニュー: Program → Erase Flash / Program Flashをクリックし、フラッシュメモリーに対してデータの削除と書き込みを実行します。
13. 書き込みが終了したら、ツールを全て閉じます。
14. DSKをリセットし、オーディオが聞こえることを確認します。

3.3 出荷時の状態に戻す方法

出荷時のDSK上のフラッシュメモリーには、DSK上のLEDを点滅させ、スピーカーが接続されていればサイン波が1秒間出力されるプログラム(POST: Power On Self Test)が書き込まれています。もしDSKを出荷時の状態に戻したい場合には、以下の手順を踏んでフラッシュメモリーを書き直してください。

1. CCSを起動し、メニュー: Tools → FlashBurnを選択し、FlashBurnウインドウを開きます。
2. <ccs_install_dir>\examples\dsk6416\bsl\post にあるpost.cddファイルを開きます。
3. FlashBurnとターゲット・デバイスが接続していることを確認してから、Erase/Programを行なってください。
4. 書き込みが完了したら、ツールを全て閉じ、リセットしてPOSTが行なわれることを確認したら終了です。

4 C671x デバイスのブート方法

C671x DSPにおけるブートは、基本的にこれまで紹介したC641x/DM64x DSPのROMブート作成方法と大きく異なる点はありません。ここでは、デバイスの仕様などによる変更点についてのみご紹介します。

4.1 ブート・コンフィグレーションの種類

C671x DSPのブート・コンフィグレーションには、ブートなしモードとPCIによるホスト・ブート(PCIを搭載していないため)を除く、HPIによるホスト・ブートとROM(8~32bit)ブートの2種類になります。

4.2 ブート・コードおよびHex 変換用コマンド・ファイルの変更点

C671x DSPはプログラマブルPLL(Phase Locked Loop)となっているため、ROMブートを選択した場合には、ブート・コードにPLLを設定するコードをEMIF設定の前に追加する必要があります。それ以外の基本的な流れはC641x/DM64xデバイスと変わりありません。

さらに、C671x DSPはEMIF空間を1つしか持っていないため、ブートコード内でセクションコピーを行なうために用いているアドレスやHex変換する際にROM上に配置するためのアドレスが 0x9000 0000 の空間のみであることに注意してください。

5 Object File Display (OFD)を用いたDSP イメージ作成方法の手引き

第3章のROMへの書き込み例は、CCSのプラグインツールであるFlashBurnを用い、DSPが直接フラッシュメモリーにhexファイルのデータを書き込みDSP ROMブート・システムを作成する方法になります。しかし、組込みシステムが必ずしもDSPのみで構築されているわけではなく、他のホスト・プロセッサが存在する場合もあります。この章では、TIが提供するObject File Display(OFD)ユーティリティを用いたホスト・プロセッサのアプリケーションに含まれるDSP(ブート)イメージの作成方法の概要を紹介します。

HPI/PCIブートにおけるDSPブート・イメージの作成方法が良い例になります。DSPコードは適切なフォーマットに変換されたDSPイメージとしてホスト・アプリケーションに含まれ、必要に応じてDSPのメモリー空間にコピーする必要性があります。このDSPイメージは一般的にCソースファイル(*.c/*.h)で提供されます。

このDSPイメージを作成するためには、1) COFFファイルの生成、2) COFFを特定フォーマット・イメージに変換、3) 特定フォーマット・イメージをCソース・ファイル(コードやヘッダ)に変換、する方法があります。

1) はTI リンカで生成することが可能で、2) でOFDユーティリティを使用します。OFDユーティリティはCOFFファイルを入力とし、その中に含まれているセクションの数やその種類、アドレス、サイズ等を特定の情報だけを抽出することもでき、それらをXMLファイルとして出力します。3) では、OFDユーティリティが出力したXMLファイルから情報を抽出します。一般的にPerlやVisual Basic, C++, JavaなどのXMLパーサモジュールが使用できます。

例として、OFDユーティリティを使ったDSPブート・イメージを作成するアプリケーション・ノートがサンプルコードを含めて提供されています。詳細は「Using OFD Utility to Create a DSP Boot Image (SPRAA64)」をご覧ください。

参照文献

1. *TMS320C6000 Assembly Language Tools User's Guide (SPRU186)*
2. *TMS320C6000 Boot Mode and Emulation Reset (SPRA978)*
3. *Creating a Second-Level Bootloader for FLASH Bootloading on TMS320C6000 Platform With Code Composer Studio* (SPRA999)
4. TMS320C6000 Tools: Vector Table and Boot ROM Creation (SPRA544)

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および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TI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TIの標準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TI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、且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、且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。

なお、日本テキサス・インスツルメンツ株式会社半導体集積回路製品販売用標準契約約款もご覧下さい。
http://www.tij.co.jp/jsc/docs/stdterms.htm

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

2001.11