参 考 資 料

JAJA033 WAS
SPRAA72– 11月 2004年

TEXAS INSTRUMENTS
www.tij.co.jp

Audio Developer's Kit (ADK)を用いたMcASPの使用方法とDSP/BIOS IOM Driverの実装

恒川 啓人　　DSP製品部アプリケーショングループ

アブストラクト

このドキュメントではAudio Developer's Kit (ADK)を例にMulti Channel Audio Serial Port (McASP) を設定し使用する方法について説明します。また、最終的にDSP/BIOSのIOMドライバとして実装し、このドライバを基にした簡単なサンプル・アプリケーションについて説明します。

目次

図

表

1 Audio Developer's Kit (ADK) 概要

ADKにはアナログ入出力として、入力にステレオ4ch(TI PCM1804×2, Full Differential Analog Input 24-BIT 192KHz Stereo ADC)、出力にステレオ8ch(TI DSD1608, 24-Bit 8ch Multi-Format DAC)を備えています。これらの信号はMcASP側ではI2Sフォーマットで転送されます。また、それとは別にデジタルオーディオ入出力として、S/PDIFを入力、出力それぞれ1ポートずつもっています。これらのADC、DACはそれぞれC6713のMcASPのAXR端子に接続されています。各入出力とMcASPのAXR端子の対応は表 1のようになっています。また、LRクロックはMcASPのAFSR及びAFSXに接続されおり、受信側は各コーデックから供給されています。送信側はMcASPよりクロックを供給しており、DACはMcBSP SPIモードを通じて制御することができます。

表 1 入出力端子とMcASP AXRピンの対応表

McASP0	入出力	McASP1	入出力
AXR0	S/PDIF Output	AXR0	Input0/1 (PCM1804)
AXR1	拡張コネクタ	AXR1	Input2/3 (PCM1804)
AXR2	拡張コネクタ	AXR2	S/PDIF Input
AXR3	拡張コネクタ	AXR3	Status
AXR4	拡張コネクタ	AXR4	Output0/1(DSD1608)
AXR5	未使用	AXR5	Output2/3(DSD1608)
AXR6	未使用	AXR6	Output4/5(DSD1608)
AXR7	未使用	AXR7	Output6/7(DSD1608)

2 McASP の設定

この章ではMcASP1の各レジスタの設定について説明します。デジタル入出力を使う設定に関してはこのドキュメントでは割愛します。よって2,3章で述べる設定はすべてMcASP1を用いた設定になります。

2.1 McASP グローバル設定

ここでは、McASPのピンの機能と入出力状態を設定します。また、McASPをI2Sインタフェースで使用するかDIT(Digital Audio Interface Transmitter)モードで使用するかも選択することができます。さらに、ミュートに関する設定、及びデバッグ時に最適なデジタルループバックモードの設定もここで行うことができますが、本ドキュメントではこれらの機能は使用しません。

2.1.1 Pin Function Register (PFUNC)

このレジスタではMcASPの各ピンの機能をMcASPかGPIOか選択します。各ピンの設定は下記のようになります。本ドキュメントでは基本的にすべてのビットクロック、LRクロック等の制御信号と、AXR3,2を除いたシリアル信号及びを使用します。ただし、AMUTEは使用していないのでデフォルトの設定になります。

```
/* PFUNC */

MCASP_FMKS(PFUNC, AFSR, MCASP)          |
MCASP_FMKS(PFUNC, AHCLKR, MCASP)        |
MCASP_FMKS(PFUNC, ACLKR, MCASP)         |
MCASP_FMKS(PFUNC, AFSX, MCASP)          |
MCASP_FMKS(PFUNC, AHCLKX, MCASP)        |
MCASP_FMKS(PFUNC, ACLKX, MCASP)         |
MCASP_FMKS(PFUNC, AMUTE, GPIO)          |
MCASP_FMKS(PFUNC, AXR7, MCASP)          |
MCASP_FMKS(PFUNC, AXR6, MCASP)          |
MCASP_FMKS(PFUNC, AXR5, MCASP)          |
MCASP_FMKS(PFUNC, AXR4, MCASP)          |
MCASP_FMKS(PFUNC, AXR3, MCASP)          |
MCASP_FMKS(PFUNC, AXR2, MCASP)          |
MCASP_FMKS(PFUNC, AXR1, MCASP)          |
MCASP_FMKS(PFUNC, AXR0, MCASP),
```

2.1.2 Pin Direction Register (PDIR)

このレジスタでは、各ピンを入出力どちらで使用するかを選択します。各ピンの設定は下記のようになります。ADKは受信側はビットクロック、LRクロックともにPCM1804から供給されます。また、出力側はMcASPからDSD1608に供給する必要があります。AMUTE及びAXR3,2は使用していないので、デフォルトのセッティングになります。下記に出力ステレオ8ch、入力4chの例を示します。

```
/* PDIR */

MCASP_FMKS(PDIR, AFSR, IN)              |
MCASP_FMKS(PDIR, AHCLKR, OUT)           |
MCASP_FMKS(PDIR, ACLKR, IN)             |
MCASP_FMKS(PDIR, AFSX, OUT)             |
MCASP_FMKS(PDIR, AHCLKX, IN)            |
MCASP_FMKS(PDIR, ACLKX, OUT)            |
MCASP_FMKS(PDIR, AMUTE, OUT)            |
MCASP_FMKS(PDIR, AXR7, DEFAULT)         |
MCASP_FMKS(PDIR, AXR6, DEFAULT)         |
MCASP_FMKS(PDIR, AXR5, OUT)             |
MCASP_FMKS(PDIR, AXR4, OUT)             |
MCASP_FMKS(PDIR, AXR3, DEFAULT)         |
MCASP_FMKS(PDIR, AXR2, DEFAULT)         |
MCASP_FMKS(PDIR, AXR1, IN)              |
MCASP_FMKS(PDIR, AXR0, IN),
```

2.1.3 DIT Mode Control Register (DITCTL)

このレジスタではMcASPをDITで使うかTDMで使うかを選択することができます。I2Sで使う場合はTDMになりますので、必ずDITENフィールドをTDMにセットします。そのほかのフィールドは設定に影響がないのでデフォルトにします。

```
/* DITCTL */

MCASP_FMKS(DITCTL, VB, DEFAULT)         |
MCASP_FMKS(DITCTL, VA, DEFAULT)         |
MCASP_FMKS(DITCTL, DITEN, TDM),
```

2.1.4 Digital Loopback Control Register (DLBCTL)

このレジスタではデジタルループバックモードの設定を行います。ADKを使う場合は必ずDLBENをディセーブルにしてください。

```
/* DLBCTL */

MCASP_FMKS(DLBCTL, MODE, DEFAULT)       |
MCASP_FMKS(DLBCTL, ORD, DEFAULT)        |
MCASP_FMKS(DLBCTL, DLBEN, DISABLE),
```

2.1.5 Audio Mute Control Register (AMUTE)

このレジスタではAMUTEピンの動作について設定することができますが、本ドキュメントでは使用しないので、MUTENフィールドをディセーブルにする他はすべてデフォルトの設定にしておきます。

```
/* AMUTE */

MCASP_FMKS(AMUTE, XDMAERR, DISABLE)     |
MCASP_FMKS(AMUTE, RDMAERR, DISABLE)     |
MCASP_FMKS(AMUTE, XCKFAIL, DISABLE)     |
MCASP_FMKS(AMUTE, RCKFAIL, DISABLE)     |
MCASP_FMKS(AMUTE, XSYNCERR, DISABLE)    |
MCASP_FMKS(AMUTE, RSYNCERR, DISABLE)    |
MCASP_FMKS(AMUTE, XUNDRN, DISABLE)      |
MCASP_FMKS(AMUTE, ROVRN, DISABLE)       |
MCASP_FMKS(AMUTE, INEN, DISABLE)        |
MCASP_FMKS(AMUTE, INPOL, ACTLOW)        |
MCASP_FMKS(AMUTE, MUTEN, DISABLE)
```

2.2 McASP 受信コンフィギュレーション

ここではMcASPの受信に関する各パラメータを設定します。また、エラーが発生した場合の対処方法についてもここで設定することができます。

2.2.1 Receive Format Unit Bit Mask Register (RMASK)

このレジスタでは受信データのどの部分をマスクしてパッドで埋めるかを選ぶことができます。ADKの場合は、32bitデータすべて受信する必要があるので、設定は0xFFFFFFFFになります。

```
/* RMASK */

MCASP_RMASK_OF(0xFFFFFFFF),
```

2.2.2 Receive Bit Stream Format Register (RFMT)

このレジスタでは受信データのフォーマットを指定することができます。受信データはI2Sフォーマットで送られてくるので、RDATDLYは1、MSBFIRST、RPAD=0、RSSZ=32bitになります。また、実用的にはMcASPのデータの送受信はDMAによって行われるので、DSPのバスはDATを選択します。ADKでのレジスタ設定は下記のようになります。

```
/* RFMT */

MCASP_FMKS(RFMT, RDATDLY, 1BIT)         |
MCASP_FMKS(RFMT, RRVRS, MSBFIRST)       |
MCASP_FMKS(RFMT, RPAD, ZERO)            |
MCASP_FMKS(RFMT, RPBIT, DEFAULT)        |
MCASP_FMKS(RFMT, RSSZ, 32BITS)          |
MCASP_FMKS(RFMT, RBUSEL, DAT)           |
MCASP_FMKS(RFMT, RROT, DEFAULT),
```

2.2.3 Receive Frame Sync Control Register (AFSRCTL)

このレジスタでは受信フレーム同期信号の設定を行います。この信号はLRクロックとして使用されます。I2Sで動作するため、RMODは2、FRWIDはWORD単位で出力します。このレジスタの設定は下記のようになります。

```
/* AFSRCTL */

MCASP_FMKS(AFSRCTL, RMOD, OF(2))          |
MCASP_FMKS(AFSRCTL, FRWID, WORD)          |
MCASP_FMKS(AFSRCTL, FSRM, EXTERNAL)       |
MCASP_FMKS(AFSRCTL, FSRP, ACTIVELOW),
```

2.2.4 Receive Clock Control Register (ACLKRCTL)

このレジスタでは受信ビットクロックの設定を行います。

```
/* ACLKRCTL */

MCASP_FMKS(ACLKRCTL, CLKRP, RISING)       |
MCASP_FMKS(ACLKRCTL, CLKRM, EXTERNAL)     |
MCASP_FMKS(ACLKRCTL, CLKRDIV, OF(0)),
```

2.2.5 Receive High-Frequency Clock Control Register (AHCLKRCTL)

このレジスタではHigh-Frequency クロックの設定を行います。ADKで使用する場合は、ACLKR、AFSRともに外部から入力されるため、これらのクロックソースとして用いられることはありません。よって、下記のように入力源はDSP内部になり、他の設定はデフォルトのままで問題ありません。

```
/* AHCLKRCTL */

MCASP_FMKS(AHCLKRCTL, HCLKRM, INTERNAL)   |
MCASP_FMKS(AHCLKRCTL, HCLKRP, DEFAULT)    |
MCASP_FMKS(AHCLKRCTL, HCLKRDIV, DEFAULT),
```

2.2.6 Receive TDM Time Slot Register (RTDM)

このレジスタでは、タイムスロットのアクティブ/インアクティブを設定します。I2Sで使用するので、TDM0及びTDM1を有効にします。

```
/* RTDM */

MCASP_RTDM_OF(0x00000000)                 |
MCASP_FMKS(RTDM, RTDMS0, ACTIVE)          |
MCASP_FMKS(RTDM, RTDMS1, ACTIVE),
```

2.2.7 Receiver Interrupt Control Register (RINTCTL)

このレジスタではMcASPの受信時の各ステータスによって割り込みを発生させるかどうかの設定を行うことができます。本ドキュメントで扱うサンプルコードではこれらの機能を使用しておりませんが、ユーザーが必要に応じて設定することにより、エラーハンドリングに役立てることができます。

```
/* RINTCTL */

MCASP_FMKS(RINTCTL, RSTAFRM, DISABLE)     |
MCASP_FMKS(RINTCTL, RDATA, DISABLE)       |
MCASP_FMKS(RINTCTL, RLAST, DISABLE)       |
MCASP_FMKS(RINTCTL, RDMAERR, DISABLE)     |
MCASP_FMKS(RINTCTL, RCKFAIL, DISABLE)     |
MCASP_FMKS(RINTCTL, RSYNCERR, DISABLE)    |
MCASP_FMKS(RINTCTL, ROVRN, DISABLE),
```

2.2.8 Receive Clock Check Control Register (RCLKCHK)

このレジスタでは受信クロックのチェック機能を使用する際のパラメータを指定します。本ドキュメントで扱うサンプルコードではこれらの機能を使用しておりませんが、ユーザー必要に応じて設定することにより、エラーハンドリングに役立てることができます。

```
/* RCLKCHK */

MCASP_FMKS(RCLKCHK, RCNT, DEFAULT)        |
MCASP_FMKS(RCLKCHK, RMAX, DEFAULT)        |
MCASP_FMKS(RCLKCHK, RMIN, DEFAULT)        |
MCASP_FMKS(RCLKCHK, RPS, DEFAULT)
```

2.3 McASP 送信コンフィギュレーション

ここではMcASPの送信に関する各パラメータを設定します。受信コンフィギュレーションと同様に、エラーが発生した場合の対処方法についてもここで設定することができます。

2.3.1 Transmit Format Unit Bit Mask Register (XMASK)

このレジスタでは送信データのどの部分をマスクしてパッドで埋めるかを選ぶことができます。ADKの場合は、32bitデータすべて送信する必要があるので、設定は0xFFFFFFFFになります。

```
/* XMASK */

MCASP_XMASK_OF(0xFFFFFFFF),
```

2.3.2 Transmit Bit Stream Format Register (XFMT)

このレジスタでは送信データのフォーマットを指定することができます。送信はI2Sフォーマットで行うので、XDATDLYは1、MSBFIRST、XPAD=0、XSSZ=32bitになります。また、実用的にはMcASPのデータの送受信はDMAによって行われるので、DSPのバスはDATを選択します。ADKでのレジスタ設定は下記のようになります。

```
/* XFMT */

MCASP_FMKS(XFMT, XDATDLY, 1BIT)          |
MCASP_FMKS(XFMT, XRVRS, MSBFIRST)        |
MCASP_FMKS(XFMT, XPAD, ZERO)             |
MCASP_FMKS(XFMT, XPBIT, DEFAULT)         |
MCASP_FMKS(XFMT, XSSZ, 32BITS)           |
MCASP_FMKS(XFMT, XBUSEL, DAT)            |
MCASP_FMKS(XFMT, XROT, DEFAULT),
```

2.3.3 Transmit Frame Sync Control Register (AFSXCTL)

このレジスタでは送信フレーム同期信号の設定を行います。この信号はLRクロックとして使用されます。I2Sで動作するため、XMODは2、FRWIDはWORD単位で出力します。受信側と違い、送信側はMcASPからDAC側に出力しますので、FSXMはINTERNALになります。

```
/* AFSXCTL */

MCASP_FMKS(AFSXCTL, XMOD, OF(2))         |
MCASP_FMKS(AFSXCTL, FXWID, WORD)         |
MCASP_FMKS(AFSXCTL, FSXM, INTERNAL)      |
MCASP_FMKS(AFSXCTL, FSXP, ACTIVELOW),
```

2.3.4 Transmit Clock Control Register (ACLKXCTL)

このレジスタでは送信ビットクロックの設定を行います。LRクロックと同様に、McASPからDACに供給する必要があるので、CLKXMはINTERNALになります。また、受信と送信はクロックを独立にする必要があるので、ASYNCフィールドはASYNCに設定する必要があります。また、DACのFsの設定もここで行います。以下の例ではAHCLKXが24.576MHzだった場合、Fsは48kHzになります。(24.576MHz/8/32bit/2=48kHz)

```
/* ACLKXCTL */

MCASP_FMKS(ACLKXCTL, CLKXP, FALLING)     |
MCASP_FMKS(ACLKXCTL, ASYNC, ASYNC)       |
MCASP_FMKS(ACLKXCTL, CLKXM, INTERNAL)    |
MCASP_FMKS(ACLKXCTL, CLKXDIV, OF(7)),
```

2.3.5 Transmit High-Frequency Clock Control Register (AHCLKXCTL)

このレジスタではHigh-Frequency クロックの設定を行います。ADKで使用する場合は、このクロックが送信側のすべてのクロックの基準になります。送信側は、ACKX,AFSXともに内部から供給する必要がありますが、AHCLKXの源発はADK上のオシレータから24.576MHzで固定になっています。送信側で使用するFsに合わせて適宜HCLKXDIVを変更する必要があります。以下の例では、AHCLKXは24.576MHzになります。

```
/* AHCLKXCTL */

MCASP_FMKS(AHCLKXCTL, HCLKXM, EXTERNAL)|
MCASP_FMKS(AHCLKXCTL, HCLKXP, FALLING)   |
MCASP_FMKS(AHCLKXCTL, HCLKXDIV, OF(0)),
```

2.3.6 Transmit TDM Time Slot Register (XTDM)

このレジスタでは、タイムスロットのアクティブ/インアクティブを設定します。I2Sで使用するので、TDM0及びTDM1を有効にします。

```
/* XTDM */

MCASP_XTDM_OF(0x00000000)                |
MCASP_FMKS(XTDM, XTDMS0, ACTIVE)         |
MCASP_FMKS(XTDM, XTDMS1, ACTIVE),
```

2.3.7 Transmit Interrupt Control Register (XINTCTL)

このレジスタではMcASPの送信時の各ステータスによって割り込みを発生させるかどうかの設定を行うことができます。本ドキュメントで扱うサンプルコードではこれらの機能を使用しておりませんが、ユーザーが必要に応じて設定することにより、エラーハンドリングに役立てることができます。

```
/* XINTCTL */

MCASP_FMKS(XINTCTL, XSTAFRM, DISABLE)    |
MCASP_FMKS(XINTCTL, XDATA, DISABLE)      |
MCASP_FMKS(XINTCTL, XLAST, DISABLE)      |
MCASP_FMKS(XINTCTL, XDMAERR, DISABLE)    |
MCASP_FMKS(XINTCTL, XCKFAIL, DISABLE)    |
MCASP_FMKS(XINTCTL, XSYNCERR, DISABLE)   |
MCASP_FMKS(XINTCTL, XUNDRN, DISABLE),
```

2.3.8 Transmit Clock Check Control Register (XCLKCHK)

このレジスタでは送信クロックのチェック機能を使用する際のパラメータを指定します。本ドキュメントで扱うサンプルコードではこれらの機能を使用しておりませんが、ユーザーが必要に応じて設定することにより、エラーハンドリングに役立てることができます。

```
/* XCLKCHK */

MCASP_FMKS(XCLKCHK, XCNT, DEFAULT)       |
MCASP_FMKS(XCLKCHK, XMAX, DEFAULT)       |
MCASP_FMKS(XCLKCHK, XMIN, DEFAULT)       |
MCASP_FMKS(XCLKCHK, XCKFAILSW, DISABLE)  |
MCASP_FMKS(XCLKCHK, XPS, DEFAULT)
```

2.4 McASP シリアライザ・コンフィギュレーション

ここではMcASPのシリアライザの設定を行います。各シリアライザを入出力どちらで使うのかの選択が行えます。また、インアクティブなTDMスロットを送受信するときや使用していないシリアライザのピンのステータスを設定することができます。以下に、4ch入力4ch出力の例を示します。この例では使用していないAXR[n]ピンはTri-stateになります。

```
/* srctl0 */

MCASP_FMKS(SRCTL, DISMOD, DEFAULT)       |
MCASP_FMKS(SRCTL, SRMOD, RCV),

/* srctl1 */

MCASP_FMKS(SRCTL, DISMOD, DEFAULT)       |
MCASP_FMKS(SRCTL, SRMOD, RCV),

/* srctl2 */

MCASP_FMKS(SRCTL, DISMOD, DEFAULT)       |
MCASP_FMKS(SRCTL, SRMOD, INACTIVE),

/* srctl3 */

MCASP_FMKS(SRCTL, DISMOD, DEFAULT)       |
MCASP_FMKS(SRCTL, SRMOD, INACTIVE),

/* srctl4 */

MCASP_FMKS(SRCTL, DISMOD, DEFAULT)       |
MCASP_FMKS(SRCTL, SRMOD, XMT),

/* srctl5 */

MCASP_FMKS(SRCTL, DISMOD, DEFAULT)       |
MCASP_FMKS(SRCTL, SRMOD, XMT),

/* srctl6 */

MCASP_FMKS(SRCTL, DISMOD, DEFAULT)       |
MCASP_FMKS(SRCTL, SRMOD, INACTIVE),

/* srctl7 */

MCASP_FMKS(SRCTL, DISMOD, DEFAULT)       |
MCASP_FMKS(SRCTL, SRMOD, INACTIVE),
```

2.5 McASP を設定する際の注意点

2.5.1 各制御信号のアクティベーションについて

McASPを使用する場合、おおまかに以下の順序で設定を行う必要があります。

1. McASP初期化(CSLの場合MCASP_open)
2. McASPを本ドキュメントの2.3及び2.4に倣って設定
   MCASP_configRcv,
   MCASP_configXmt
   MCASP_configSrctl
   MCASP_configGbl
3. 各制御信号及びステートマシーン、シリアライザをアクティベートする

   MCASP_enableHclk
   MCASP_enableClk
   MCASP_enableSers
   MCASP_enableSm
   MCASP_enableFsync

ここで、3.を行う時、必ずMcASPグローバル・コントロール・レジスタ及びXSTAT、RSTATを参照して確実にアクティベートしているかどうか確認しつつ次のアクティベーションに進んでください。コード例を下記に示します。

```
/* Start the respective serial clocks */
MCASP_enableHclk(hMcasp, MCASP_XMTRCV);

while(!MCASP_FGET(GBLCTL1,RHCLKRST)
   || !MCASP_FGET(GBLCTL1, XHCLKRST));

/* Start the respective serial clocks */
MCASP_enableClk(hMcasp, MCASP_XMTRCV);
while(!MCASP_FGET(GBLCTL1,RCLKRST)
   || !MCASP_FGET(GBLCTL1, XCLKRST));

/* Setup edma/core read/write */
some code to setup transfer/receiver

/* Activate serializers */
MCASP_RSET(XSTAT1,0x000001FF);
MCASP_RSET(RSTAT1, 0xFFFF);
while(MCASP_FGET(XSTAT1, XERR))
   MCASP_RSET(XSTAT1, 0x000001FF);
MCASP_enableSers(hMcasp, MCASP_XMTRCV);
while(!MCASP_FGET(GBLCTL1,RSRCLR)
   || !MCASP_FGET(GBLCTL1, XSRCLR));
/* Verify that all transmit buffers are serviced.*/
while(MCASP_FGET(XSTAT1, XDATA));

/* Release state machines from reset */
MCASP_enableSm(hMcasp, MCASP_XMTRCV);
while(!MCASP_FGET(GBLCTL1,RSMRST)
   || !MCASP_FGET(GBLCTL1, XSMRST));

/* Release flame sync generator */
MCASP_enableFsync(hMcasp, MCASP_XMTRCV);
while(!MCASP_FGET(GBLCTL1,RFRST)
   || !MCASP_FGET(GBLCTL1, XFRST));
```

2.5.2 送信タイミングについて

McASPをトランスミッタとして使用する場合、シリアライザをアクティベートする際に送信データをXRBUF[n]に用意する必要があります。タイミング的には上記ソースコード例の位置になります。これを行わないとアンダーランエラーが発生し、McASP自体の送信がとまってしまいますので、ご注意ください。

2.5.3 デバッグについて

デバッガにより、DSPをHaltした場合、McASPへの送信データの受け渡し、及び受信データの引渡しが行われなくなるので、アンダーランエラー及びオーバーランエラーが発生することがあります。これにより、デバッガで再度Runをさせても、McASPがエラー状態から復帰しないため動作しません。

3 EDMA の設定

この章ではMcASPの送受信にEDMAを使う際の設定例を説明します。

3.1 EDMA Event Selector Registers

C6713にはEDMAのチャネルイベントに対し、EDMA Event Selector Registers(ESEL) を用いてイベントを選択することができます。McASPの各送受信イベントはデフォルトで割り付けされていないので、EDMAの設定を行う前に使用するEDMAチャネルに対して使用するMcASPイベントを割り付けする必要があります。以下に、ESEL0の設定例を示します。この例ではEDMAチャネル0にMcASP1送信イベントを、チャネル1にMcASP1受信イベントを割り付けています。

```
/* ch0, AXEVT1 */

EDMA_FSET(ESEL0, EVTSEL0, 0x28);

/* ch1, AREVT1 */

EDMA_FSET(ESEL0, EVTSEL1, 0x2B);
```

3.2 EDMA Parameter RAM

ここではADKで使用する際に設定すべきと思われるパラメータRAMの各項目及びコード例について説明します。

3.2.1 EDMA Channel Options Parameter (OPT)

EDMA channel option Parameter (OPT) では、EDMAの転送タイプを指定します。McASPの送受信に使用する際は、DATパスを用いて転送を行うので、ソースまたはディスティネーションアドレスはXRBUFになります。また、ADKの場合は、エレメントサイズは32bit(うち24bitデータが有効)になります。下記にMcASPからフレーム・バッファに転送をする場合の設定例を示します。この設定の場合、EDMA転送終了コードは0になります。

```
/* OPT */

EDMA_FMKS(OPT,  PRI,  HIGH)                 |
EDMA_FMKS(OPT,  ESIZE,  32BIT)              |
EDMA_FMKS(OPT,  2DS,  NO)                   |
EDMA_FMKS(OPT, SUM, INC)                    |
EDMA_FMKS(OPT,  2DD,  NO)                   |
EDMA_FMKS(OPT,  DUM,  NONE)                 |
EDMA_FMKS(OPT,  TCINT,  YES)                |
EDMA_FMKS(OPT,  TCC,  OF(0))                |
EDMA_FMKS(OPT,  LINK,  YES)                 |
EDMA_FMKS(OPT, FS, YES),
```

3.2.2 EDMA Channel Source Address Parameter (SRC)

EDMA Channel Source Address Parameter (SRC)では、EDMA転送のソースアドレスを指定します。下記の例では、ソースアドレスにping_xというバッファを指定しています。

```
/* SRC */

EDMA_SRC_OF((Uint32)&ping_x),
```

3.2.3 EDMA Channel Transfer Count Parameter (CNT)

EDMA Channel Transfer Count Parameter (CNT)では、1Dフレーム転送の場合EDMA転送の転送エレメント数及びフレームカウントを指定します。ADKで使用する場合、ELECNTは入力または出力に使用するシリアライザの本数になります。これはシリアライザで、LR2チャネル分を送受信しているためです。以下にコード例を示します。この例の場合、フレームバッファは16フレーム、チャネルはステレオ4チャネルになります。

```
/* CNT */

EDMA_FMKS(CNT, FRMCNT, OF(0x000F)          |
EDMA_FMKS(CNT, ELECNT, OF(0x0002)),
```

3.2.4 EDMA Channel Destination Address Parameter (DST)

EDMA Channel Destination Address Parameter (DST) では、EDMA転送のディスティネーションアドレスを指定します。下記の例では、ディスティネーションアドレスにMcASP1のXBUF (DATバス経由) を指定しています。

```
 /* DST */

 EDMA_DST_OF(MCASP_ADDR(XBUF1)),
```

3.2.5 EDMA Channel Index Parameter (IDX)

EDMA Channel Index Parameter (IDX)では、1D及び2Dのアレイ同期転送でソース/ディスティネーション更新モードにインデックスを用いた転送を行ったときに使用します。この機能を使うことにより、送受信時にコアのリソースを使用することなくEDMA転送によりメモリ上へLRチャネルを振り分けながら転送を行うことが可能です。

```
/* IDX */

EDMA_FMKS(IDX, FRMIDX, DEFAULT)          |
EDMA_FMKS(IDX, ELEIDX, DEFAULT),
```

3.2.6 EDMA Channel Count Reload/Link Address Parameter (RLD)

EDMA Channel Count Reload/Link Address Parameter (RLD)では、1Dエレメント同期転送で使用する、エレメントリロードカウント及び転送リンク先のパラメータRAMを指定します。

```
/* RLD */

EDMA_FMKS(RLD, ELERLD, DEFAULT)          |
EDMA_FMKS(RLD, LINK, DEFAULT),
```

4 generic driver を用いた ADK driver の実装

4.1 デバイスドライバ・モデル概要

この章で扱うデイバスドライバはIOM mini-driverの部分になります。2レイヤ・デバイスドライバ・モデルの下位レイヤとして実装されています。上位レイヤはクラスドライバと呼ばれ、DSP/BIOS GIO、SIO/DIO、PIP/PIOモジュールのいずれかになります。クラスドライバは独立性と汎用的なAPI、様々なmini-driverの差異を吸収する機能を提供します。アプリケーションはI/Oリクエストのための汎用的なインタフェースを利用できます。図 1はDSP/BIOSデバイスドライバ・アーキテクチャの全体を示しています。GIOとSIO/DIO、PIP/PIOモジュールだけでなく、IOMデバイスドライバ・モデルに関する詳細も、*DSP/BIOS Device Driver Developer's Guide* (SPRU616) をご覧ください。

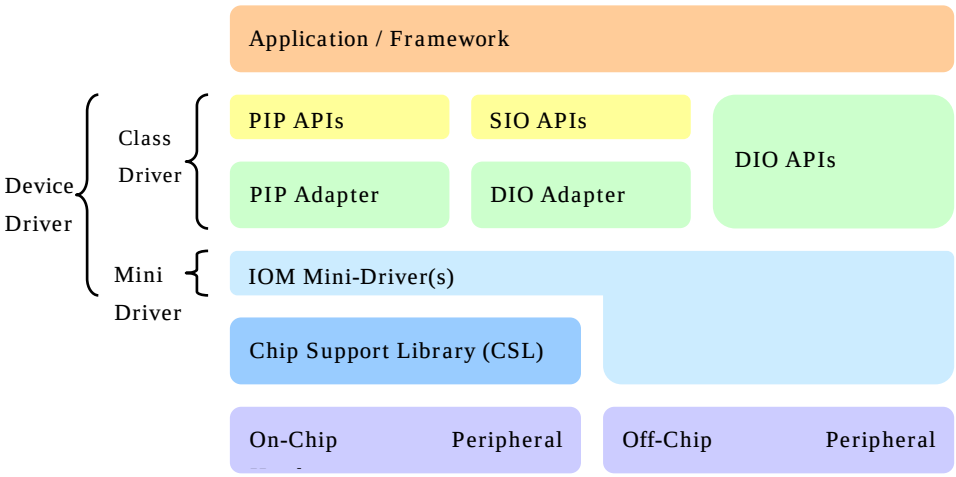

図 1 DSP/BIOS IOM Device Driver Model

多くのmini-driverはコーデックの差異に対応するためコーデック依存部分と汎用的な部分に別けて実装されています。Figure 2はmini-driverが汎用的な部分とコーデック依存部分に別けられたシステムでのコンポーネント間のデータフローを示しています。このデバイスドライバはシリアルポートからデータを送受信するために、汎用TMS320C6x1x EDMA McASPデバイスドライバを利用しています。このデバイスドライバを使うということは、アプリケーションはこのデバイスドライバ・ライブラリ(*adk6713.lib*)だけでなく、汎用デバイスドライバ・ライブラリ(*c6x1x_edma_mcasp.l67*)をリンクする必要があります。その汎用デバイスドライバの利用はユーザーから隠蔽されています。

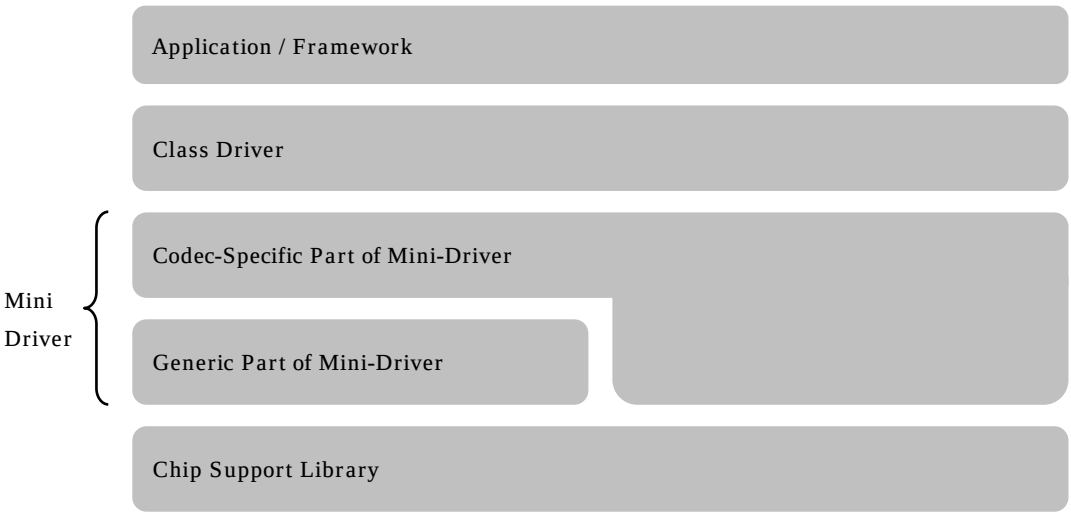

図 2 Codec Device Driver Partitioni

4.2 デバイスドライバ･サンプル

4.2.1 コンフィギュレーション

このドライバを使うためには、コンフィグレーションツールでデバイスエントリを追加し、設定する必要があります。このデバイスドライバは汎用TMS320c6x1x EDMA McASPドライバを必要に応じて設定します。

- **Init function**: _ADK6713_initを入力
- **Function table ptr**: _ADK6713_Fxnsを入力
- **Function table type**: IOM_Fxnsを選択
- **Device id**:ADK6713ではMcASP port 1だけを利用するため、このプロパティはこのデバイスドラバでは無視されます。
- **Device params ptr**: デバイス・パラメータ構造体のインスタンスのポインタです。0x0を設定すると、デフォルトパラメータになります。パラメータ構造体とそのデフォルトは後述します。
- **Device global data ptr**: このプロパティは0x0に指定してください。

4.2.2 デバイス・パラメータ

以下にデバイス･パラメータの各要素について説明します。

```
typedef struct ADK6713_DevParams {
/* Set to the version number used by the application */
   Int versioned;

/* Set to TRUE if buffers are in external memory */
   Bool cacheCalls;

/* Set VALUE for internal clock generator */
   Uns enableClkg;

/* Set VALUE for internal high frequency clock generator */
   Uns enableHclkg;

/* Set VALUE for internal framesync generator */
   Uns enableFsyncg;

/* IRQ number to use for EDMA interrupt */
   Int irqId;

/* IRQ number used for McASP Event interrupt */
   Int inEvtIrqId;

/* IRQ number used for McASP Event interrupt */
   Int outEvtIrqId;

/* Register events callback */
   C6X1X_EDMA_MCASP_EvtCallback *evtCallback;

/* Interrupt mask, set while executing input ISR */
   Uns inEvtIntrMask;

/* Interrupt mask, set while executing output ISR */
   Uns outEvtIntrMask;

/* Interrupt mask, set while executing EDMA ISR */
   Uns edmaIntrMask;

/* Set value for McASP enable channel */
   Uns enableMcaspCh;

   enum SAMPLE_RATE SampleRate;

} ADK6713_DevParams;
```

- **versionId**: このパラメータはドライバの現在の正しいバージョンを示します。
- **cacheCalls**: もしこのパラメータがTRUEならば、デイバスドライバはデバイスのIOMチャネルにつながれたバッファが、キャッシュの有効なメモリとL2データキャッシュにあるとして扱います。このパラメータのデフォルトはTRUEです。
- **enableClkg, enableHclkg, enableFsyncg**: これらのパラメータはクロックソースを選択します。パラメータの詳細は汎用 C6x1X_EDMA_MCASP driver manual (SPRA870)をご覧ください。
- **irqId, inEvtIrqId, outEvtIrqId**: これらのパラメータは割り込みが利用するIRQ番号を選択します。パラメータの詳細は汎用 C6x1X_EDMA_MCASP driver manual (SPRA870)をご覧ください。
- **evtCallback**: このパラメータはイベント・コールバック構造体のポインタです。NULLにするとイベントは登録されません。
- **edmaIntrMask, inEvtIntrMask, outEvtIntrMask**: これらはISRが呼ばれる前に割り込みを無効にするための割り込みマスクです。
- **enableMcaspCh**: このパラメータは利用するMcASPチャネルを選択します。
- **SampleRate**: このパラメータはサンプリングレートを48kHzか 96kHz、192kHzの中から選択します。

4.2.3 チャネル･パラメータ

このドライバはチャネルパラメータを実装していません。チャネルパラメータとして渡されたすべての値は無視されます（NULLを推奨します）。

4.2.1 コントロール･コマンド

このデバイスドライバは実行時コントロールコマンドを実装していません。

4.3 デバイス･ドライバ･アーキテクチャ

mini-driverのコーデック依存部分は汎用TMS320C6x1x EDMA McASPドライバの特徴を継承しています。このドライバではC6713 DSKとADK6713をセットアップするために、２つのコーデック依存の関数 (mdBindDev(), mdCreateChan())を使います。これらの関数は汎用ドライバのmdBindDev()関数とmdCreateChan()関数を呼び、初期化の汎用的な部分を完了させます。コーデック依存部分はコーデックをセットアップするだけで、データの転送は汎用デバイスドライバに任せます。このデバイスドライバは汎用デバイスドライバをユーザーから隠蔽して利用するので、アプリケーションに汎用デバイスドライバをリンクしなければなりません。このmdBindDev()関数は、渡されたADK6713_DevParams構造体に基づいてチャネルを制御し、コーデックの設定をします。mdCreateChan()関数はデータ転送で利用するEDMAの設定情報を生成します。

4.4 ドライバサンプル/サンプルアプリケーションのビルド方法

ここではサンプルデバイスドライバ、及びそれを利用したサンプルプログラムの使用方法を説明します。以下にディレクトリ構成とファイル構成を示します。

adk_app/	サンプル・アプリケーション
adk_driver/	デバイスドライバ･サンプルコード
doc/	ドキュメント
readme.txt	始めにお読みください

4.4.1 デバイスドライバのビルド方法

1. **以下のファイルを用意します。**
   - adk6713.[ch] (ADK6713 device driver)
   - adk_init.[ch] (ADK6713 initialize routine)
2. Code Composer Studio (CCS) **で以下の作業を行ってデバイスドライバをビルドするためのプロジェクトを作成します。**
   a. Project**メニューから**New Project**を選択**
   b. Project Type: Library (lib)**を選択**
   c. Target Family: TMS320C67XX**を選択**
   d. Project**メニューから**Add file to Project...**を選択**
   e. adk6713.c, adk_init.c**を追加**
3. Project**メニューから**Build Options...**を選択をし、ビルドのためのオプションを指定します。**
   a. Compiler **タブで**Preprocessor**を選択**
   b. Include Search Path (-i): C:\ti\ddk\include
   c. Define Symbols (-d): _DEBUG;CHIP_6713
4. Project**メニューから**Rebuild All**を選択。**
   ./Debug/adk6713.lib **が生成されます。**

4.4.2 SWI, SIO サンプルアプリケーションのビルド方法

このサンプルプログラムは入力4ch (stereo 2ch)を出力8ch (stereo 4ch)から出力します。

AD 0ch -> DA 4ch, DA 6ch
AD 1ch -> DA 5ch, DA 7ch

1. **以下のファイルを用意します。**
   - adk6713swi.cdb (DSP/BIOS Configuration database)
   - adk6713_devParams.c (device parameter)
   - swi_audio.c (application program)
   - adk_alignment.[ch] (translate from TDM alignment into channel alignment)
   - ../adk_driver/adk6713.h (ADK6713 device driver header file)
   - ../adk_driver/adk6713/Debug/adk6713.lib (ADK6713 device driver)
   - adk6713swi.cmd (User specified liker command file)
2. CCS**で以下の作業を行い、プロジェクトを作成します。**
   a. Project**メニューから**New Project**を選択**
   b. Project Type: Executable (.out)**を選択**
   c. Target Family: TMS320C67XX**を選択**
   d. Project**メニューから**Add file to Project...**を選択**
   e. 1.**で用意したファイルをすべてプロジェクトに登録**
3. Project**メニューから**Build Options...**を選択し、ビルドオプションを指定します。**
   a. Compiler**タブの**Preprocessor**をクリック**
   b. Include Search Path (-i): (Proj_dir)\..\adk_driver;C:\ti\ddk\include (device driver directory**と**DDK include directory**を設定**)
   c. Defile Symbols (-d): _DEBUG;CHIP_6713
   d. Linker**タブの**Basic**で、**Library Search Path: $(Proj_dir)\..\adk_driver\Debug;C:\ti\ddk\lib
   e. Output filename: $(Proj_dir)\Debug\adk6713swi.out
   f. Map filename: $(Proj_dir)\Debug\adk6713swi.map
4. **コマンドファイル**(adk6713swi.cmd) **を開き、作成したデバイスドライバ** (adk6713.lib) **、**DDK**の**generic driver (c6x1x_edma_mcasp.l67) **、 及び** C6713DSK Board Support Library (dsk6713bsl.lib) **が指定されていることを確認してください。**

```
/* include config-generated link command file */

-l ads6713swicfg.cmd

/* include libraries for the IOM driver and PIO
adapter */
/* ~/adk_driver/Release/ */
-l adk6713.lib

/* C:\ti\ddk\lib */
-l c6x1x_edma_mcasp.l67

/* C:\ti\c6000\dsk6713\lib\dsk6713bsl.lib */
-l dsk6713bsl.lib
```

5. Project**メニューから**Rebuild All**を選択**
6. File**メニューから**Load Program**を選択し、**
   .\Debug\adk6713swi.out **を**Load**します。**
7. Debug**メニューから**Run**を選択し、実行します。**

4.4.3 バッファの型とアライメント

このIOMデバイスドライバからアプリケーションへのデータ(IOM_Packet)の並びは時分割多重方式です。例えば、4ch (stereo 2ch)入力の場合、IOM_Packetには各チャネルのsigned intのデータが以下の順番で格納されます。

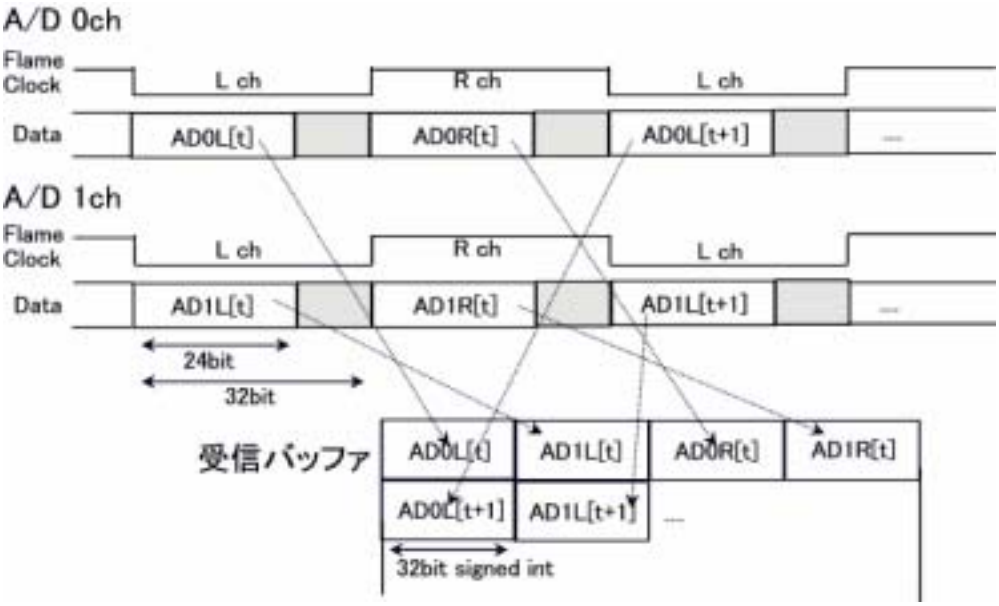

図 3　A/D変換後のデータ配置

これを浮動小数点演算可能なDSPで信号処理をしやすいように、チャネルごとに並び替え、floatに型変換する、またその逆の処理をするサブルーチンを用意しました。adk_alignment.c に、以下の二つの関数があります。

- **void *ADK_inbufalign*(int **inbuf*, float **outbuf*, int *size*, int *chNum*);**
  時分割多重方式で格納されているIOM_Packet**を**singed int**から**float**に型変換しながらチャネル毎に並べ替え、格納します。**IOM_Packet**のバッファのポインタ**inbuf**、出力先バッファのポインタ**outbuf**、バッファのサイズ**size**、多重化チャネル数**chNum**を指定します。**
- **void *ADK_outbufalign*(float **inbuf*, int **outbuf*, int *size*, int *chNum*);**
  ADK_inbufalignの逆の処理をします。チャネル毎に並べ替えられた**float**型のバッファからデバイスドライバに渡す**IOM_Packet**用の**signed int**の時分割多重方式に変換します。入力元バッファのポインタ**inbuf**、IOM_Packetのバッファのポインタ**outbuf**、バッファのサイズ**size**、多重化チャネル数**chNum**を指定します。**

参考文献

1. *A DSP/BIOS EDMA McASP Device Driver for TMS320C6x1x DSPs* (SPRA870)
2. *ADK-6713 Hardware Reference Manual* (Momentum Data System)
3. *DSP/BIOS Driver Developer's Guide* (SPRU616)
4. *TMS320C6000 Chip Support Library API Reference Guide* (SPRU401)
5. *TMS320C6000 DSP/BIOS Application Programming Interface (API) Reference Guide* (SPRU403)
6. *TMS320C6000 DSP Multichannel Audio Serial Port (McASP) Reference Guide* (SPRU041)
7. *TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Controller Reference Guide* (SPRU234)

付録 A. Device Driver Data Sheet

A.1 Device Driver Library Name

adk6713.lib
アプリケーションをビルドするとき、汎用c6x1x_edma_mcasp.l67ライブラリが必要です。

A.2 DSP/BIOS Modules Used

汎用TMS320C6x1x EDMA McASPデバイスドライバと同じです。

A.3 DSP/BIOS Objects Used

汎用TMS320C6x1x EDMA McASPデバイスドライバと同じです。

A.4 CSL Modules Used

汎用TMS320C6x1x EDMA McASPデバイスドライバと同じです。

A.5 CPU Interrupts Used

汎用TMS320C6x1x EDMA McASPデバイスドライバと同じです。

A.6 Peripherals Used

汎用TMS320C6x1x EDMA McASPデバイスドライバと同じです。

A.7 Maximum Interrupt Latency

汎用TMS320C6x1x EDMA McASPデバイスドライバと同じです。

A.8 Memory Usage

汎用TMS320C6x1x EDMA McASPデバイスドライバメモリの使用度を含んでいます
。

表 2 Device Driver Memory Usage

	Uninitialized memory	Initialized memory
CODE	-	320 words
DATA	60 words	86 words

更新履歴

版	ページ	追加/変更/削除項目
初版 Nov/04		初版リリース

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及び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