お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

M32C/80 シリーズ

内蔵周辺機能を利用したデジタルオーディオ I/F ドライバ

RJJ05B0180-0110Z
Rev.1.10
2003.07.25

1. 要約

この資料は、M32C/80 シリーズの内蔵周辺機能を利用したデジタルオーディオ I/F ドライバの作成例を掲載しています。

2. はじめに

この資料で説明している作成例は、次のマイコンに適用できます。
・マイコン　：M32C/80 シリーズ

目次

3. デジタルオーディオ I/F ドライバの概要

音楽 CD(CD-DA)などで扱われているデジタルオーディオデータは、サンプリング周波数 44.1kHz、量子化ビット数が 16 ビットの PCM(Pulse Code Modulation)データです。このデータを DAC(Digital Analog Converter)が要求するタイミングに合わせて入力することで音楽再生することが出来ます。

本作成例は、マイコンの内蔵周辺機能を利用して、デジタルオーディオデータを外付け DAC へ入力することにより音楽再生を行うデジタルオーディオ I/F ドライバの作成例について説明しています。

3.1 H/W 接続例

本作成例では、クロック同期形シリアル I/O(CSIO)を使用してデジタルオーディオデータを外付け DAC へ入力することにより音楽再生することを想定しています。
以下に、本作成例で想定しているオーディオ I/F 部分についての H/W 接続例を示します。

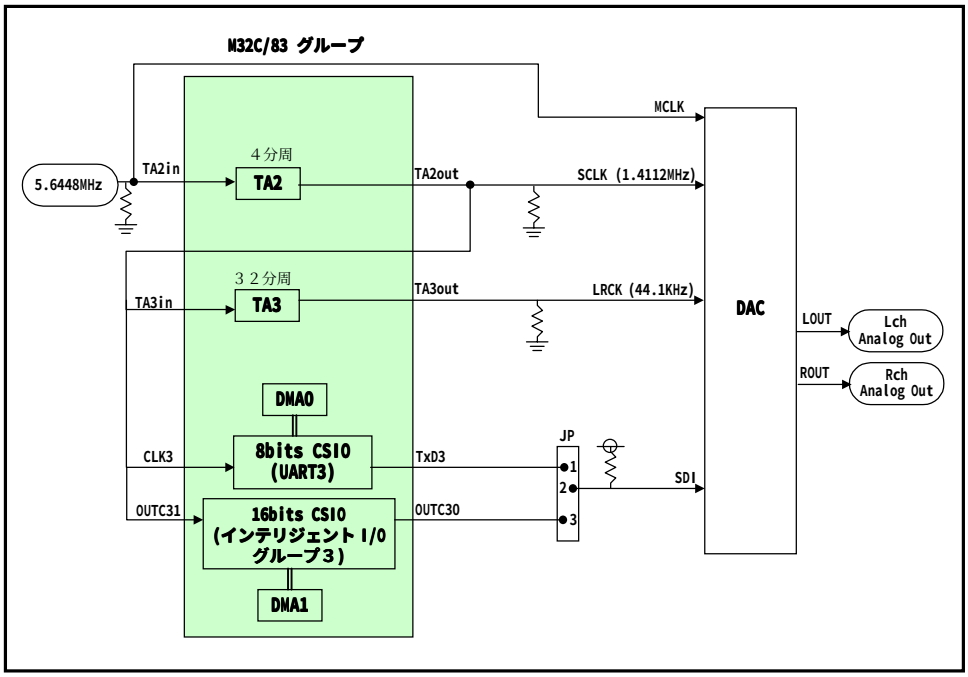


図 1.H/W 接続例

【注意】16 ビット CSIO が使用できるのは M32C/83 グループのみです。
16 ビット CSIO はインテリジェント I/O グループ 3 を使用します。

MCLK：マスタクロック(DAC 用マスタクロック)
SCLK：DAC 用シリアルクロック（シリアルデータ通信の同期用に使用）
LRCK：DAC 用 Lch,Rch クロック（Lch,Rch 同期用クロック）
SDI　：シリアルデータ（DAC に入力するオーディオデジタルデータ）

3.2 デジタルオーディオ信号

本作成例で想定しているデジタルオーディオ信号フォーマットと各信号タイミングの仕様を以下に示します。

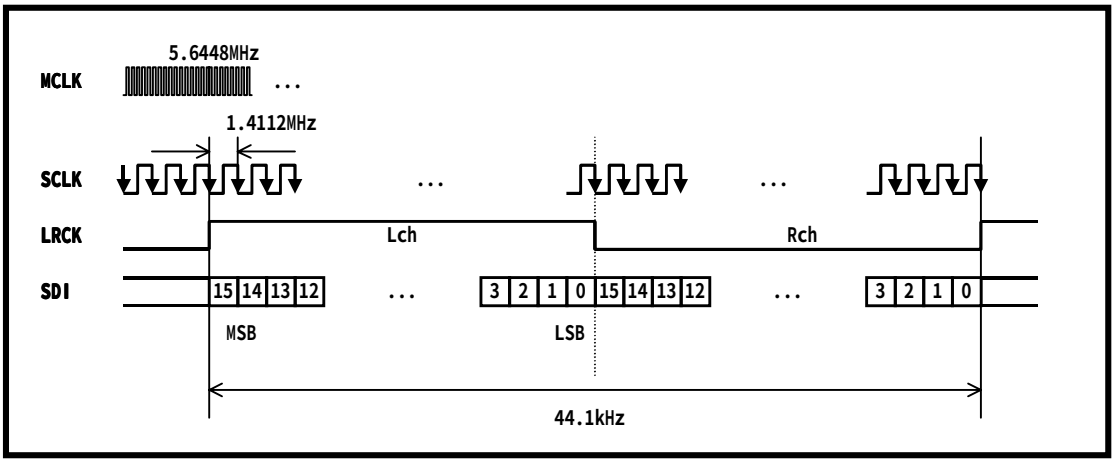


図 2.デジタルオーディオ信号フォーマット

- サンプリング周波数：44.1kHz
- 量子化ビット数：16 ビット
- ステレオ PCM データ

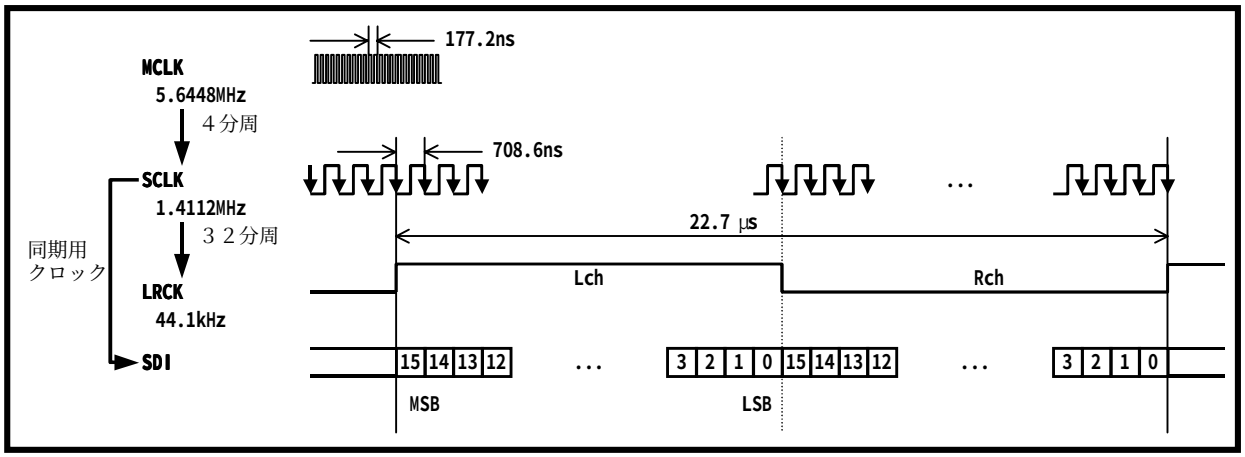


図 3.デジタルオーディオ信号タイミング

- SCLK の立下りで LRCK の極性が変わる
- SCLK の立下りでデータの同期をとる

3.3 ドライバ構成

ここでは作成例のデジタルオーディオ I/F ドライバの構成について説明します。

3.3.1 DMA 転送

作成例では、DAC へデジタルオーディオデータを送信する処理は、マイコンに内蔵されている DMAC と CSIO を組み合わせて実現しています。また、再生元のデジタルオーディオ信号は HDD などのストレージデバイスに格納されているものとします。以下に、データフローから見たシステム例を示します。

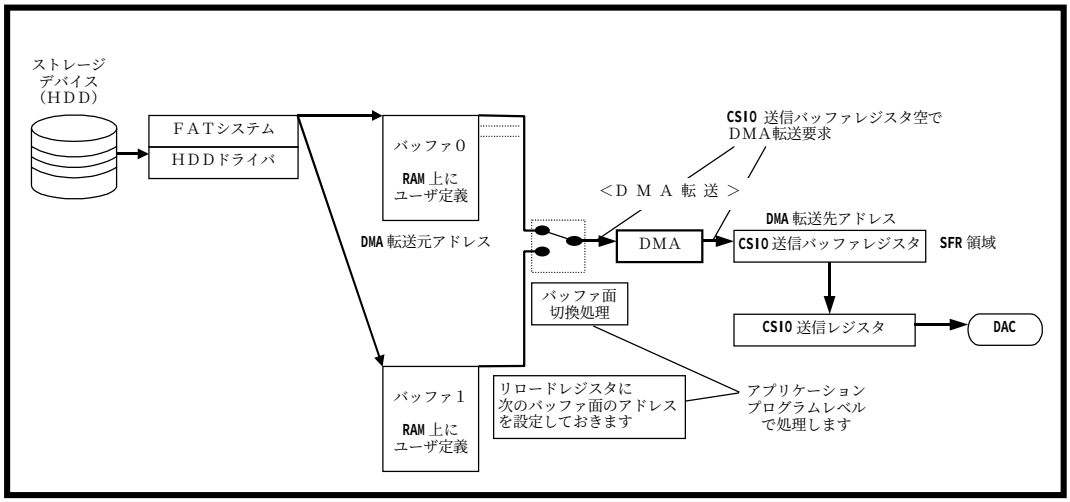


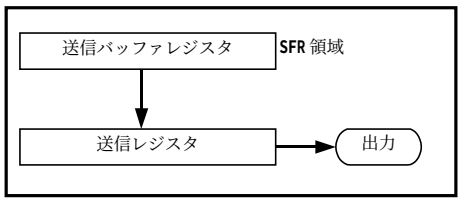
図 4.データフローシステム例

(1) 図 4 のようにバッファ(RAM)に、ストレージデバイスから読み込んだデジタルオーディオデータを格納していきます。バッファに格納したデータは DMAC により CSIO 送信バッファレジスタへ DMA 転送します。
(2) DMA 転送元、転送先および DMA 要求要因は以下の設定にします。
DMA 転送元：バッファ(RAM)
DMA 転送先：CSIO 送信バッファレジスタ
DMA 要求要因：CSIO の送信割込み要求
(3) 音楽再生の連続性を保つ為に、本作成例ではバッファを二面構成としバッファ面を切換えて使用します。片方のバッファの内容を DMA 転送しながら、他方のバッファにデータを格納していく構成にします。
(4) DMA 転送モードはリピート転送モードとし、M32C/80 シリーズの DMAC が持つ「転送カウンタ」と「アドレス」についてのリロードレジスタを使用します。
バッファに格納されたデータの出力が完了すると、DMA 割込みがかかります。DMA 割込みまでに、リロードレジスタを設定しておくことで、バッファ面の切換処理時に DMA 転送の中断を避けています。

【補足】
バッファ面数については本作成例では二面構成としていますが、二面以上のバッファ面を持たせたシステム構成も考えられます。

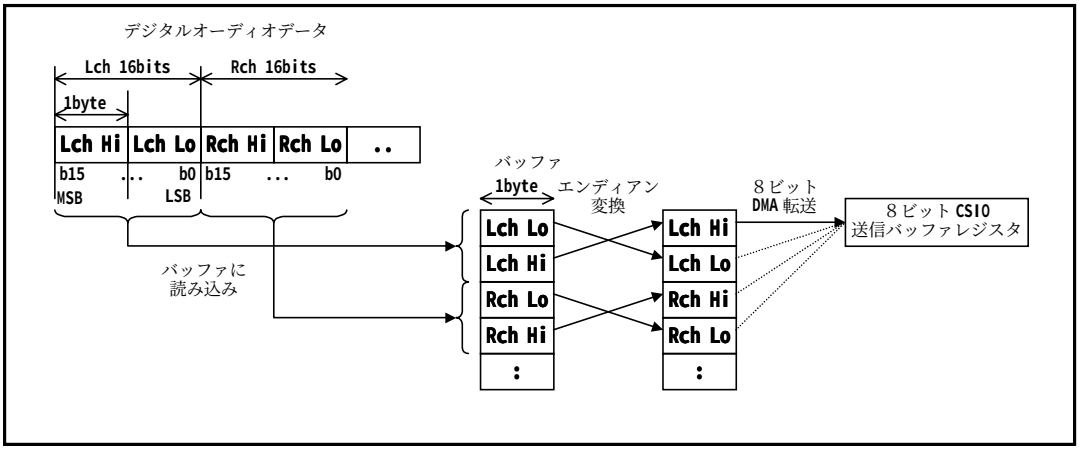
3.3.2 CSIO 割込み要因

クロック同期形シリアル I/O(CSIO)の割込み要因は「送信バッファ空」を設定します。
シリアル I/O のレジスタ構成は以下のように、送信バッファレジスタと送信レジスタの二段構成になっています。
「送信バッファレジスタ」が空になれば CSIO の割り込み要求がかかり、DMA が「送信バッファレジスタ」にデータを転送します。



3.3.3 8 ビット CSIO 使用時のエンディアン変換

M32C/80 シリーズはリトルエンディアンのマイコンです。本作成例で扱うデジタルオーディオデータは
1 チャネル当たり 16 ビットのデータで以下のような構成になっています。
その為、16 ビットのデータをバッファに読み込んで、8 ビット CSIO を使用して 8 ビット単位でデータ送信を行う場合は、エンディアンの変換処理が必要です。



4. 周辺機能設定

「図 1.H/W 接続例」に示すように、本作成例はマイコンの内蔵周辺機能を利用してデジタルオーディオ信号を再生します。
ここでは、「3.デジタルオーディオ I/F ドライバの概要」で想定したシステムについて、デジタルオーディオ I/F ドライバで使用する各周辺機能の設定方法と設定フローについて説明します。

4.1 使用する周辺機能と設定内容

本作成例で使用している周辺機能一覧を表 1 に、その設定値を表 2－表 4 に示します。

表 1.デジタルオーディオ I/F ドライバで使用する周辺機能一覧

CSIO	周辺機能	設定概要
CSIO8,CSIO16 共通	TA2	SCLK 生成、イベントカウントモード
CSIO8,CSIO16 共通	TA3	LRCK 生成、イベントカウントモード
CSIO8	UART3	８ビット転送 クロック同期形シリアル、SDI データの転送に使用します。 転送クロック：外部クロック(SCLK)
CSIO8	DMA0	８ビットデータを UART3 送信バッファレジスタに転送します。 リピート転送 転送方向：メモリ→固定アドレス(UART3 送信バッファ)
CSIO16	ｲﾝﾃﾘｼﾞｪﾝﾄ I/O グループ 3	16 ビット転送 クロック同期形シリアル、SDI データの転送に使用します。 転送クロック：外部クロック(SCLK)
CSIO16	DMA1	16 ビットデータをｲﾝﾃﾘｼﾞｪﾝﾄ I/O グループ 3 SI/O 送信バッファレジスタに転送します。 リピート転送 転送方向：メモリ→固定アドレス(ｲﾝﾃﾘｼﾞｪﾝﾄ I/O グループ 3 SI/O 送信バッファレジスタ) （ｲﾝﾃﾘｼﾞｪﾝﾄ I/O グループ 3 を使用する場合、使用できる DMA は、DMA1 か DMA3 に限定されます）

表 2.周辺機能設定値（8 ビット CSIO/16 ビット CSIO 共通）

周辺機能	用　途	項　目	設　定　値
TA2	MCLK を 4 分周して SCLK を生成します	モード	イベントカウントモード
		カウントソース	TA2IN 端子に入力された外部信号(MCLK→TA2IN 端子)の立上がりをカウントします(TA2IN 端子を入力ポートに設定)
		カウント動作	アップダウンフラグの内容（ダウンカウント） リロードタイプ
		分周比	外部信号(MCLK)を 4 分周して TA2OUT 端子にパルス出力して SCLK を生成します。 （パルス出力はアンダーフローする毎に TAiOUT 端子の極性が反転するので 2 分周を設定します）
		TAiIN 端子機能	TA2IN 端子を入力ポートに設定
		TAiOUT 端子機能	パルス出力機能 （対応する機能選択レジスタで TA2OUT 出力を選択します）
		割込みレベル	0
TA3	SCLK を 32 分周して LRCK を生成します	モード	イベントカウントモード
		カウントソース	TA3IN 端子に入力された外部信号(SCLK)の立下りをカウントします。(TA3IN 端子を入力ポートに設定)
		カウント動作	アップダウンフラグの内容（ダウンカウント） リロードタイプ
		分周比	外部信号(SCLK)を 32 分周して TA3OUT 端子にパルス出力して LRCK を生成します。 パルス出力はアンダーフローする毎に TAiOUT 端子の極性が反転するので 16 分周を設定します。但し、SCLK、LRCK 及び SDI を同期させる為に、初回のみ 17 分周を設定します。（図 5 参照）
		TAiIN 端子機能	TA3IN 端子を入力ポートに設定
		TAiOUT 端子機能	パルス出力機能 （対応する機能選択レジスタで TA3OUT 出力を選択します）
		割込みレベル	0

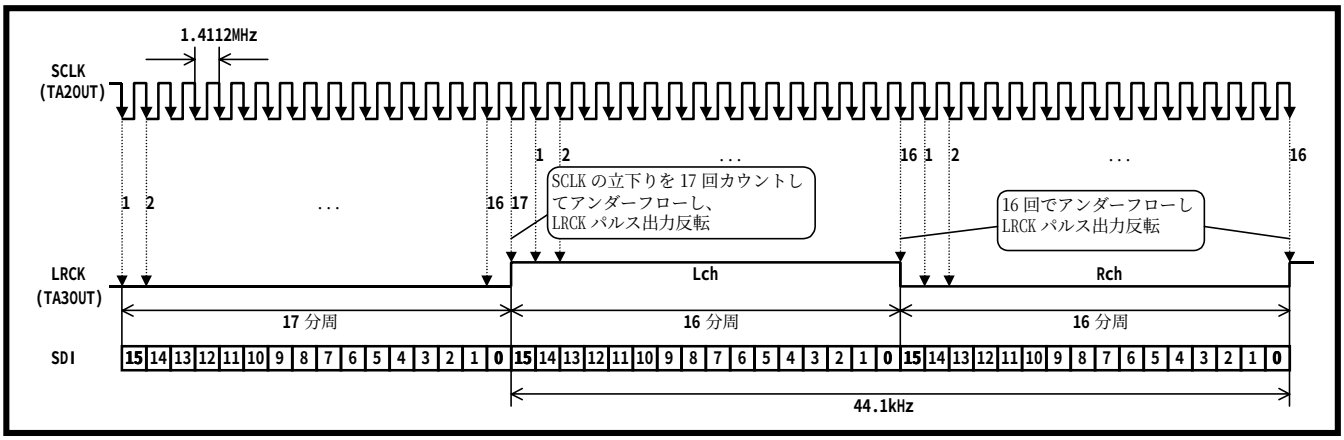


図 5.LRCK(TA3)の分周比設定

表 3.周辺機能設定値（8 ビット CSIO）

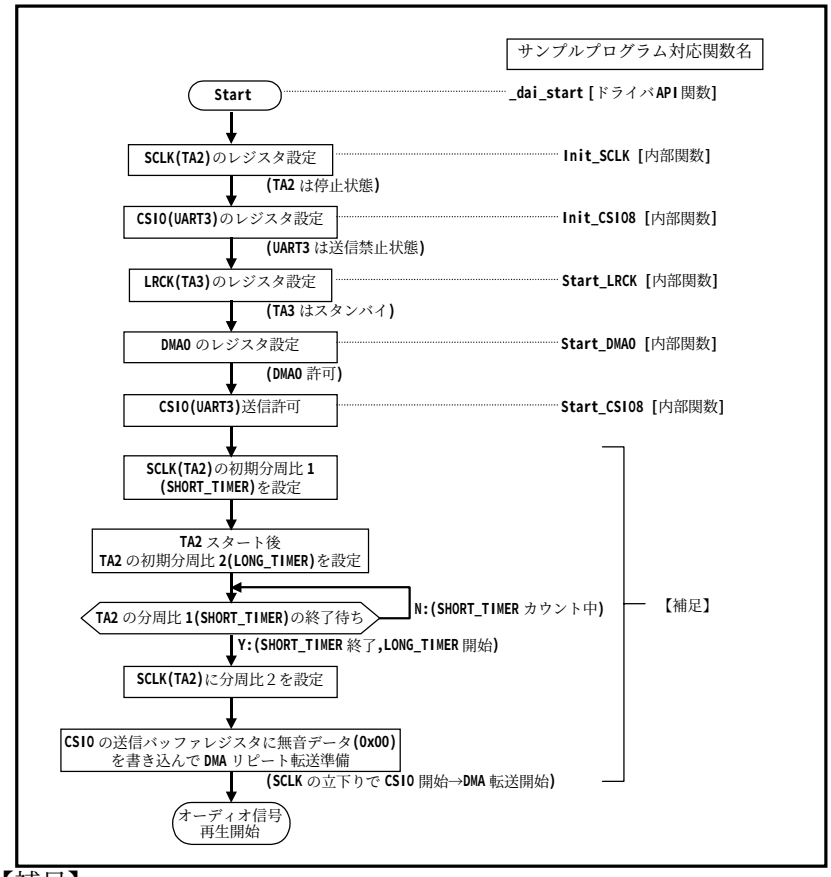
周辺機能	用　途	項　目	設　定　値
UART3	ﾃﾞｼﾞﾀﾙｵｰﾃﾞｨｵﾃﾞｰﾀをクロック同期送信します	モード	クロック同期形シリアル I/O モード
		転送データフォーマット	転送データ長　8 ビット MSB ファースト
		転送クロック	外部クロック(CLK3 端子から入力) （対応する機能選択レジスタ A で入出力ポートに設定）
		BRG 選択	f1（カウントソースを分周しない）
		CLKi 極性選択	転送クロックの立下りで送信データ出力
		TxD 端子	対応する機能選択レジスタ A,B,C で TxD 出力を設定
		割込み要因選択	送信バッファ空
		その他	CTS/RTS 機能禁止、データ論理反転なし
		割込みレベル	0
DMA0	バッファに格納されたデータを UART3 送信ﾊﾞｯﾌｧﾚｼﾞｽﾀへ転送します	モード	リピート転送
		DMA 要求要因	UART3 の送信割込み要求
		転送単位	8 ビット
		転送方向	メモリ→固定アドレス(UART3 送信バッファレジスタ)
		割込みレベル	7

表 4.周辺機能設定値（16 ビット CSIO）

周辺機能	用　途	項　目	設　定　値
ｲﾝﾃﾘｼﾞｪﾝﾄ I/O ｸﾞﾙｰﾌﾟ 3	ﾃﾞｼﾞﾀﾙｵｰﾃﾞｨｵﾃﾞｰﾀをクロック同期送信します	モード	クロック同期形シリアル I/O モード
		転送データフォーマット	転送データ長　16 ビット （この機能はｲﾝﾃﾘｼﾞｪﾝﾄ I/O ｸﾞﾙｰﾌﾟ 3(SI/O 通信ﾓｰﾄﾞﾚｼﾞｽﾀ 3)のみ） MSB ファースト
		転送クロック	外部クロック(ISCLK3 端子から入力) （対応する機能選択レジスタ A で入出力ポートに設定）
		BRG 選択	f1（カウントソースを分周しない）
		CLKi 極性選択	選択不可（転送クロックの立下りのみ）
		TxD 端子	対応する機能選択レジスタ A,B,C で II/O ｸﾞﾙｰﾌﾟ 3 出力に設定
		割込み要因選択	送信バッファ空
		その他	ｸﾞﾙｰﾌﾟ 3 波形生成制御：通信機能制御の出力をポートに割り当てます 機能許可レジスタ：チャネル 0,1 の機能を動作させます
		割込みレベル	0
DMA1	バッファに格納されたデータをｸﾞﾙｰﾌﾟ 3S/IO 送信ﾊﾞｯﾌｧﾚｼﾞｽﾀへ転送します	モード	リピート転送
		DMA 要求要因	ｲﾝﾃﾘｼﾞｪﾝﾄＩ/O割り込み制御レジスタ１０
		転送単位	16 ビット
		転送方向	メモリ→固定アドレス(ｸﾞﾙｰﾌﾟ 3 SI/O 送信ﾊﾞｯﾌｧﾚｼﾞｽﾀ)
		割込みレベル	7

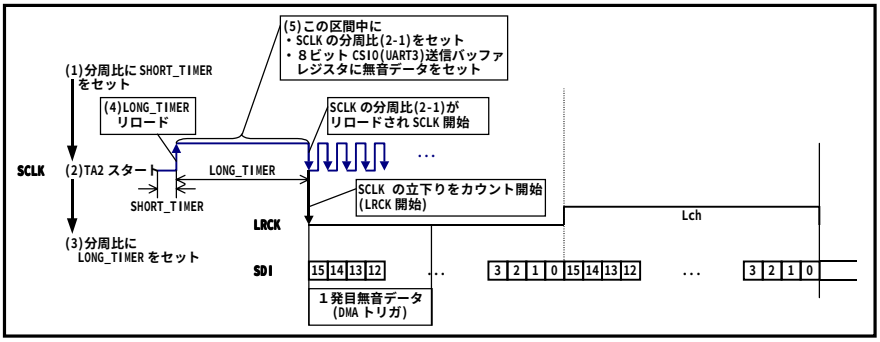
4.2 周辺機能設定フローチャート

4.2.1 デジタルオーディオ信号再生開始 設定フロー（8 ビット CSIO）

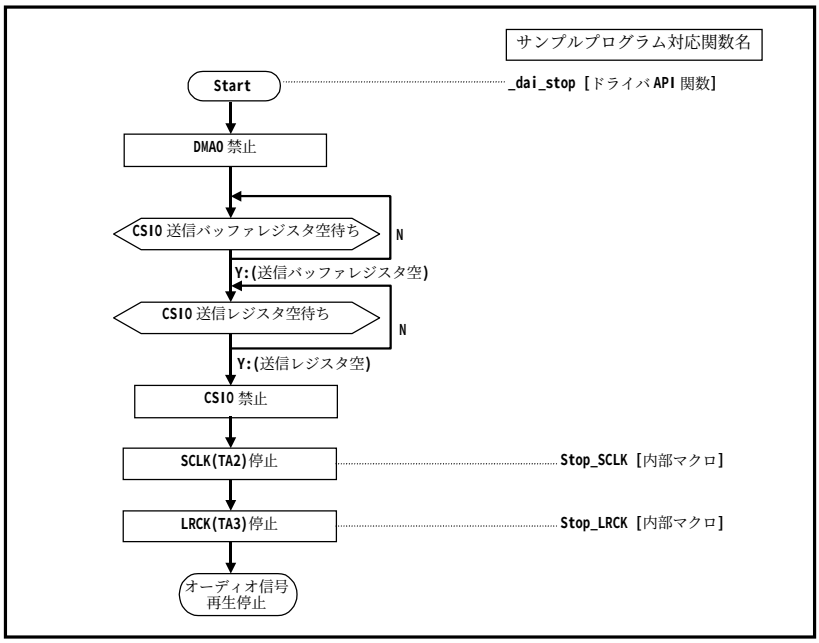


【補足】

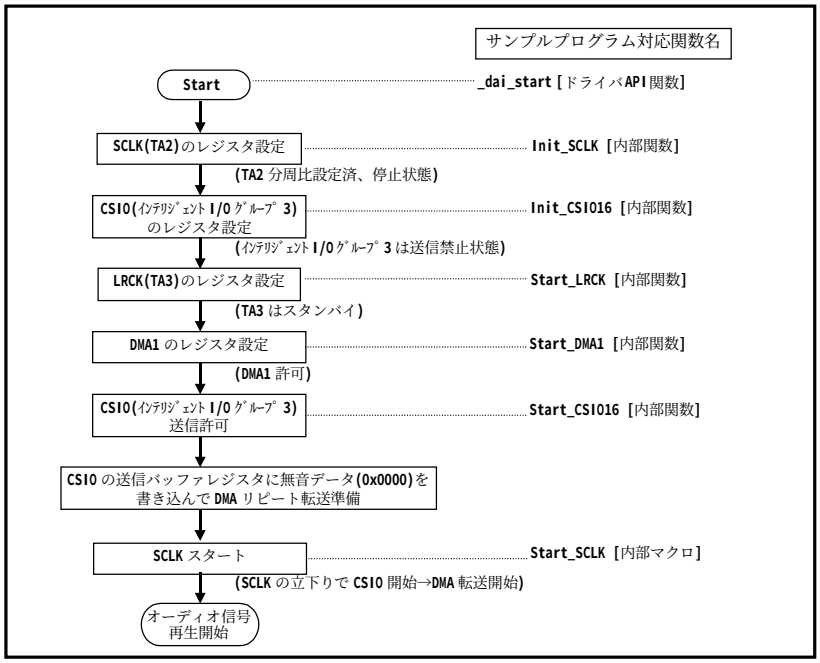
8 ビット CSIO 使用時は、外部クロック選択時かつ CLKi 極性を”立下り”を選択した場合、送信開始条件として、CLKi 端子の入力が”H”であることが必要です。このため、作成例では SCLK(TA2)の初期化の際に、CLKi(=SCLK=TA2 のパルス出力)を一度”H”に吊り上げて、その間に DMA トリガとなる 1 発目の無音データを送信バッファに書き込んでいます。
この SCLK(TA2 パルス出力)の初期化部分について以下に図示します。



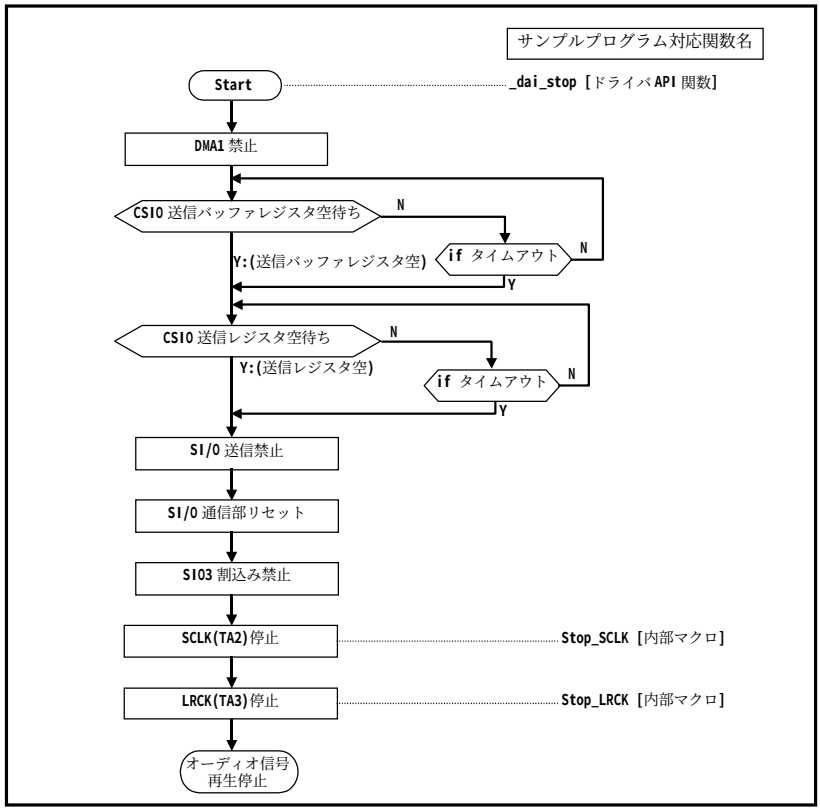
4.2.2 デジタルオーディオ信号停止 設定フロー（8 ビット CSIO）



4.2.3 デジタルオーディオ信号再生開始 設定フロー（16 ビット CSIO）



4.2.4 デジタルオーディオ信号停止 設定フロー（16 ビット CSIO）



5. ドライバ関数仕様

ここでは、作成例のデジタルオーディオ I/F ドライバ関数仕様を示します。
作成例のドライバは以下のように分類されます。
(1) API(Application Program Interface)関数：アプリケーションプログラムから Call される関数
(2) 内部関数と内部マクロ：ドライバ内部から Call される関数

関数名	区分	処理概要
_dai_start	API	・デジタルオーディオデータのストリームを DAC に出力開始します。
_dai_stop	API	・オーディオデータのストリーム出力を停止します。 ・DMA 転送を禁止、CSIO を送信禁止にします。 ・SCLK, LRCK を停止します。
_cnv_endian	API（8 ビット CSIO）	・8 ビット CSIO を使用する際のエンディアン変換を行います。
Init_SCLK	内部関数	・SCLK のレジスタ設定を行います。
Start_LRCK	内部関数	・LRCK のレジスタ設定を行います。 ・LRCK はスタート状態でスタンバイしています。
Init_CSIO8	内部関数 （8 ビット CSIO）	・UART3 を 8 ビット CSIO に設定します。 ・外部クロック SCLK を同期クロックとします。
Start_DMA0	内部関数 （8 ビット CSIO）	・8 ビット CSIO 使用時の DMA 設定を行います。 ・リピート転送モードでスタンバイしており、 CSIO の送信バッファ空割込み要求で送信開始します。
Start_CSIO8	内部関数 （8 ビット CSIO）	・Init_CSIO8 で設定した CSIO を送信許可状態にします。
Init_CSIO16	内部関数 （16 ビット CSIO）	・インテリジェント I/O グループ 3 を 16 ビット CSIO に設定します。 ・外部クロック SCLK を同期クロックとします。
Start_DMA1	内部関数 （16 ビット CSIO）	・16 ビット CSIO 使用時の DMA 設定を行います。 ・リピート転送モードでスタンバイしており、 CSIO の送信バッファ割込み要求で転送開始します。
Start_CSIO16	内部関数 （16 ビット CSIO）	・Init_CSIO16 で設定した CSIO を送信許可状態にします。

関数名	_dai_start		
区分	API		
機能	・ユーザの指定したバッファに以下のデータを読み込んで本関数を Call すると、 デジタルオーディオデータのストリーム出力を開始します。 *buf0 ← 無音データ(0h) *buf1 ← デジタルオーディオデータ ・引数 buf0 に格納されたデータから出力します。		
宣言	void _dai_start(unsigned short far *buf0, unsigned short far *buf1, unsigned short size, unsigned short zero_cnt)		
仮引数	*buf0	I	バッファ０の先頭アドレス（バッファ０はユーザ定義）
	*buf1	I	バッファ１の先頭アドレス（バッファ１はユーザ定義）
	size	I	バッファのサイズ=*buf1 に格納したデータサイズ (WORD 単位で指定して下さい)
	zero_cnt	I	無音データの出力カウントを設定します（＜バッファサイズ） 8 ビット CSIO 使用時：1+4n 回(n=1,2,...)を設定します。 16 ビット CSIO 使用時：2n 回(n=1,2,...)を設定します。 無音データ出力カウントと L,R チャネル出力の関係については図 6 を参照下さい。
戻り値	なし		
下位関数	Init_SCLK; 内部関数 Start_LRCK; 内部関数 Start_CSIO8／Start_CSIO16;内部関数 Start_DMA0／Start_DMA1;内部関数		
注意	引数 zero_cnt に０を設定しないで下さい。 *buf0 に無音データ(＝０)、*buf1 に音楽データを格納したバッファのポインタを指定して下さい。		

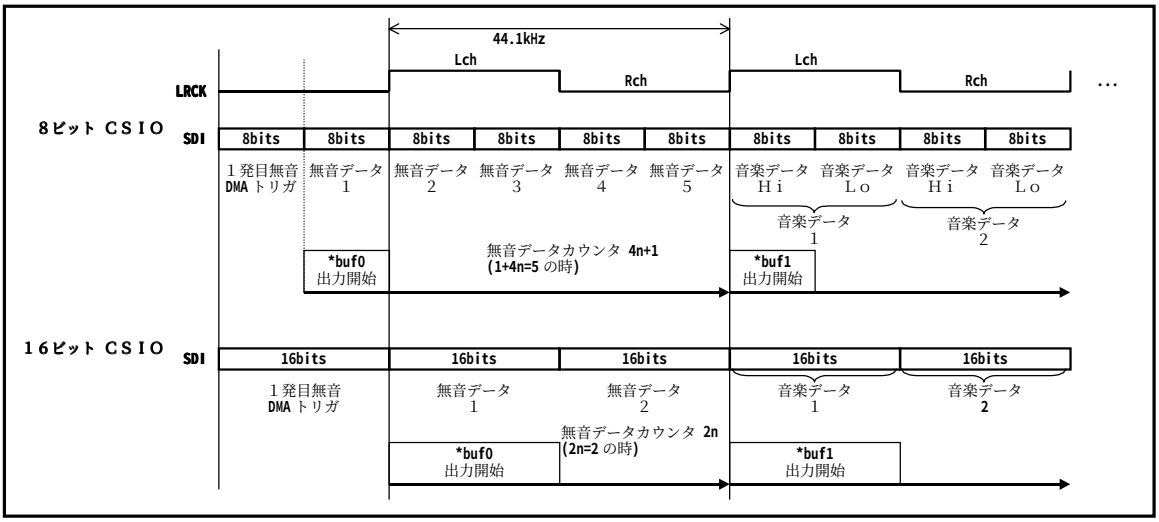


図 6.無音データ出力カウントと L,R チャネル出力

【解説】
(1) 本関数の各引数は以下のレジスタに設定されます。

引数	設定先レジスタ
*buf0	DMA メモリアドレスレジスタ（転送元）
*buf1	DMA メモリアドレスリロードレジスタ（転送元）
size	DMA カウントリロードレジスタ
zero_cnt	DMA カウントレジスタ

(2) 「図 6」で示される 1 発目無音データ DMA トリガは、シリアル送信バッファに無音データ(0h)を書き込むことで、リピート DMA 転送を開始する為のトリガとしています。
(3) 本関数がアプリケーションプログラムから Call されると、
(i)無音データが格納されている*buf0 から、zero_cnt で指定した回数だけ無音データを出力し始めます。
(ii)無音データが出力完了した時点で DMA 割込み要求が入り、*buf1 のアドレスと size のカウント値が、それぞれアドレスレジスタ、カウントレジスタにリロードされ、*buf1 に格納してある音楽データが出力し始めます。

関数名	_dai_stop		
区分	API		
機能	・デジタルオーディオデータのストリーム再生を停止します。 ・デジタルオーディオ I/F に使用していた DMA 転送を禁止、CSIO 転送を禁止します。 ・SCLK,LRCK を停止します。		
宣言	void _dai_stop(void)		
仮引数	なし		
戻り値	なし		
下位関数	なし		
補足	(1) DMA 転送カウントレジスタ、DMA 転送カウントリロードレジスタについては DMA が停止した時点の値が残ります。 (2) 本関数内部では、CSIO 送信バッファレジスタと送信レジスタが空になるのを待ちます。		

関数名	_cnv_endian		
区分	API		
機能	・2 バイト(WORD)データのエンディアン変換を行います。 ・8 ビット CSIO を使用する場合のみ必要な処理です。		
宣言	void _cnv_endian(unsigned char far *ram_buff, unsigned short Bsize) #pragma PARAMETER _cnv_endian(A0, R1)		
仮引数	*ram_buff	I/O	エンディアン変換するデータを格納したバッファのアドレス
	Bsize	I	エンディアン変換するバイト数（偶数であること）
戻り値	なし		
下位関数	なし		
注意	(1) Bsize には偶数バイト数を指定して下さい。 (2) Bsize には、2～(FFFFh-1)まで指定可能です。 (3) Bsize に 0 を指定しないで下さい。		
補足	(1) M32C/80 シリーズはリトルエンディアンのマイコンです。本作成例で扱っている音楽データ長は１チャネル当たり 16 ビットであるため、バッファに読み込んでからエンディアン変換を行う必要があります。 (2) 本作成例で扱っている音楽データは、Lch,Rch(32 ビット=4 バイト)で１単位となっている為、バッファサイズは４で割り切れるサイズを推奨します。		

関数名	Init_SCLK		
区分	内部関数		
機能	・SCLK(TA2)のレジスタ設定を行います。 ・本関数では TA2 は停止状態にあります。		
宣言	void Init_SCLK(void)		
仮引数	なし		
戻り値	なし		
下位関数	なし		
補足	(1) TA2 の割込みレベルは 0（割り込み禁止）にしています。		

関数名	Start_LRCK		
区分	内部関数		
機能	・LRCK(TA3)のレジスタ設定を行います。 ・本関数では TA3 のレジスタ設定後、TA3 をスタート状態にしています。 (LRCK は SCLK のイベントをカウントするので、SCLK がスタートするまで LRCK は停止しています)		
宣言	void Start_LRCK(void)		
仮引数	なし		
戻り値	なし		
下位関数	なし		
補足	(1) TA3 の割込みレベルは 0（割り込み禁止）にしています。		

関数名	Init_CSIO8		
区分	内部関数（8 ビット CSIO 使用時）		
機能	・UART3 を 8 ビットクロック同期形シリアルモードに設定します。 ・外部クロック SCLK を同期クロックとします。 ・クロックの立下りでデータ送出します。		
宣言	void Init_CSIO8(void)		
仮引数	なし		
戻り値	なし		
下位関数	なし		
補足	(1) 8 ビット CSIO 使用時は、外部クロック選択時かつ CLKi 極性選択が”立下り”の場合、送信開始条件として CLKi 端子の入力が”H”であることが必要です。		

関数名	Start_DMA0		
区分	内部関数（8 ビット CSIO 使用時）		
機能	・8 ビット CSIO 使用時の DMAC 設定を DMA0 に設定します。 ・本関数を Call すると、8 ビット CSIO 送信バッファレジスタ空の割込み要求で DMA 転送開始する状態でスタンバイします。		
宣言	void Start_DMA0(unsigned short far *buf0, unsigned short far *buf1, unsigned shrot size, unsigned short zero_cnt)		
仮引数	*buf0	I	バッファ 0 の先頭アドレス（バッファ 0 はユーザ定義）
	*buf1	I	バッファ 1 の先頭アドレス（バッファ 1 はユーザ定義）
	size	I	バッファのサイズ(WORD 単位)
	zero_cnt	I	無音データの出力カウント
戻り値	なし		
下位関数	なし		
補足	(1) DMA0 の割込みレベルは 7 に設定しています。 (2) 各引数は、上位関数の_dai_start から渡されます。 (3) DMA モードレジスタ 0 は、DMA0 と DMA1 の両方の設定ビットを含んでいますが、本関数では、DMA0 のみを設定します。(DMA1 の設定は保存されます)		

関数名	Start_CSIO8		
区分	内部関数（8 ビット CSIO 使用時）		
機能	・Init_CSIO8 で設定した CSIO を送信許可状態にします。		
宣言	void Start_CSIO8(void)		
仮引数	なし		
戻り値	なし		
下位関数	なし		
補足			

関数名	Init_CSIO16		
区分	内部関数（16 ビット CSIO 使用時）		
機能	・ｲﾝﾃﾘｼﾞｪﾝﾄ I/O ｸﾞﾙｰﾌﾟ 3 を 16 ビットクロック同期形シリアルモードに設定します。 ・外部クロック SCLK を同期クロックとします。 ・クロックの立下りでデータ送出します。		
宣言	void Init_CSIO16(void)		
仮引数	なし		
戻り値	なし		
下位関数	なし		
補足	(1) 本関数で使用するｲﾝﾃﾘｼﾞｪﾝﾄ I/O ｸﾞﾙｰﾌﾟ 3 をクロック同期形シリアルモードで使用する場合、クロックの立下りでデータ送出します。(データ送出時のクロック極性は立下りのみです)		

関数名	Start_DMA1		
区分	内部関数（16 ビット CSIO 使用時）		
機能	・16 ビット CSIO 使用時の DMAC 設定を DMA1 に設定します。 ・本関数を Call すると、16 ビット CSIO 送信バッファレジスタ空の割込み要求で DMA 転送開始する状態でスタンバイします。		
宣言	void Start_DMA1(unsigned short far *buf0, unsigned short far *buf1, unsigned shrot size, unsigned short zero_cnt)		
仮引数	*buf0	I	バッファ 0 の先頭アドレス（バッファ 0 はユーザ定義）
	*buf1	I	バッファ 1 の先頭アドレス（バッファ 1 はユーザ定義）
	size	I	バッファのサイズ(WORD 単位)
	zero_cnt	I	無音データの出力カウント
戻り値	なし		
下位関数	なし		
補足	(1) DMA1 の割込みレベルは 7 に設定しています。 (2) 各引数は、上位関数の_dai_start から渡されます。 (3) DMA モードレジスタ 0 は、DMA0 と DMA1 の両方の設定ビットを含んでいますが、本関数では、DMA1 のみを設定します。(DMA0 の設定は保存されます)		

関数名	Start_CSIO16		
区分	内部関数（16 ビット CSIO 使用時）		
機能	・Init_CSIO16 で設定した CSIO を送信許可状態にします。		
宣言	void Start_CSIO16(void)		
仮引数	なし		
戻り値	なし		
下位関数	なし		
補足			

6. デジタルオーディオ I/F ドライバ サンプルプログラム

ここでは、「5.ドライバ関数仕様」で示したデジタルオーディオ I/F ドライバ作成例のプログラムを示します。本プログラムはサンプルプログラムであり、各ユーザアプリケーションに応じた変更及び調整が必要です。また、このサンプルプログラムは任意のシステム/アプリケーションにおいて動作を保証するものではありません。

■ ファイル構成

ファイル名	内容
dai_drv.h	アプリケーション用インクルードファイル ドライバ関数をアプリケーションで使用する為にアプリケーションソースからインクルードするファイルです。
C_dai_drv.c	デジタルオーディオ I/F ドライバサンプルプログラム
A_endian.a30	2 バイト(WORD)データのエンディアン変換サンプルプログラム

■ コンパイルオプション
16 ビット CSIO を使用する場合は、”CSIO16”を定義して下さい。
サンプルプログラムでは”CSIO16”が未定義の場合、8 ビット CSIO を使用したドライバを構築します。
以下に、”CSIO16”を定義する例を示します。

(例 1) サンプルプログラム中で、#define CSIO16 を記述することにより定義する
(例 2) NC308 の”-D”オプションを使用して、コンパイル時に -DCSIO16 として定義する

dai_drv.h

```
/*""FILE COMMENT""*****************************************************************
 * System Name : Digital Audio I/F Driver sample
 * File Name   : dai_drv.h
 * Version     : 0.01
 * Contents    : Digital Audio I/F Driver sample include file
 * CPU         : M32C/80 series
 * Compiler    : NC308WA (V.3.10 Release 3)
 * OS          :
 * Note        : (1) The 16 bits CSIO can use only on the M32C/83 group.
 *               (2) When using 16 bits CSIO, define "CSIO16".
 *                   This sample default generates the driver using 8 bits CSIO.
 *
 ***********************************************************************************
 * Copyright(C)2003, Renesas Technology Corp.
 * Copyright(C)2003, Renesas Solutions Corp.
 * All rights reserved.
 ***********************************************************************************
 * History     : 2002.10.01 Ver 0.01
 *             :
 *""FILE COMMENT END""*************************************************************/


/*--------------------------------------*/
/*  Defines                             */
/*--------------------------------------*/
```

```
/****************************************/
/*  Prototypes                          */
/****************************************/
extern void _dai_start( unsigned short far *buf0,
                        unsigned short far *buf1,
                        unsigned short size,
                        unsigned short zero_cnt );

extern void _dai_stop( void );
extern void _dai_pause( void );


#if !defined (CSIO16)
extern void _cnv_endian( unsigned char far *ram_buff, unsigned short Bsize );
#pragma PARAMETER _cnv_endian( A0, R1 )
#endif


/****************************************/
/*  Global variables                    */
/****************************************/
```

C_dai_drv.c

```
/*""FILE COMMENT""*****************************************************************
 * System Name : Digital Audio I/F Driver sample
 * File Name   : C_dai_drv.c
 * Version     : 0.02
 * Contents    : Digital Audio I/F Driver sample programs
 * CPU         : M32C/80 series
 * Compiler    : NC308WA (V.3.10 Release 3)
 * OS          :
 * Note        : (1) The 16 bits CSIO can use only on the M32C/83 group.
 *               (2) When using 16 bits CSIO, define "CSIO16".
 *                   This sample default generates the driver using 8 bits CSIO.
 *
 ***********************************************************************************
 * Copyright(C)2003, Renesas Technology Corp.
 * Copyright(C)2003, Renesas Solutions Corp.
 * All rights reserved.
 ***********************************************************************************
 * History     : 2002.10.01 Ver 0.01
 *             : 2002.10.25 Ver 0.02
 *             :
 *""FILE COMMENT END""*************************************************************/


/****************************************/
/* Include headers                      */
/****************************************/
#include "sfr32c83.h"   /* File contains SFR (Special Function Register) definitions for M32C/83 */
```

```
/****************************************/
/* Defines                              */
/****************************************/
#define Start_SCLK      ta2s = 1
#define Stop_SCLK       ta2s = 0
#define Stop_LRCK       ta3s = 0


/****************************************/
/* Global variables                     */
/****************************************/


/****************************************/
/* Prototypes                           */
/****************************************/
void _dai_start( unsigned short far *buf0,
                 unsigned short far *buf1,
                 unsigned short size,
                 unsigned short zero_cnt );
void _dai_stop( void );


void Init_SCLK( void );
void Start_LRCK( void );

#if defined (CSIO16)
/******* 16bits CSIO ****************************************************/
    void Init_CSIO16( void );
    void Start_CSIO16(void);
    void Start_DMA1( unsigned short far *buf0,
                     unsigned short far *buf1,
                     unsigned short size,
                     unsigned short init_cnt );

#else
/******* 8bits CSIO *****************************************************/
    void Init_CSIO8( void );
    void Start_CSIO8( void );
    void Start_DMA0( unsigned short far *buf0,
                     unsigned short far *buf1,
                     unsigned short size,
                     unsigned short zero_cnt );
#endif
/************************************************************************/
```

```
/*""FUNC COMMENT""************************************************************
 * ID           :
 * Abstract     : Start of digital audio data play
 *-----------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-----------------------------------------------------------------------------
 * Declaration  : void _dai_start( unsigned short far *buf0,
 *              :                  unsigned short far *buf1,
 *              :                  unsigned short size,
 *              :                  unsigned short zero_cnt )
 *-----------------------------------------------------------------------------
 * Function     : Start outputting digital audio data streaming to the DAC.
 *              : This function needs to be called after storing following data in buffers.
 *              :     buf0 <-- silence data (0 clearance)
 *              :     buf1 <-- digital audio data
 *-----------------------------------------------------------------------------
 * Argument     : unsigned short far *buf0  ; Top address of the buffer 0 defined by user
 *              : unsigned short far *buf0  ; Top address of the buffer 1 defined by user
 *              : unsigned short size       ; Size of buffer(Specify in WORD unit)
 *              : unsigned short zero_cnt   ; Sets the number of outputting silence data
 *-----------------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *-----------------------------------------------------------------------------
 * Subroutines  : Init_SCLK
 *              : Init_CSI08 / Init_CSI016
 *              : Start_LRCK
 *              : Start_DMA0 / Start_DMA1
 *              : Start_CSI08 / Start_CSI016
 *-----------------------------------------------------------------------------
 * Note         : Case of 8bits CSI0
 *              :    To the zero_cnt, set the value 1+4n(n=1,2,...)
 *              : Case of 16bits CSI0
 *              :    To the zero_cnt, set the value 2n(n=1,2,...)
 *              :
 *              : Before calling this function, initialize the DAC.
 *-----------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""********************************************************/
#define SHORT_TIMER 5           /* <-- Changes and adjustment which tuned to the user system are necessity */
#define LONG_TIMER  10000       /* <-- Changes and adjustment which tuned to the user system are necessity */
void _dai_start( unsigned short far *buf0,  /* Top address of the buffer 0 defined by user */
                 unsigned short far *buf1,  /* Top address of the buffer 0 defined by user */
                 unsigned short size,       /* Size of the buffer (Specify in WORD unit) */
                 unsigned short zero_cnt )  /* Sets the number of outputting silence data */
{
    /* Set the SCLK(TA2) registers */
    Init_SCLK();


#if defined (CSI016)
/******* 16bits CSI0 ********************************************************/
    /* Set the CSI0 (Intelligent I/O group 3) registers */
    Init_CSI016();
#else
/******* 8bits CSI0 *********************************************************/
    /* Set the CSI0 (UART3) registers */
    Init_CSI08();
#endif
/*****************************************************************************/
```

```
    /* Set the LRCK (TA3) registers, The LRCK is stopping until the SCLK starts */
    Start_LRCK();


#if defined (CSIO16)
/******* 16bits CSIO ****************************************************/
    /* Set the DMA1 registers */
    Start_DMA1( buf0, buf1, size, zero_cnt );

    /* Enable transmission of CSIO */
    Start_CSIO16();

    /* By writing the silence data, start repeating DMA transfer */
    g3tb = 0x0000;

    /* Start the SCLK */
    Start_SCLK;
#else
/******* 8bits CSIO *****************************************************/
    /* Set the DMA0 registers */
    Start_DMA0( buf0, buf1, size, zero_cnt );

    /* Enable transmission of CSIO */
    Start_CSIO8();

    /* Starts processing of the SCLK */
    ta2 = (SHORT_TIMER -1 );    /* Set the initial divide ratio SHORT_TIMER */
    Start_SCLK;                 /* Start the SCLK with SHORT_TIMER and output pulse is lifted to H */
    ta2 = (LONG_TIMER -1 );     /* Set the initial divide ratio LONG_TIMER to the reload-register and reload at next time */
    while( ir_ta2ic != 1 );     /* Wait finishes counting the SHORT_TIMER */
                                /* Start counting LONG_TIMER in H level */
    ta2 = (2-1);                /* In this meantime, set divide ratio 2 to SCLK (TA2) and set transmit-buffer-register */

    /* By writing the silence data, start repeating DMA transfer */
    u3tbl = 0x00;
    ir_ta2ic = 0;
#endif
/************************************************************************/

}



/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Abstract     : Initialize the SCLK (TA2)
 *-----------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-----------------------------------------------------------------------
 * Declaration  : void Init_SCLK( void )
 *-----------------------------------------------------------------------
 * Function     : Sets the SCLK (TA2) registers.
 *                SCLK (TA2) is stopping in this function.
 *
 *                  Mode; Event counter mode
 *                  Count source; Counts the rising edge of external signal which
 *                                inputted into TA2IN pin (MCLK --> TA2IN pin)
 *                  Divide ratio; dividing external clock (MCLK) by 4
 *                  TAiIN pin function; Set TA2IN pin as input port
 *                  TAiOUT pin function; Pulse output
 *-----------------------------------------------------------------------
```

```
 * Argument     : None
 *------------------------------------------------------------------------
 * Return value : None
 *------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *------------------------------------------------------------------------
 * Subroutines  : None
 *------------------------------------------------------------------------
 * Note         :
 *------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""*****************************************************/
void Init_SCLK( void )
{
    /* Stop counting */
    ta2s = 0;

    /* Disable interruption */
    ta2ic = 0;

    /* Set mode */
    ta2mr = (0x09); /* Timer Ai mode register */
        /* b[1:0]; Operation mode                [01;Event counter mode] */
        /* b[2];   invalid                       [X; invalid in M32C/80 series] */
        /* b[3];   Count polarity select         [1; Counts external signal's rising edges] */
        /* b[4];   Up/down switching case select  [0; Up/down flag's content] */
        /* b[5];   Set to 0 in Event counter mode [0; (Set to 0)] */
        /* b[6];   Count operation type select    [0; Reload type] */
        /* b[7];   Two-phase pulse signal processing operation select [0; fixed on TA2] */

    /* Trigger select */
    ta2tgl = 0;     /* Input on TA2IN is selected */
    ta2tgh = 0;

    /* TA2IN pin function */
    p7_5 = 0;       /* Set TA2IN pin as input port */
    pd7_5 = 0;


    /* TA2OUT pin function */
    psc_4 = 0;      /* function select register C   : Timer output (TA2OUT) */
    psl1_4 = 0;     /* function select register B1  : Function that was selected in PSC_4 */
    ps1_4 = 1;      /* function select register A1  : Function that was selected in PSL1_4 */


#if defined (CSI016)
    ta2 = (2-1);    /* Generate the SCLK by dividing MCLK by 4 and outputting pulse to TA2OUT pin */
#endif

}
```

```
/*""FUNC COMMENT""************************************************************
 * ID           :
 * Abstract     : Start LRCK(TA3)
 *-----------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-----------------------------------------------------------------------------
 * Declaration  : void Start_LRCK( void )
 *-----------------------------------------------------------------------------
 * Function     : Sets the LRCK (TA3) registers.
 *                After setting registers of TA3, TA3 is started in this function.
 *                The LRCK counts event (falling edge) of the SCLK, so the LRCK is
 *                stopping until the SCLK starts.
 *
 *                  Mode; Event counter mode
 *                  Count source; Counts the falling edge of the external signal which
 *                                inputted into TA3IN pin.
 *                  Divide ratio; dividing external clock(SCLK) by 32
 *                  TAiIN pin function; Set TA3IN pin as input port
 *                  TAiOUT pin function; Pulse output
 *-----------------------------------------------------------------------------
 * Argument     : None
 *-----------------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *-----------------------------------------------------------------------------
 * Subroutines  : None
 *-----------------------------------------------------------------------------
 * Note         :
 *-----------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""*********************************************************/
void Start_LRCK( void )
{
    /* Stop counting */
    ta3s = 0;

    /* Disable interruption */
    ta3ic = 0;

    /* Set mode */
    ta3mr = (0x01); /* Timer Ai mode register */
        /* b[1:0]; Operation mode                [01;Event counter mode] */
        /* b[2];   invalid                       [X; invalid in M32C/80 series] */
        /* b[3];   Count polarity select         [0; Counts external signal's falling edges] */
        /* b[4];   Up/down switching case select  [0; Up/down flag's content] */
        /* b[5];   Set to 0 in Event counter mode [0; (Set to 0)] */
        /* b[6];   Count operation type select    [0; Reload type] */
        /* b[7];   Two-phase pulse signal processing operation select   [0; not using two-phase pulse signal] */

    /* Trigger select */
    ta3tgl = 0;     /* Input on TA3IN is selected */
    ta3tgh = 0;

    /* TA3IN pin function */
    p7_7 = 0;       /* Set TA3IN pin as input port */
    pd7_7 = 0;

    /* TA3OUT pin function */
    psl1_6 = 1;     /* function select register B1  : Timer output (TA3OUT) */
    psl_6 = 1;      /* function select register A1  : Function that was selected in PSL1_6 */
```

```
    /* Set the initial divide ratio */
    ta3 = 17-1;

    /* Start timer */
    ta3s = 1;
    ta3 = 16-1;         /* divide ratio = 32 (Set the re-load register) */
}



#if defined (CSI016)
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Abstract     : Initialize the Intelligent I/O group 3 as 16 bits CSIO
 *-----------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-----------------------------------------------------------------------------
 * Declaration  : void Init_CSI016( void )
 *-----------------------------------------------------------------------------
 * Function     : Set the Intelligent I/O group 3 registers as 16 bits CSIO.
 *              : Uses the external clock SCLK as synchronous clock.
 *              : Transmit data is output at falling edge of transfer clock.
 *-----------------------------------------------------------------------------
 * Argument     : None
 *-----------------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *-----------------------------------------------------------------------------
 * Subroutines  : None
 *-----------------------------------------------------------------------------
 * Note         :
 *-----------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""*********************************************************/
void Init_CSI016( void )
{

    /* Set Base timer */
    g3bcr0 = 0x7F;  /* Group 3 base timer control register 0 */
        /* b[1:0]; Count source select                 [11;    f1] */
        /* b[6:2]; Count source division ratio select  [11111; No division] */
        /* b[7];   Base timer interrupt select         [0;     Bit 15 overflow] */

    g3bcr1 = 0x00;  /* Group 3 base timer control register 1 */
        /* b[0]; Base timer reset cause select         [0; Synchronize the base timer 2 reset without resetting the timer] */
        /* b[1]; Base timer reset cause select         [0; Does not reset the base timer when it matches WG register ch0] */
        /* b[2]; Reserved bit                          [0; (Set to 0)] */
        /* b[3]; -                                     [0; (Set to 0)] */
        /* b[4]; Base timer start                      [0; Base timer reset] */
        /* b[6:5]; -                                   [00;(Set to 0)] */
        /* b[7]; Parallel real-time port function select [0; Not use] */

    /* Group 3 waveform generation control register 0,1 */
    g3pocr0 = 0x07; /* Group 3 WG control register 0 */
        /* b[2:0]; Operation mode select               [111; Assigns communication output to a port] */
        /* b[3];   Parallel RTP output trigger select  [0;   Match of WG register j isn't trigger] */
        /* b[4];   Output initial value select         [0;   Outputs 0 as the initial value] */
        /* b[5];   Reload timing select                [0;   Reload a new count when CPU writes the count] */
        /* b[6];   RTP function select                 [0;   Not use] */
        /* b[7];   Inverted output function select     [0;   Output is not inverted] */
```

```
    g3pocr1 = 0x07; /* Group 3 WG control register 1 */
        /* b[2:0]; Operation mode select              [111; Assigns communication output to a port] */
        /* b[3];   Parallel RTP output trigger select  [0;   Match of WG register j isn't trigger] */
        /* b[4];   Output initial value select         [0;   Outputs 0 as the initial value] */
        /* b[5];   Reload timing select                [0;   Reload a new count when CPU writes the count] */
        /* b[6];   RTP function select                 [0;   Not use] */
        /* b[7];   Inverted output function select     [0;   Output is not inverted] */


    /* Group3 channel i operation */
    g3fe = 0x03;    /* Group 3 function enable register */
        /* b[7:0]; Ch i function enable bit            [0000 0011; Enables function on ch0 and ch1] */


    /* Communication mode */
    g3mr = 0x45;    /* Group 3 SI/O communication mode register */
        /* b[1:0]; Communication mode select           [01; Serial I/O mode] */
        /* b[2];   Internal/external clock select      [1;  External clock] */
        /* b[3];   Transmission data length select     [0;  16 bits] */
        /* b[5:4]; -                                   [00; (Set to 0)] */
        /* b[6];   Transfer direction select           [1;  MSB first] */
        /* b[7];   Transmit interrupt cause select     [0;  Transmit buffer is empty] */


    /* Communication control */
    g3cr = 0x00;    /* Group 3 SI/O communication control register */
        /* b[0]; Transmit enable bit                   [0; Transmission disabled] */
        /* b[1]; Transmit register empty flag          [0; Data present in transmit register (R-Only)] */
        /* b[2]; Transmit buffer empty flag            [0; Data present in transmit buffer register (R-Only)] */
        /* b[3]; -                                     [0; (Set to 0)] */
        /* b[4]; Receive enable bit                    [0; Reception disabled] */
        /* b[5]; Receive complete flag                 [0; No data present in receive buffer register (R-Only)] */
        /* b[6]; TxD output polarity reverse select    [0; No reverse] */
        /* b[7]; RxD input polarity reverse select     [0; No reverse] */


    /* Set the input pin which inputs external clock (ISCLK3; P12_1) as the input port */
    ps6_1 = 0;              /* P12_1 ; Input port */

    /* Set the TxD pin (ICTxD3; P12_0) as the II/O group 3 output */
    ps6_0 = 1;              /* P12_0 ; II/O group 3 output (OUTC30) */


    /* Set the interrupt request bit of the II/O SI03t */
    iio10ie = 0x10;         /* Enable the interrupt request from communication function of II/O group 3 */
}


/*""FUNC COMMENT""*****************************************************************
 * ID           :
 * Abstract     : Start the Intelligent I/O group 3 as 16bits CSIO
 *-------------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-------------------------------------------------------------------------------
 * Declaration  : void Start_CSIO16( void )
 *-------------------------------------------------------------------------------
 * Function     : Enables CSIO transmission which is set by the Init_CSIO16 function
 *-------------------------------------------------------------------------------
 * Argument     : None
 *-------------------------------------------------------------------------------
 * Return value : None
```

```
 *-------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *-------------------------------------------------------------------------
 * Subroutines  : None
 *-------------------------------------------------------------------------
 * Note         :
 *-------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""*****************************************************/
void Start_CSI016( void )
{
    /* Communication control */
    g3cr = 0x01;    /* Group 3 SI/O communication control register */
        /* b[0]; Transmit enable bit                    [1; Transmission enabled] */
        /* b[1]; Transmit register empty flag           [0; Data present in transmit register (R-Only)] */
        /* b[2]; Transmit buffer empty flag             [0; Data present in transmit buffer register (R-Only)] */
        /* b[3]; -                                      [0; (Set to 0)] */
        /* b[4]; Receive enable bit                     [0; Reception disabled] */
        /* b[5]; Receive complete flag                  [0; No data present in receive buffer register (R-Only)] */
        /* b[6]; TxD output polarity reverse select     [0; No reverse] */
        /* b[7]; RxD input polarity reverse select      [0; No reverse] */

}


#else
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Abstract     : Initialize the UART3 as 8bits CSIO
 *-------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-------------------------------------------------------------------------
 * Declaration  : void Init_CSI08( void )
 *-------------------------------------------------------------------------
 * Function     : Set the UART3 registers as 8 bits clock synchronous serial I/O mode.
 *              : Uses the external clock SCLK as synchronous clock.
 *              : Transmit data is output at falling edge of transfer clock
 *-------------------------------------------------------------------------
 * Argument     : None
 *-------------------------------------------------------------------------
 * Return value : None
 *-------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *-------------------------------------------------------------------------
 * Subroutines  : None
 *-------------------------------------------------------------------------
 * Note         :
 *-------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""*****************************************************/
void Init_CSI08( void )
{
    /* Disable transmission */
    u3c1 = 0x02;    /* UARTi transmit/receive control register 1 */
        /* b[0]; Transmit enable bit                    [0; Transmission disabled] */
        /* b[1]; Transmit buffer empty flag             [1; No data present in transmit buffer register (R only)] */
        /* b[2]; Receive enable bit                     [0; Reception disabled] */
        /* b[3]; Receive complete flag                  [0; Data present in receive buffer register (R only)] */
        /* b[4]; UARTi transmit interrupt cause select  [0; Transmit buffer empty] */
        /* b[5]; UARTi continuous receive mode enable bit [0; Continuous receive mode disabled] */
        /* b[6]; Data logic select bit                  [0; No reverse] */
        /* b[7]; Clock divide synchronizing stop bit    [0; Synchronizing stop] */
```

```
    /* Set transmit/receive mode */
    u3mr = 0x09;    /* UARTi transmit/receive mode register */
        /* b[2:0]; Serial I/O mode select               [001; Serial I/O mode (Clock synchronous)] */
        /* b[3];   Internal/external clock select       [1;   External clock] */
        /* b[4];   Stop bit length select               [X;   Invalid] */
        /* b[5];   Odd/even parity select               [X;   Invalid] */
        /* b[6];   Parity enable bit                    [X;   Invalid] */
        /* b[7];   TxD, RxD input/output polarity switch bit [0; No reversed] */

    u3c0 = 0x98;    /* UARTi transmit/receive control register 0 */
        /* b[1:0]; BRG count source select              [00; f1 is selected] */
        /* b[2];   CTS/RTS function select              [0;  See bit 4, valid when bit 4 = 0] */
        /* b[3];   Transmit register empty flag         [1;  No data present in transmit register (R only)] */
        /* b[4];   CTS/RTS disable bit                  [1;  CTS/RTS function disabled] */
        /* b[5];   Data output select bit               [0;  TxDi pin is CMOS output] */
        /* b[6];   CLK polarity select bit              [0;  Transmit data is output at falling edge of transfer clock] */
        /* b[7];   Transfer format select bit           [1;  MSB first] */


    /* Set bit rate */
    u3brg = 0x00;   /* UARTi bit rate generator */
        /* b[7:0]; Assuming that set value = n, BRGi divides the count source by n+1 [0; f1] */


    /* Set the CLK3 pin (P9_0) as the I/O port */
    prc2 = 1;       /* Release of the protection (Enable writing) */
    ps3_0 = 0;      /* Set port P9_0 as I/O port */
                    /* (Input direction when reset) */

    /* Set the TxD3 pin (P9_2) as the UART3 output */
    psl3_2 = 0;     /* UART3 output */
    prc2 = 1;       /* Release of the protection (Enable writing) */
    ps3_2 = 1;      /* Function that was selected in PSL3_2 */
}


/*""FUNC COMMENT""************************************************************
 * ID           :
 * Abstract     : Start the UART3 as 8bits CSIO
 *-----------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-----------------------------------------------------------------------------
 * Declaration  : void Start_CSIO8( void )
 *-----------------------------------------------------------------------------
 * Function     : Enables CSIO transmission which is set by the Init_CSIO8 function
 *-----------------------------------------------------------------------------
 * Argument     : None
 *-----------------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *-----------------------------------------------------------------------------
 * Subroutines  : None
 *-----------------------------------------------------------------------------
 * Note         :
 *-----------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""********************************************************/
void Start_CSIO8( void )
{
```

```
    /* Enable transmission */
    u3c1 = 0x03;    /* UARTi transmit/receive control register 1 */
        /* b[0]; Transmit enable bit                    [1; Transmission enabled] */
        /* b[1]; Transmit buffer empty flag             [1; No data present in transmit buffer register (R only)] */
        /* b[2]; Receive enable bit                     [0; Reception disabled] */
        /* b[3]; Receive complete flag                  [0; Data present in receive buffer register (R only)] */
        /* b[4]; UARTi transmit interrupt cause select  [0; Transmit buffer empty] */
        /* b[5]; UARTi continuous receive mode enable bit [0; Continuous receive mode disabled] */
        /* b[6]; Data logic select bit                  [0; No reverse] */
        /* b[7]; Clock divide synchronizing stop bit    [0; Synchronizing stop] */
}
#endif


#if defined (CSIO16)
/*""FUNC COMMENT""***************************************************************
 * ID           :
 * Abstract     : Set DMA1 and enable DMA1
 *-------------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-------------------------------------------------------------------------------
 * Declaration  : void Start_DMA1( unsigned short far *buf0,
 *              :                  unsigned short far *buf1,
 *              :                  unsigned short size,
 *              :                  unsigned short zero_cnt )
 *-------------------------------------------------------------------------------
 * Function     : Set DMA1 as DMAC when using 16 bits CSIO.
 *              : By calling this function, the DMA will be in a standby state
 *                which starts transmission by the interruption request that occurs
 *                when the 16 bits CSIO transmit buffer register becomes empty.
 *-------------------------------------------------------------------------------
 * Argument     : unsigned short far *buf0  ; Top address of the buffer 0 defined by user
 *              : unsigned short far *buf0  ; Top address of the buffer 1 defined by user
 *              : unsigned short size       ; Size of buffer (Specify in WORD unit)
 *              : unsigned short zero_cnt   ; Set the number of outputting silence data
 *-------------------------------------------------------------------------------
 * Return value : None
 *-------------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *-------------------------------------------------------------------------------
 * Subroutines  : None
 *-------------------------------------------------------------------------------
 * Note         :
 *-------------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""***************************************************/
void Start_DMA1( unsigned short far *buf0,  /* Top address of the buffer 0 defined by user */
                 unsigned short far *buf1,  /* Top address of the buffer 1 defined by user */
                 unsigned short size,       /* Size of buffer (Specify in WORD unit) */
                 unsigned short zero_cnt )  /* Set the number of outputting silence data */
{
    unsigned short  dma_cnt;
    unsigned char tmp;

    dma_cnt   = size;   /* DMA transfer count (Transfer count per 16 bits) */

    /* DMA inhibit */
    asm("   stc     DMD0,   $$[FB]", tmp);
    asm("   and.b   #00Fh,  $$[FB]", tmp);  /* Holds setting of the DMA0 */
    asm("   ldc     $$[FB], DMD0"  , tmp);
```

```
    /* Select the DMA request cause at the DMA inhibit state */
    dm1sl = 0x9C;   /* DMAi request cause select register */
        /* b[4:0]; DMA request cause select     [11100; Intelligent I/O interrupt control register 10] */
        /* b[5];   Software DMA request bit     [0;     ignore] */
        /* b[6];   -                            [0;     (set to 0)] */
        /* b[7];   DMA request bit (DRQ)        [1;     (set to 1, See Note1)] */
        /* Note1:  Set DMA inhibit before changing the DMA request cause, Set DRQ bit to 1 simultaneously */

    /* Set the DMA transfer count */
    asm("   ldc $$[FB], DCT1", zero_cnt);   /* DMAi transfer count register */
    asm("   ldc $$[FB], DRC1", dma_cnt);    /* DMAi transfer count reload register */

    /* Set the DMA source address (When the transfer direction select bits is 1, this register is source address) */
    asm("   ldc $$[FB], DMA1", buf0);       /* DMAi memory address register */
    asm("   ldc $$[FB], DRA1", buf1);       /* DMAi memory address reload register */

    /* Set the DMA destination address */
    asm("   ldc #017Ch, DSA1");             /* DMAi SFR address register [Group 3 SI/O transmit buffer register] */

    /*
        Insert dummy cycles if needed
        (Adjust cycles depending on the number of DMA using in the application)
    */

    /* Set DMA mode and enable DMA transfer */
    asm("   or.b    #0F0h,  $$[FB]", tmp);
    asm("   ldc     $$[FB], DMD0"  , tmp);  /* DMA mode register 0 */
        /* b[1:0]; Channel 0 transfer mode select       [XX; Hold setting of DMA0] */
        /* b[2];   Channel 0 transfer unit select       [X;  Hold setting of DMA0] */
        /* b[3];   Channel 0 transfer direction select  [X;  Hold setting of DMA0] */
        /* b[4:5]; Channel 1 transfer mode select       [11; Repeat transfer] */
        /* b[6];   Channel 1 transfer unit select       [1;  16 bits] */
        /* b[7];   Channel 1 transfer direction select  [1;  Memory to fixed address] */

    /* Interrupt priority level */
    dm1ic = 0x07;
}
#else


/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Abstract     : Setting DMA0 and enable DMA0
 *-----------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-----------------------------------------------------------------------------
 * Declaration  : void Start_DMA0( unsigned short far *buf0,
 *              :                  unsigned short far *buf1,
 *              :                  unsigned short size,
 *              :                  unsigned short zero_cnt )
 *-----------------------------------------------------------------------------
 * Function     : Set DMA0 as DMAC when using 8 bits CSI0.
 *              : By calling this function, the DMA will be in a standby state
 *                which starts transmission by the interruption request that occurs
 *                when the 8 bits CSI0 transmit buffer register becomes empty.
 *-----------------------------------------------------------------------------
 * Argument     : unsigned short far *buf0  ; Top address of the buffer 0 defined by user
 *              : unsigned short far *buf0  ; Top address of the buffer 1 defined by user
 *              : unsigned short size       ; Size of buffer (Specify in WORD unit)
 *              : unsigned short zero_cnt   ; Set the number of outputting silence data
 *-----------------------------------------------------------------------------
```

```
 * Return value : None
 *--------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *--------------------------------------------------------------------------
 * Subroutines  : None
 *--------------------------------------------------------------------------
 * Note         :
 *--------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""*****************************************************/
void Start_DMA0( unsigned short far *buf0,  /* Top address of the buffer 0 defined by user */
                 unsigned short far *buf1,  /* Top address of the buffer 1 defined by user */
                 unsigned short size,       /* Size of buffer (Specify in WORD unit) */
                 unsigned short zero_cnt )  /* Set the number of outputting silence data */
{
    unsigned short  dma_cnt;
    unsigned char tmp;

    dma_cnt   = size * 2;   /* DMA transfer count (Transfer count per 16 bits) */

    /* DMA inhibit */
    asm("   stc     DMD0,   $$[FB]", tmp);
    asm("   and.b   #0F0h,  $$[FB]", tmp);  /* Holds setting of the DMA1 */
    asm("   ldc     $$[FB], DMD0"  , tmp);

    /* Select the DMA request cause at the DMA inhibit state */
    dm0sl = 0x94;   /* DMAi request cause select register */
        /* b[4:0]; DMA request cause select     [10100; UART3 transmit] */
        /* b[5];  Software DMA request bit      [0;     ignore] */
        /* b[6];  -                             [0;     (set to 0)] */
        /* b[7];  DMA request bit (DRQ)         [1;     (set to 1, See Note1] */
        /* Note1:  Set DMA inhibit before changing the DMA request cause, Set DRQ bit to 1 simultaneously */

    /* Set the DMA transfer count */
    asm("   ldc $$[FB], DCT0", zero_cnt);   /* DMAi transfer count register */
    asm("   ldc $$[FB], DRC0", dma_cnt);    /* DMAi transfer count reload register */

    /* Set the DMA source address (When the transfer direction select bits is 1, this register is source address) */
    asm("   ldc $$[FB], DMA0", buf0);       /* DMAi memory address register */
    asm("   ldc $$[FB], DRA0", buf1);       /* DMAi memory address reload register */

    /* Set the DMA destination address */
    asm("   ldc #032Ah, DSA0");             /* DMAi SFR address register [UART3 transmit buffer register] */

    /*
        Insert dummy cycles if needed
        (Adjust cycles depending on the number of DMA using in the application)
    */

    /* Set DMA mode and enable DMA transfer */
    asm("   or.b    #00Bh,  $$[FB]", tmp);
    asm("   ldc     $$[FB], DMD0"  , tmp);  /* DMA mode register 0 */
        /* b[1:0]; Channel 0 transfer mode select       [11; Repeat transfer] */
        /* b[2];   Channel 0 transfer unit select       [0;  8 bits] */
        /* b[3];   Channel 0 transfer direction select  [1;  Memory to fixed address] */
        /* b[4:5]; Channel 1 transfer mode select       [XX; Hold setting of DMA1] */
        /* b[6];   Channel 1 transfer unit select       [X;  Hold setting of DMA1] */
        /* b[7];   Channel 1 transfer direction select  [X;  Hold setting of DMA1] */

    /* Interrupt priority level */
    dm0ic = 0x07;
}
#endif
```

```
/*""FUNC COMMENT""*************************************************************
 * ID           :
 * Abstract     : Stop outputting digital audio data play
 *-----------------------------------------------------------------------------
 * Include      : "sfr32c83.h"
 *-----------------------------------------------------------------------------
 * Declaration  : void _dai_stop( void )
 *-----------------------------------------------------------------------------
 * Function     : Stop outputting digital audio data streaming.
 *              : Stop the SCLK and the LRCK
 *-----------------------------------------------------------------------------
 * Argument     : None
 *-----------------------------------------------------------------------------
 * Return value : None
 *-----------------------------------------------------------------------------
 * Inputs       : None
 * Outputs      : None
 *-----------------------------------------------------------------------------
 * Subroutines  : None
 *-----------------------------------------------------------------------------
 * Note         :
 *-----------------------------------------------------------------------------
 * History      :
 *""FUNC COMMENT END""*********************************************************/
void _dai_stop( void )
{
    unsigned char tmp;



#if defined (CSI016)
/******* 16bits CSI0 ****************************************************/
#define TIME_OUT (5000) /* approx 1 msec : M32C/83 @20MHz */
    unsigned short time_out;

    /* DMA1 inhibit */
    dm1ic = 0;                              /* DMA1 interrupt disable */
    asm("   stc     DMD0,   $$[FB]", tmp);  /* Store DMA mode register to tmp */
    asm("   and.b   #00Fh,  $$[FB]", tmp);  /* Holds setting of the DMA0 */
    asm("   or.b    #0C0h,  $$[FB]", tmp);
    asm("   ldc     $$[FB], DMD0"  , tmp);  /* DMA mode register 0 */
        /* b[1:0]; Channel 0 transfer mode select       [XX; Hold setting of DMA0] */
        /* b[2];   Channel 0 transfer unit select       [X;  Hold setting of DMA0] */
        /* b[3];   Channel 0 transfer direction select  [X;  Hold setting of DMA0] */
        /* b[4:5]; Channel 1 transfer mode select       [00; DMA1 inhibit] */
        /* b[6];   Channel 1 transfer unit select       [1;  16 bits] */
        /* b[7];   Channel 1 transfer direction select  [1;  Memory to fixed address] */

    /* Loops until the transmit buffer register of CSI0 becomes empty */
    time_out = TIME_OUT;
    while( ti_g3cr == 0 ){
        if( time_out -- == 0 ) break;
    }
    /* Loops until the transmit register of CSI0 becomes empty */
    time_out = TIME_OUT;
    while( txept_g3cr == 0 ){
        if( time_out -- == 0 ) break;
    }
```

```
    /* Disable CSI0 transmission */

    /* Group 3 SI/O communication control register */
    te_g3cr = 0;    /* Transmission disabled */

    /* Group 3 SI/O communication mode register */
    g3mr = 0;       /* Communication part is reset */

    /* II/O SIO3 interrupt request bit */
    sio3te = 0;     /* Disable the interrupt request from communication function of II/O group 3 */


#else
/******* 8bits CSI0 ****************************************************/
    /* DMA0 inhibit */
    dm0ic = 0;                              /* DMA0 interrupt disable */
    asm("   stc     DMD0,   $$[FB]", tmp);  /* Store DMA mode register to tmp */
    asm("   and.b   #0F0h,  $$[FB]", tmp);  /* Holds setting of the DMA1 */
    asm("   or.b    #008h,  $$[FB]", tmp);
    asm("   ldc     $$[FB], DMD0"  , tmp);  /* DMA mode register 0 */
        /* b[1:0]; Channel 0 transfer mode select       [00; DMA0 inhibit] */
        /* b[2];   Channel 0 transfer unit select       [0;  8 bits] */
        /* b[3];   Channel 0 transfer direction select  [1;  Memory to fixed address] */
        /* b[4:5]; Channel 1 transfer mode select       [XX; Hold setting of DMA1] */
        /* b[6];   Channel 1 transfer unit select       [X;  Hold setting of DMA1] */
        /* b[7];   Channel 1 transfer direction select  [X;  Hold setting of DMA1] */


    /* Loops until the transmit buffer register of CSI0 becomes empty */
    while( ti_u3c1 == 0 );
    /* Loops until the transmit register of CSI0 becomes empty */
    while( txept_u3c0 == 0 );


    /* Disable CSI0 transmission */
    u3c1    = 0x02;     /* UARTi transmit/receive control register 1 */
        /* b[0]; Transmit enable bit                    [0; Transmission disabled] */
        /* b[1]; Transmit buffer empty flag             [1; No data present in transmit buffer register (R only)] */
        /* b[2]; Receive enable bit                     [0; Reception disabled] */
        /* b[3]; Receive complete flag                  [0; Data present in receive buffer register (R only)] */
        /* b[4]; UARTi transmit interrupt cause select  [0; Transmit buffer empty] */
        /* b[5]; UARTi continuous receive mode enable bit [0; Continuous receive mode disabled] */
        /* b[6]; Data logic select bit                  [0; No reverse] */
        /* b[7]; Clock divide synchronizing stop bit    [0; Synchronizing stop] */
#endif
/***********************************************************************/


    /* Stop the SCLK and the LRCK */
    Stop_SCLK;
    Stop_LRCK;

}
```

A_endian.c

```
;""FILE COMMENT""****************************************************************
; System Name   : Digital Audio I/F Driver sample
; File Name     : A_endian.a30
; Version       : 0.01
; Contents      : Digital Audio I/F Driver subroutine
;                   Convert endian
;
; CPU           : M32C/80 series
; Compiler      :
; Assembler     : AS308(Version 3.10 Release2)
; Note          : 8bits CSIO needs this module
;*********************************************************************************
;   Copyright(C)2003, Renesas Technology Corp.
;   Copyright(C)2003, Renesas Solutions Corp.
;   All rights reserved.
;*********************************************************************************
; History       : 2002.10.01 Ver 0.01
;               :
;""FILE COMMENT END""************************************************************


.section    program, align
.glb    __cnv_endian
;""SUBR COMMENT""****************************************************************
; Module        : _cnv_endian
;                 void _cnv_endian ( unsigned char far *ram_buff,
;                                    unsigned short Bsize )
;                 #pragma PARAMETER _cnv_endian( A0, R1 )
;---------------------------------------------------------------------------------
; Abstract      : Converts endian
; Function      : Converts endian of 2 bytes (WORD) data
;---------------------------------------------------------------------------------
; Inputs        : *ram_bur [A0] ; The address to the buffer which stored data to convert.
;               : Bsize    [R1] ; The size of conversion bytes. (Set the even number to Bsize)
;
; Outputs       : *ram_buf  ; Converted data are stored to the buffer indicated by this pointer.
;
; Returns       : None
;---------------------------------------------------------------------------------
; Subroutines   : None
;---------------------------------------------------------------------------------
; Note          : (1) Specify the even number bytes to the Bsize.
;               : (2) Bsize can be specified from 2 to (FFFFh-1).
;               : (3) Do not set 0 to the Bsize.
; History       :
;""SUBR COMMENT END""************************************************************
__cnv_endian:
    pushm   A0, R1

    shl.w   #-2,    R1          ; cnt = cnt /4 (LONG)
    jnc     CONV_4BYTE          ; if(cannot divide by 4)
                                ;    Previously, 2bytes are converted
    mov.w   [A0],   R0
    mov.b   R0H,    [A0]        ; Store Hi -->  [A0]
    mov.b   R0L,    1[A0]       ; Store Lo --> 1[A0]
    add.L:Q #2,     A0          ;   ram_buff+=2
    cmp.w   #0,     R1          ; if( cnt == 0 ) goto END_CNV
    jeq     END_CNV
```

```
;=======================================================================
CONV_4BYTE:
    mov.w   [A0],   R0          ;
    mov.b   R0H,    [A0]        ; Store Hi -->  [A0]
    mov.b   R0L,    1[A0]       ; Store Lo --> 1[A0]


    mov.w   2[A0],  R0          ;
    mov.b   R0H,    2[A0]       ; Store Hi --> 2[A0]
    mov.b   R0L,    3[A0]       ; Store Lo --> 3[A0]


    add.L:Q #4,     A0          ;   ram_buff+=4
    adjnz.w #-1, R1, CONV_4BYTE ;   cnt--
                                ;   if ( cnt !=0 ) goto CONV_4BYTE
;=======================================================================
END_CNV:
    popm    A0, R1

RTS

.END
```

7. 参考ドキュメント

データシート
M32C/83 グループデータシート 暫定仕様書 REV.B3
（最新版をルネサス テクノロジホームページから入手してください。）

8. ホームページとサポート窓口

ルネサス テクノロジホームページ
http://www.renesas.com/

M16C ファミリ MCU 技術サポート窓口
E-mail：support_apl@renesas.com

改訂記録	内蔵周辺機能を利用したデジタルオーディオ I/F ドライバ アプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
1.10	2003.07.25	-	初版発行

安全設計に関するお願い

- 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

- 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
- 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ（http://www.renesas.com）などを通じて公開される情報に常にご注意ください。
- 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
- 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
- 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
- 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
- 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。