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。

H8/300L SLP シリーズ

SLP トーンジェネレータ (ToneGen)

要旨

H8/38024 SLP MCU を使用してトーンを生成するには以下の 2 つの方法があります。

1. パルス幅変調 (Pulse Width Modulation)
2. タイマトグル出力

動作確認デバイス

H8/38024

目次

1. 仕様

トーンジェネレータは，トーン信号を音楽的連続で定義して曲を生成する 1 つの方法です。ここに，2 種類の方法を示します。2 種類とも，同じ音楽的トーン（以下，トーンと略す）データと，2 つのトーンの間のリズムを使用します。(トーンデータの大きさを小さくするために，リズムは固定しています。)

1.1 トーン (音符)

長い空洞の管をたたくと，かなり一定な音 (音の高さ) を聞くことができます。これは，一定の速度の衝撃波が管に沿って振動するためです (周波数) 。「音符」は音の周波数で表すことができます。つまり，ピアノの鍵盤やギターの弦の音の高さです。慣例的に，音符は以下のように呼ばれます。

A，A#，B，C，C#，D，D#，E，F，F#，G，G#

接尾辞 「#」はシャープを表し「b」はフラットを表します。また， A# = Bb, C# = Db，D# = Eb，F# = Gb, G# = Ab です。ほとんどの音楽でこれらの名前は事実上の標準規格です。

ある音符の「オクターブ」とは，元の 2 倍の周波数です。たとえば，ある長さの空洞の管から発する音の周波数を 264 Hz とし，それを C と呼ぶことにします。長さが元の半分であれば，周波数は倍になります。これは，もとのＣより 1 オクターブ高い新たなＣです (264 x 2 = 528 Hz)。

表 1　音符，オクターブ，周波数

ヘルツ	オクターブ = 0	オクターブ = 1	オクターブ = 2	オクターブ = 3	オクターブ = 4	オクターブ = 5
A	55.000	110.000	220.000	440.000	880.000	1760.000
A#	58.270	116.541	233.082	466.164	932.328	1864.655
B	61.735	123.471	246.942	493.883	987.6\767	1975.533
C	65.406	130.813	261.626	523.251	1046.502	2093005
C#	69.296	138.591	277.183	554.365	1108.731	2217.461
D	73.416	146.832	293.655	587.330	1174.659	2349.318
D#	77.782	155.563	311.127	622.254	1244.508	2489.016
E	82.407	164.814	329.628	659.255	1318.510	2637.020
F	87.307	174.614	349.228	698.456	1396.913	2793.826
F#	92.499	184.997	369.994	739.989	1479.978	2959.955
G	97.999	195.998	391.995	783.991	1567.982	3135.963
G#	103.826	207.652	415.305	830.609	1661.219	3322.438
A	110.000	220.000	440.000	880.000	1760.000	3520.000

1.2 PWM モジュール

内蔵の 10 ビット PWM モジュールを使って，特定のデューティ比の連続した PWM パルスを生成します。また，ローパスフィルタに接続すると D/A 変換器としても使用できます。入力クロックとして 4 つのクロックソースがあります。10 ビットの分解能では，各変換期間に 4 種類の連続したパルスを取得できます。上記に示すように，レジスタのビットを設定することにより，4 種類の変換期間が設定できます。この PWM モジュールは，未使用時，省電力のためにスタンバイモードに独立して設定することができます。

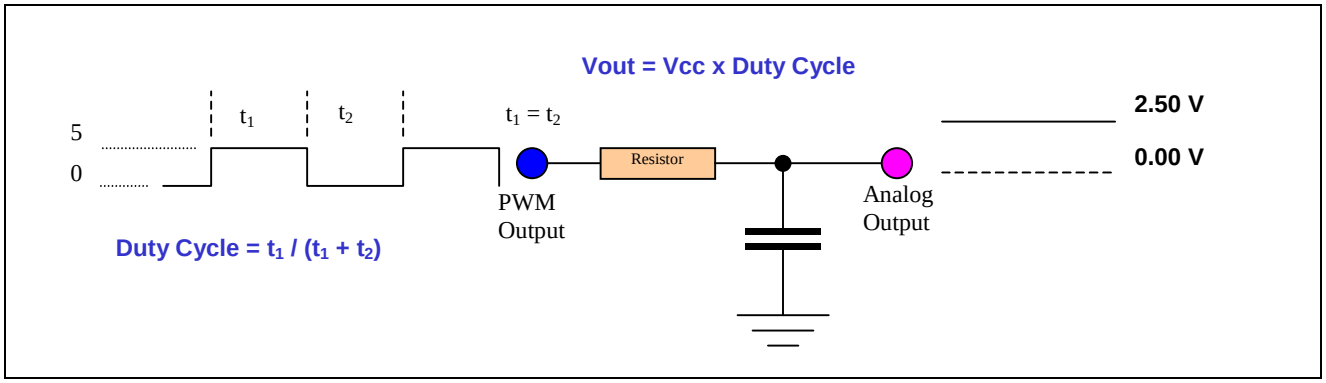


図1　PWM モジュールを D/A 変換器として使用するとき

10 ビット PWM モジュールの第一の目的は，外付けのローパスフィルタを使用して高分解能の D/A 変換器を提供することです。D/A 変換器の基本的な動作は 2 進数をアナログの電圧または電流に変換することです。昔からある D/A 変換器は，分解能の観点から CMOS 工程技術で実現するのは難しいため，出力したデューティ比をソフトウェアで変更できるカウンタを作成する方法があります。そのソフトウェアによる制御をパルス幅変調 (PWM) とよびます。

単純なローパスフィルタ (または DC 成分が不要のときは帯域フィルタ) を使用したとき，フィルタのアナログ出力は，基本的には，Vcc x デューティ比 (理想的には，出力は，周波数ではなくデューティ比の関数であることに注意) です。

例： Vout = 5.00 V x 50 % デューティ比 = 2.5 V

生成された DC 電圧レベルが正弦曲線の場合，正弦波が生成されます。

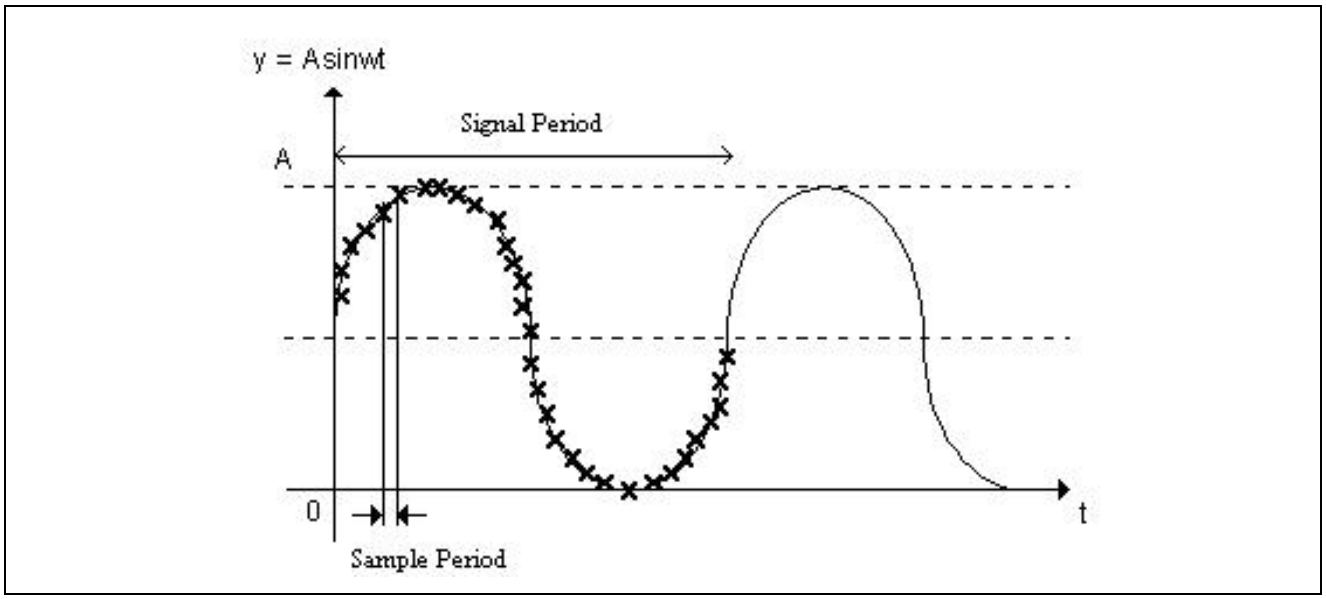


図2　正弦波の例

サンプリング期間は 2 つの PWM 値の期間と一致します。通常，タイマを使用して PWM モジュールに正弦波の値をリロードします。つまり，非同期イベントカウンタ (Asynchronous Event Counter) タイマを用います。

たとえば，使用する水晶発振子の周波数が 9.8304 MHz のとき

1 回 AEC 割り込みが発生する時間：$T_{interrupt}$

$T_{interrupt}$ = ((1/(∅/2)) x 256 **回**　　**注**： ∅ = ∅osc/2
= (1 / [(∅osc/2)/ 2]) x 256 count
= (1 / (9.8304MHz / 4) x 256 count
= 104.16μs

サンプリング周期は 1 回の AEC 割り込みの発生周期に等しくなります。割り込みサービスルーチン (Interrupt Service Routine) により，算出されたパルス幅が PWM 幅レジスタに格納されます。

サンプリング周波数　= $1/T_{interrupt}$
= 9600 Hz

パルス幅の計算にはインクリメントカウンタ値が必要です。インクリメントカウンタ値は以下のように求められます。

仮定条件：

- 完全な正弦波表に必要なサンプル数：256 個
- サンプリング周波数 = 9600 Hz
- 信号周波数 = 440 Hz (例：三つ目のオクターブの「A」の音符)

インクリメントカウンタ値　=　256/**インクリメント数**

インクリメント数は，サンプリング周波数と信号周波数に依存します。また，1 つの完全なサイクルのなかの正弦波表において特定の信号のインクリメント数に等しくなります。

インクリメント数　=　**サンプリング周波数/信号周波数**
∴　**インクリメントカウンタ値**=　256/(**サンプリング周波数/信号周波数**)
=　256 * **信号周波数/サンプリング周波数**
=　256 * (440 Hz)/(9600 Hz)
=　11.73

これらの計算はすべてコンパイラが行います。したがって，他のパラメータでこの計算を行なうときには，デフォルト値を変更してください。

1.3 タイマトグル出力の適用

トーンの生成をソフトウェアで行なうにはいくつかの方法があります。たとえば，トグル出力とアウトプットコンペア機能を備えたタイマ F を使用します。トグル出力の初期値を設定します。タイマ F のカウンタ値は，入力クロックパルスごとにインクリメントされます。タイマ F のカウンタ値は常にアウトプットコンペアレジスタ F に設定された値と比較されます。二つの値が一致すると，カウンタをクリアしたり，割り込み要求を出したり，出力をトグルしたりできます。タイマ F を二つの独立した 8 ビットタイマとして使用することもできます。

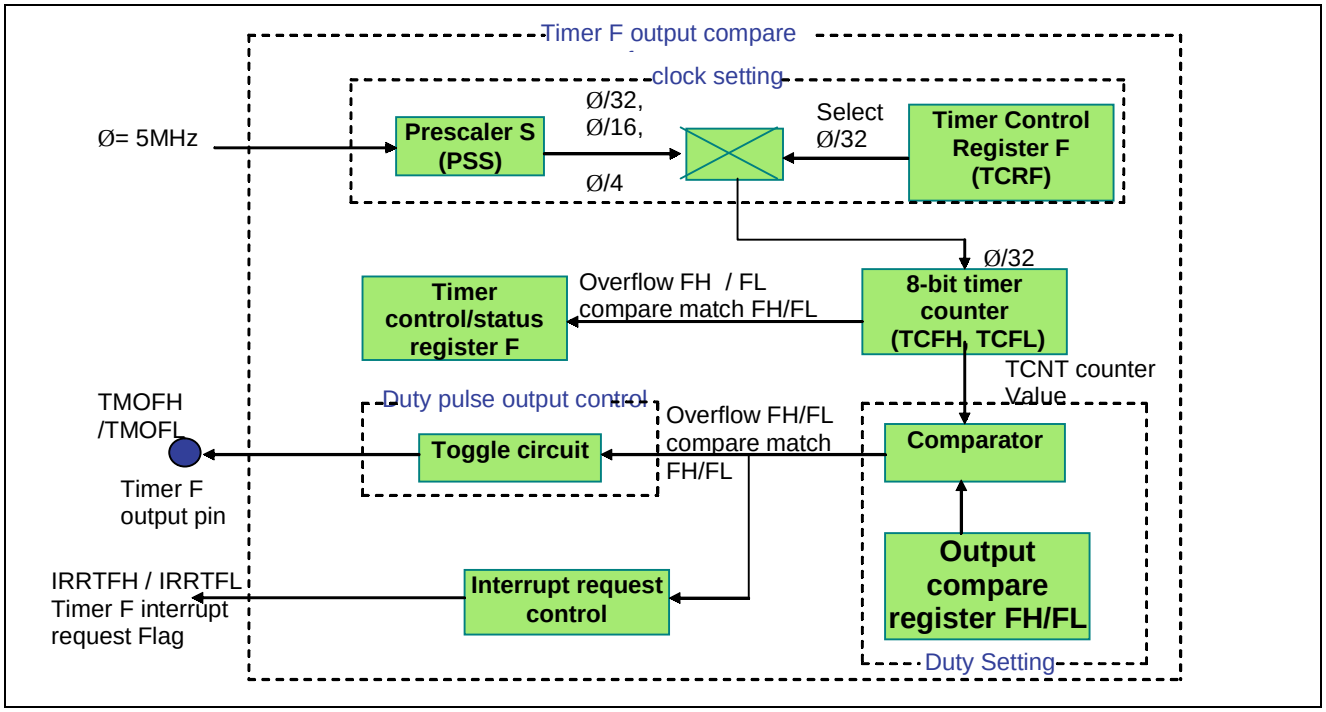


図3　タイマ F アウトプットコンペア動作のブロック図

図 3 に，タイマ F のアウトプットコンペア機能を使用して TMOFH/TMOFL ピンから PWM データを出力する方法を示します。

- 5 MHz のシステムクロックがプリスケーラ S に入力され，クロックを 32，16，および 4 で分周します。
- TCRF は 8 ビットの読み出し専用レジスタで，入力クロックの選択および TMOFL ピンからの出力レベルの設定を行ないます。
- タイマカウンタ FL と FH (TCFL/TCFH) は 8 ビットの読み出し/書き込み可能なアップカウンタです。この例では，入力クロックは Ø/32 です。
- タイマコントロール/ステータスレジスタ F (TCSRF) は，コンペアマッチで，TCFL のクリアをディセーブルにして，カウンタ FL オーバーフロー割り込みをイネーブルにします。
- アウトプットコンペアレジスタ FL (OCRFL) のデータは常にタイマカウンタ (TCFL) のデータと比較されます。
- 両方のレジスタの値が一致すると，コンペアマッチが生成されて TMOFL ピンがトグルされます。同時に，コンペアマッチフラグ L (CMFL) が 1 にセットされて CPU に対して割り込み要求が出されます。

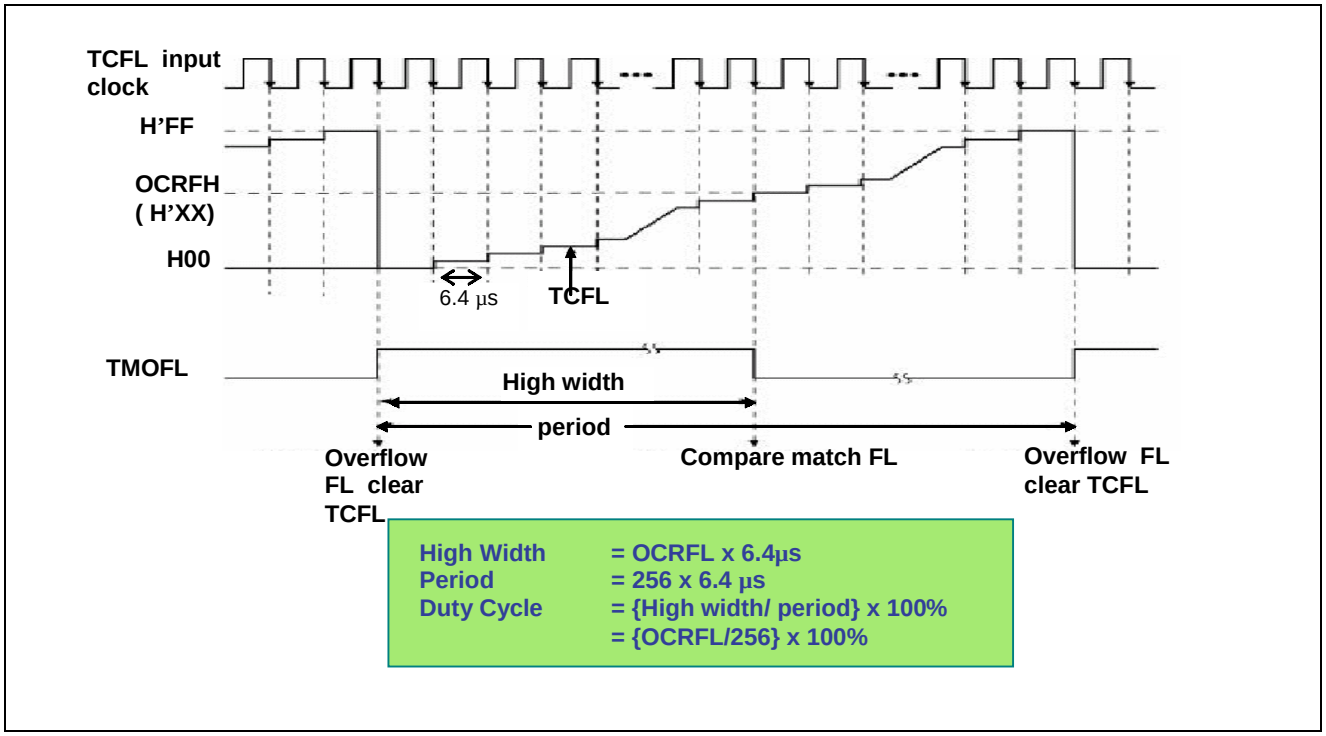


図4　タイマ F アウトプットコンペア動作

図 4 にタイマ F コンペアマッチ機能を使用して特定のデューティ比をもつパルス，つまりデジタルトーン信号を生成する方法を示します。タイマカウンタレジスタ FL (TCFL) では出力波形のトーン信号クロックサイクルまたは期間を決定します。一方，アウトプットコンペアレジスタ (OCRFL) に格納された値はデューティ比を決定します。任意のデューティ比の計算は上述の式によって求められます。タイマ F のプログラミングは 1 回だけです。出力のデューティ比を変更しない限り，OCRFL をリロードする必要はありません。

ユーザは，二つのタイマ F のトグル出力 (TMOFL と TMOFH) を組み合わせて，二つのデジタルトーン，たとえば，トレブル (高周波数) 用とバス (低周波数) 用を生成できます。図 5 にタイマトグル出力によるトーン生成のブロック図を示します。

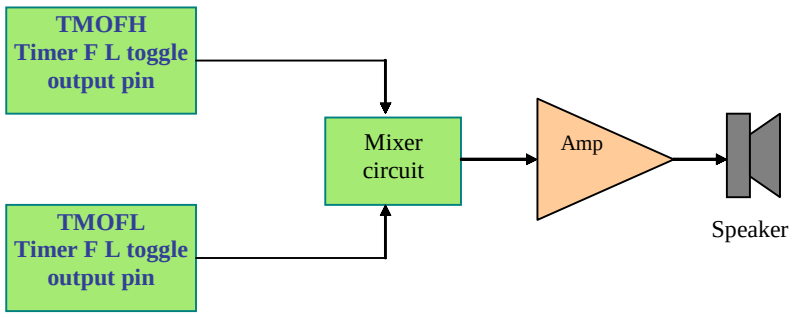


図5　タイマトグル出力によるトーン生成のブロック図

2. ハードウェアの適用

2.1 PWM モジュール

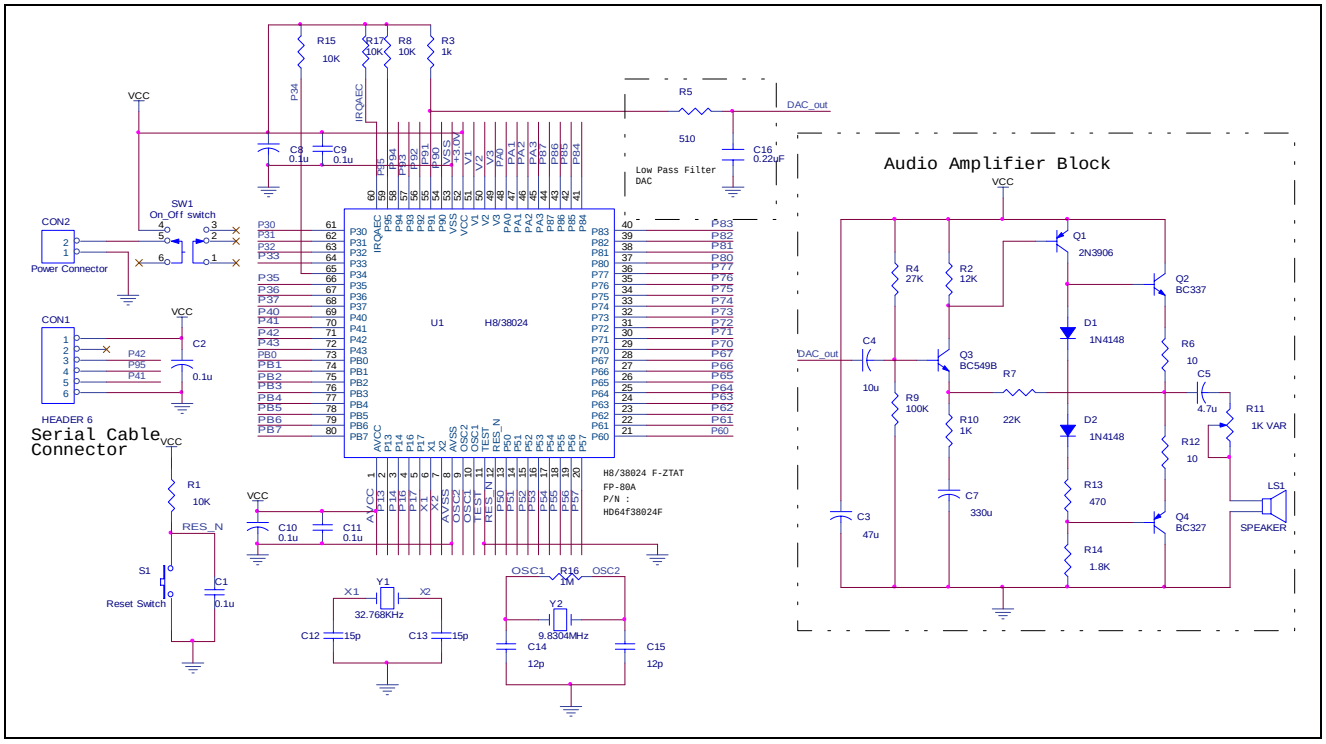
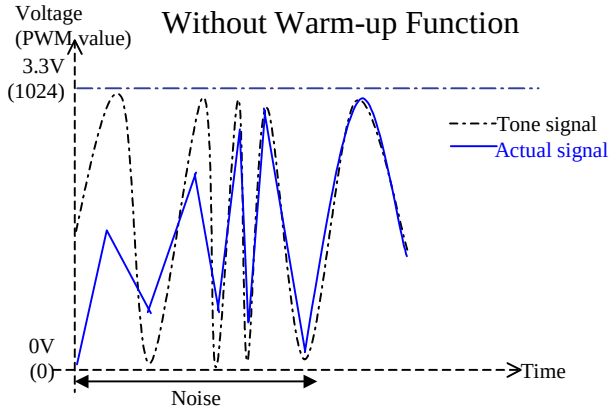
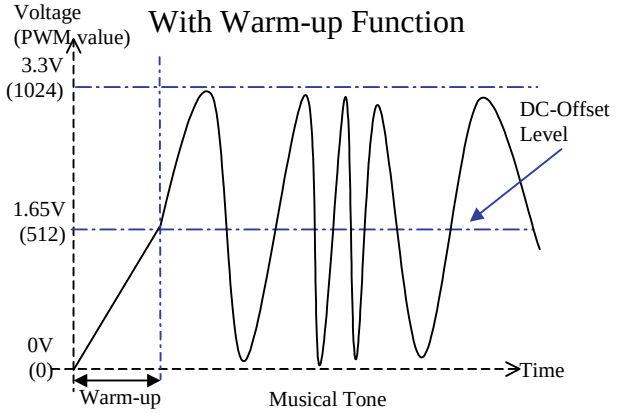


図6　PWM トーン生成の概要

SLP MCU の PWM モジュールでは，音を生成します。ソフトウェアにより，正弦波の信号を，固定した周期でパルス幅が変化する連続したパルスに変調します。パルスの幅の相違は正弦波の電圧レベルに対応します。PWM 出力端子にローパスフィルタ (Low Pass Filter) を接続することにより，PWM 信号を復調します。LPF は連続したパルスをアナログの正弦波信号に変換する積分器の役割をします。その後，トーンはオーディオアンプに送られ，音が出力されます。

ウォームアップ機能：通常，オーディオ信号は，0 V のグランドレベルに平均値があります。(電圧は，正と負の値の間を行き来します。) しかし，PWM モジュールを使用したこの応用例では，電圧レベルは正の値だけです。つまり，1/2Vcc レベルの DC オフセットが必要です。これを，オーディオアンプのウォーミングアップ機能と呼びます。これは電源投入時にのみ (コンデンサを充電するため) 必要で，早い段階でノイズ出力を抑えます。





2.2 タイマトグル出力の適用

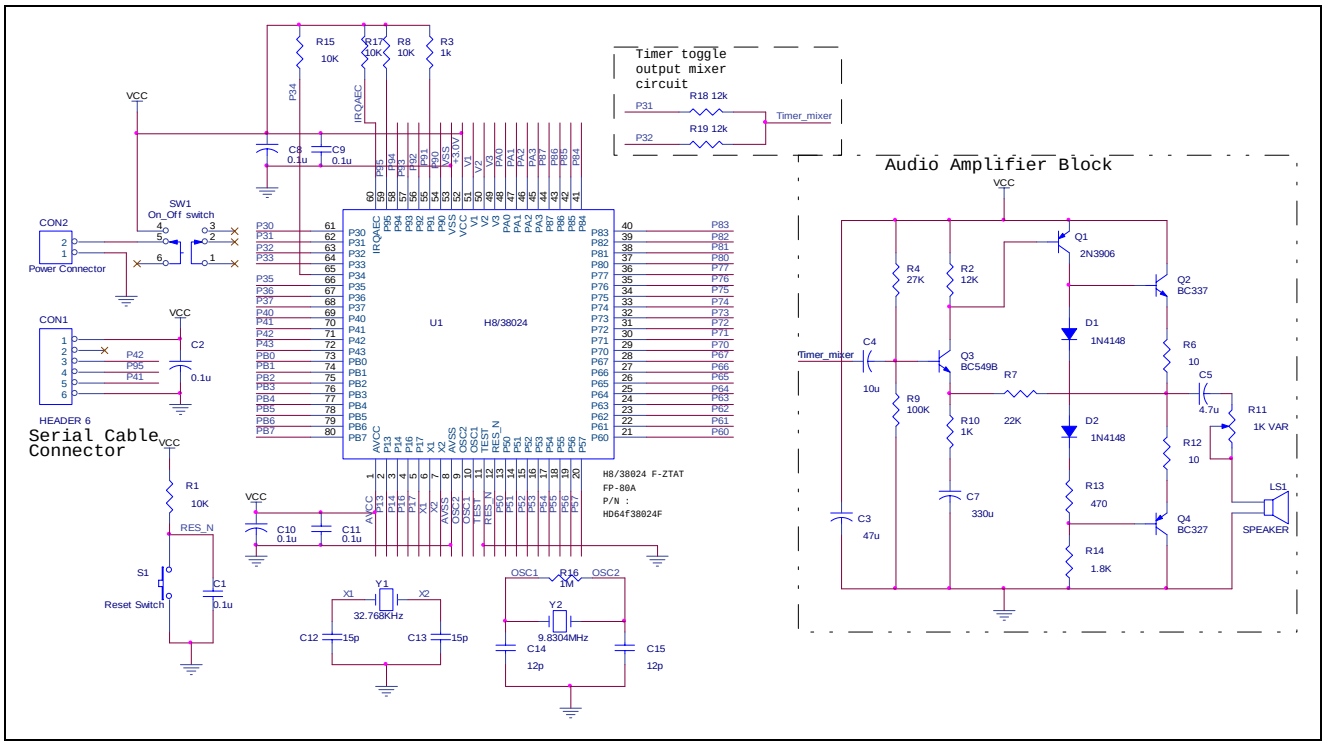


図7　タイマトグル出力によるトーン生成の概要

SLP MCU のタイマ F トグル出力によりデジタルトーンが生成されます。タイマ F のアウトプットコンペア値が新しい値にリロードされると，ソフトウェアによりパルス幅の異なる信号が生成されます。二つのタイマ F トグル出力 (ローカウンタとハイカウンタ) が組み合わされ，その結果，同時に 2 つのデジタルトーンが生成されます。2 つのデジタルトーンは，オーディオアンプに送信される前に，抵抗ミキサに送信されます。ユーザはスピーカからトーンを聞くことができます。

3. 動作と考察

ハードウェア回路により，フラッシュメモリへの書き込み機能が提供されます。ユーザは PC のシリアルポートからトーン生成のデモプログラムをダウンロードできます。このプログラムをダウンロードするのに必要な PC アプリケーションソフトウェアはフリーウェアです。このフラッシュ開発ツールキット (Flash Development Toolkit) は，以下の URL からダウンロードできます。

www.eu.renesas.com.

フラッシュ開発ツールキットのプログラムを正常にダウンロードした後，MCU をリセットして，プログラムを実行してください。実行中，スピーカから音を聞くことができるはずです。デモプログラムでは，同じ曲を繰り返し演奏します。

PWM トーン生成デモプログラムは，#define 文で XTAL の値を変更することによって，他の水晶発振子の値を使用できます。

例：

水晶発振子	= 9.8304 MHz のとき	➔	#define XTAL	9830400L	(デフォルト)
水晶発振子	= 4 MHz のとき	➔	#define XTAL	4000000L	

H8/38024F MCU には PWM モジュールが 2 チャネルあります。ソースプログラムをコンパイルする前に，使用する PWM モジュールを定義してください。

例：

PWM1 使用時	➔	#define PWM_use	1	(デフォルト)
PWM1 使用時	➔	#define PWM_use	2	

4. プログラムリスト

H8/38024F SLP MCU 用の HEW プロジェクトジェネレータを使用して添付プログラムを生成しました。無償の SLP/Tiny ツールチェインを使用できます。

4.1 PWM の適用

図 8 に PWM 適用時のフローチャートを示します。PWM_tone.c のソースプログラムリストを示します。

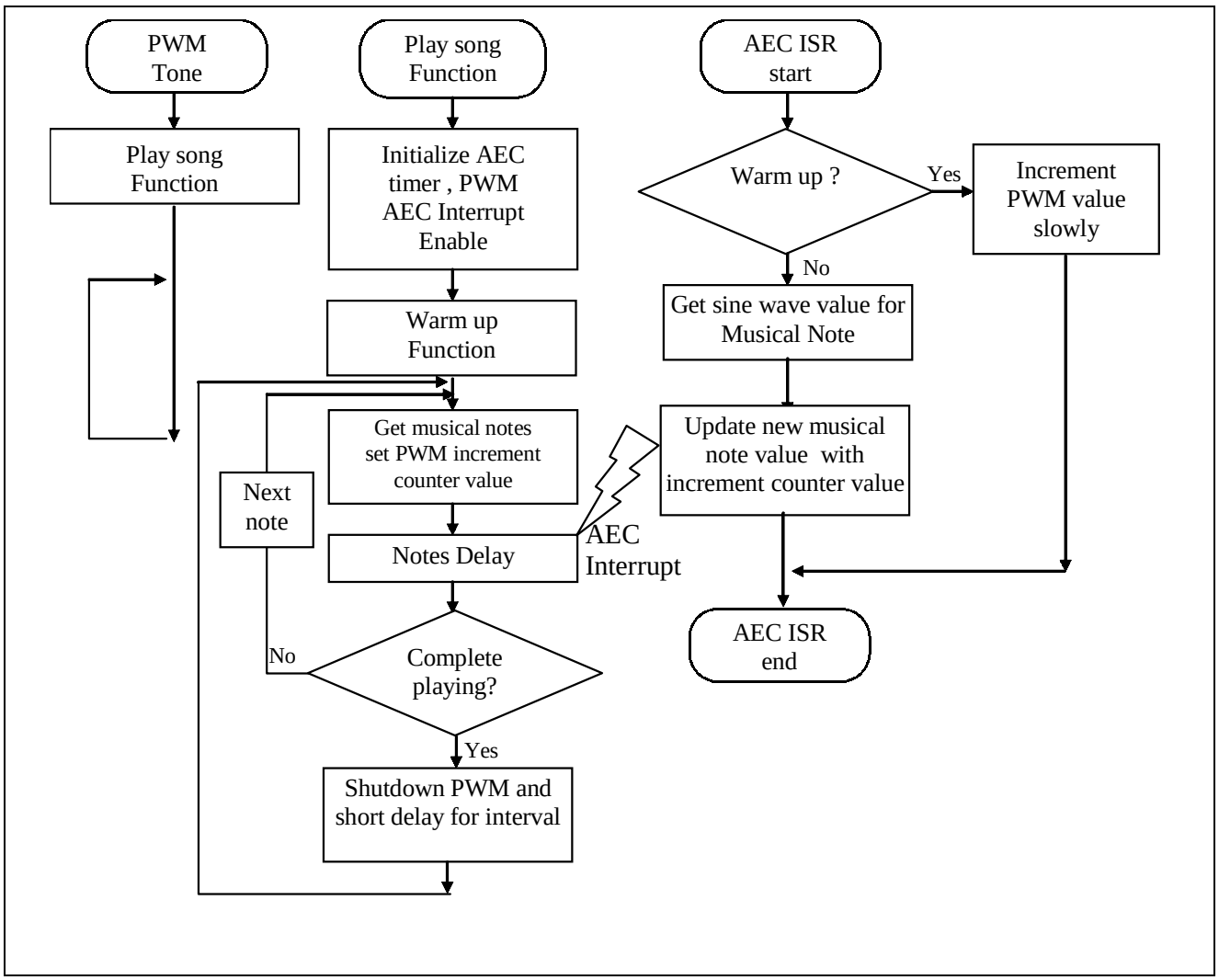


図8　PWM_Tone.c のフローチャート

```
/*****************************************************************/
/*                                                               */
/*  FILE        :PWM_Tone.c                                      */
/*  DATE        :Tue, Sep 09, 2003                               */
/*  DESCRIPTION :Main Program                                    */
/*  CPU TYPE    :H8/38024F                                       */
/*                                                               */
/*  This file is generated by Renesas Project Generator (Ver.2.1).*/
/*                                                               */
/*****************************************************************/

/************************************************************/
/*  File Include                                            */
/************************************************************/
#include    <machine.h>
#include    "iodefine.h"
#include    <math.h>
/************************************************************/
/* define                                                   */
/************************************************************/
#define  XTAL        9830400L
#define  sample_freq (XTAL/4L) / 256L //256 clock cycles per interrupt

#define  C1          ((256L * 523L)/100)/(sample_freq/100)
#define  C1S         ((256L * 554L)/100)/(sample_freq/100)
#define  D1          ((256L * 587L)/100)/(sample_freq/100)
#define  D1S         ((256L * 622L)/100)/(sample_freq/100)
#define  E1          ((256L * 659L)/100)/(sample_freq/100)
#define  F1          ((256L * 698L)/100)/(sample_freq/100)
#define  F1S         ((256L * 740L)/100)/(sample_freq/100)
#define  G1          ((256L * 784L)/100)/(sample_freq/100)
#define  G1S         ((256L * 830L)/100)/(sample_freq/100)
#define  A1          ((256L * 880L)/100)/(sample_freq/100)
#define  A1S         ((256L * 932L)/100)/(sample_freq/100)
#define  B1          ((256L * 987L)/100)/(sample_freq/100)

#define  C2          ((256L * 1046L)/100)/(sample_freq/100)
#define  C2S         ((256L * 1109L)/100)/(sample_freq/100)
#define  D2          ((256L * 1174L)/100)/(sample_freq/100)
#define  D2S         ((256L * 1244L)/100)/(sample_freq/100)
#define  E2          ((256L * 1318L)/100)/(sample_freq/100)
#define  F2          ((256L * 1396L)/100)/(sample_freq/100)
#define  F2S         ((256L * 1480L)/100)/(sample_freq/100)
#define  G2          ((256L * 1568L)/100)/(sample_freq/100)
#define  G2S         ((256L * 1661L)/100)/(sample_freq/100)
#define  A2          ((256L * 1760L)/100)/(sample_freq/100)
#define  A2S         ((256L * 1864L)/100)/(sample_freq/100)
#define  B2          ((256L * 1864L)/100)/(sample_freq/100)

#define  C3          ((256L * 2093L)/100)/(sample_freq/100)
#define  C3S         ((256L * 2217L)/100)/(sample_freq/100)
#define  D3          ((256L * 2349L)/100)/(sample_freq/100)
```

```
#define  PWM_use     2      //select "1" for PWM channel 2
                            //select "0" for PWM channel 1
/*************************************************************/
/*  Function define                                          */
/*************************************************************/

void init_PWM(unsigned char);
void storeCount(unsigned short);
void aecint( void );
void init_AEC(void);
void init_Tone(void);void off_DTMF(void);
void init_PWM1(unsigned char selClk1);
void init_PWM2(unsigned char selClk2);
void warm_up(void);
void play_song(void);

/*************************************************************/
/*Constant Look up Table for Sine Wave value
/*************************************************************/
const unsigned int song1[]=
{
B2, B2, B2, A2S, G2S, A2S,
F2S, C2S, C2, F2S, F2, F2S,
A2S, G2S, B2, B2, A2S, G2S,
A2S, F2S, A1S, A1S, D2S, D2,
D2S, F2S, F2, F2, F2, F2S,
F2,  C2S, F2, D2S, B1, C2S,
D2S, C2S, D2S, F2, F2S, F2,
F2S, F2S, G2S, A2S, A2S, G2S,
G2S, G2S, 0xFF
};


const unsigned int   Sine_Table[256]=
{
512,518,525,531,537,543,550,556,
562,568,574,580,586,592,598,604,
610,616,621,627,633,638,644,649,
654,659,664,669,674,679,684,688,
693,697,702,706,710,714,717,721,
725,728,731,734,737,740,743,746,
748,750,753,755,756,758,760,761,
762,763,764,765,766,766,766,767,
767,767,766,766,766,765,764,763,
762,760,759,757,755,754,751,749,
747,744,742,739,736,733,730,726,
723,719,715,712,708,704,699,695,
691,686,681,677,672,667,662,657,
652,646,641,635,630,624,619,613,
607,601,595,589,583,577,571,565,
559,553,546,540,534,528,521,515,
509,503,496,490,484,478,471,465,
```

```
459,453,447,441,435,429,423,417,
411,405,400,394,389,383,378,372,
367,362,357,352,347,343,338,333,
329,325,320,316,312,309,305,301,
298,294,291,288,285,282,280,277,
275,273,270,269,267,265,264,262,
261,260,259,258,258,257,257,257,
257,257,258,258,259,260,261,262,
263,264,266,268,269,271,274,276,
278,281,284,287,290,293,296,299,
303,307,310,314,318,322,327,331,
336,340,345,350,355,360,365,370,
375,380,386,391,397,403,408,414,
420,426,432,438,444,450,456,462,
468,474,481,487,493,499,506,512
};

/************************************************************/
/*Global variable
/************************************************************/
unsigned char PWDR_L2, PWDR_U2;
unsigned int i=0,j=0, count=0, inc1=0, inc2=0, final=0;
unsigned int lowcnt=0, hicnt=0;
unsigned char Ready = 0, DIGIT = 0;
unsigned int hold=0;

/************************************************************/
/*  Main Program                                            */
/************************************************************/
void main ( void )
{   play_song();
    while (1)
    {
      //Write user program here
    }
}

/************************************************************/
/*  Initialize Program                                      */
/************************************************************/
//Initialize tone generation function
void init_Tone(void)
{
    set_imask_ccr(1);                    // Interrupt Disable
   init_AEC();
   #if (PWM_use==1)
   init_PWM1(0); //Select conversion period = 512/(PWM input clock)
   #else
   init_PWM2(0); //Select conversion period = 512/(PWM input clock)
   #endif
}

void init_PWM1(unsigned char selClk1)
```

```
{
   if (selClk1 <= 3)        // Check if valid, otherwise PWM2 is off
   {
      P_IO.PMR9.BIT.PWM1 = 1;      // Configure P91 as PWM2 output pin
      P_PWM1.PWCR1.BYTE = selClk1; // Clock select for PWM2,write only
   }
}

void init_PWM2(unsigned char selClk2)
{
   if (selClk2 <= 3)              // Check if valid, otherwise PWM2 is off
   {
      P_IO.PMR9.BIT.PWM2 = 1;      // Configure P91 as PWM2 output pin
      P_PWM2.PWCR2.BYTE = selClk2; // Clock select for PWM2,write only
   }
}

void off_DTMF(void)
{
   P_SYSCR.IENR2.BIT.IENEC = 0;
                      // AEC Interrupt Request, 1-Enable, 0-Disable
   //compiler directive to select which code to be compile
   #if (PWM_use==1)
   P_IO.PMR9.BIT.PWM1 = 0;         // Turn off PWM1
   #else
   P_IO.PMR9.BIT.PWM2 = 0;         // Turn off PWM2
   #endif
}

/*************************************************************/
/*  Initialize Program                                       */
/*************************************************************/
void warm_up(void)
{
   set_imask_ccr(0);               // Interrupts, 0-Enable, 1-Disable
   while(count<0x3000) ;
   set_imask_ccr(1);               // Interrupts, 0-Enable, 1-Disable
   Ready = 1;
}

/*************************************************************/
/*  play_song Program                                        */
/*************************************************************/
void play_song(void)
{
   i=0;

   init_Tone();

   warm_up();
   while(1)
   {
      while (song1[i]!=0xFFFF)
```

```
      {  i++;
         inc1 = song1[i++];
         set_imask_ccr(0);         // Interrupts, 0-Enable, 1-Disable
         for (j=0; j<0x35000; j++) ;
      }

      storeCount(512);
      for (j=0; j<10000; j++) ;          // short delay Tone
      set_imask_ccr(1);                  // Interrupts, 0-Enable, 1-Disable
      i = 0;
   }

   off_DTMF();

}


/*************************************************************/
/*  Write each digital code into PWDR registers              */
/*************************************************************/
void storeCount(unsigned short PWDRval_2)
{
   //compiler directive to select which code to be compile
   #if (PWM_use==1)
   P_PWM1.PWDRL1.BYTE = (unsigned char)(PWDRval_2 & 0x00FF);
                                  // Write lower 8bits of 10bits data
   P_PWM1.PWDRU1.BYTE = (unsigned char) ((PWDRval_2 & 0x0300) >> 8);
                                  // Write upper 8bits of 10bits data
   #else
   P_PWM2.PWDRL2.BYTE = (unsigned char)(PWDRval_2 & 0x00FF);
                                  // Write lower 8bits of 10bits data
   P_PWM2.PWDRU2.BYTE = (unsigned char) ((PWDRval_2 & 0x0300) >> 8);
                                  // Write upper 8bits of 10bits data
   #endif
}


/*************************************************************/
/*  AEC Interrupt Service Routine                            */
/*************************************************************/
void aecint (void)
{
   P_SYSCR.IRR2.BIT.IRREC = 0;     // Clear IRREC flag

   if(P_AEC.ECCSR.BIT.OVL == 1)    // Check for ECL overflow flag
   {  P_AEC.ECCSR.BIT.OVL = 0;     // Clears flag

      if(Ready == 0)
      {
         storeCount(count++/128);
      }
      else
      {  final = (Sine_Table[lowcnt]);
```

```
          storeCount(final);
          lowcnt = lowcnt + inc1;
          if(lowcnt>255) lowcnt = lowcnt-255;
                              // If reached end of 1 period, then reset
          hicnt = hicnt + inc2;
            if(hicnt>255) hicnt = hicnt-255;
                              // If reached end of 1 period, then reset
       }
    }
 }

 void init_AEC(void)
 {
    P_AEC.ECCSR.BYTE = 0x15;
    P_AEC.ECCR.BYTE = 0x10;
    P_SYSCR.IRR2.BIT.IRREC = 0;           // Clear IRREC flag
    P_SYSCR.IENR2.BIT.IENEC = 1;          // AEC Interrupt Request, 1-Enable, 0-
 Disable
 }
```

割り込みサービスプログラム intprg.c のプログラムリストを以下に示します。以下のコードを挿入してください。

```
 extern void aecint (void);               //insert AEC ISR function
 .
 .
 .
 .
 .
 .
 __interrupt(vect=12) void INT_Counter(void)
 {
    aecint();                             //insert AEC ISR function
 }
```

4.2 タイマトグル出力の適用

図 9 にタイマトグル出力のフローチャートを示します。timer_tone.c のプログラムリストを示します。

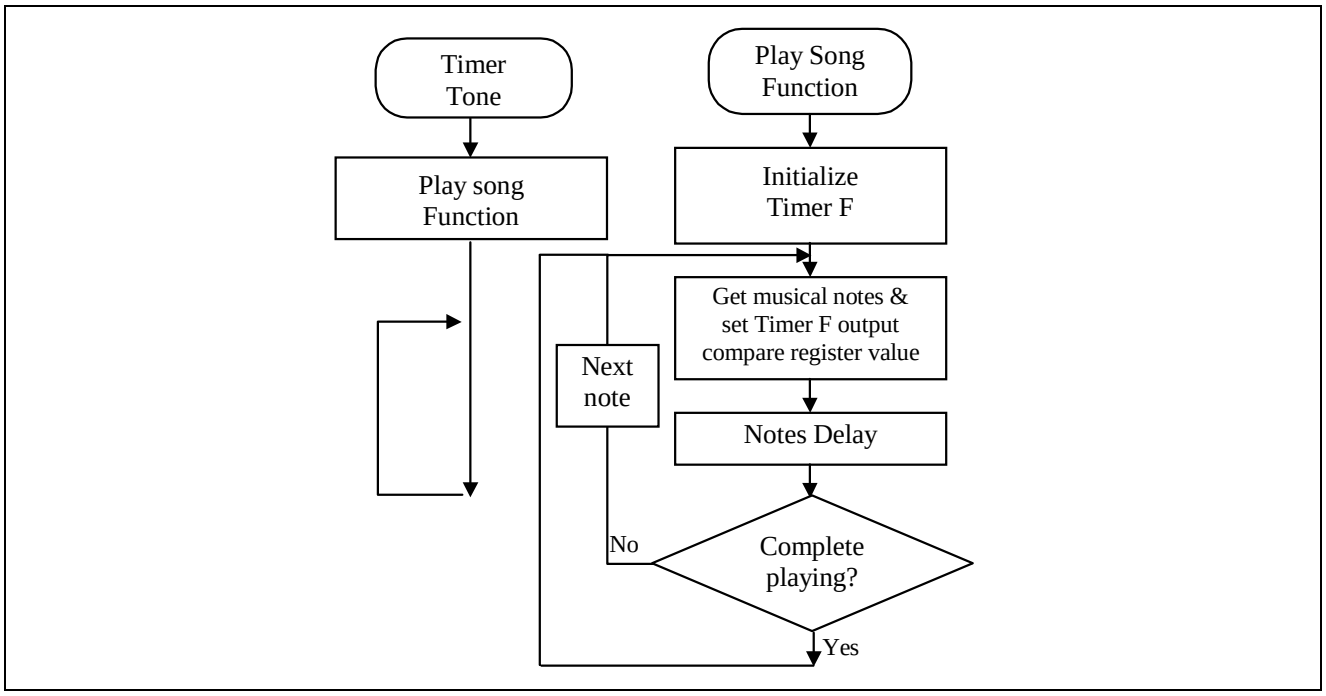


図9　timer_tone.c のフローチャート

```
/*********************************************************************/
/*                                                                   */
/*  FILE        :Timer_tone.c                                        */
/*  DATE        :Fri, Sep 12, 2003                                   */
/*  DESCRIPTION :Main Program                                        */
/*  CPU TYPE    :H8/38024F                                           */
/*                                                                   */
/*  This file is generated by Renesas Project Generator (Ver.2.1).   */
/*                                                                   */
/*********************************************************************/


/************************************************************/
/*  File Include                                            */
/************************************************************/
#include    <machine.h>
#include    "iodefine.h"
/************************************************************/
/* define                                                   */
/************************************************************/
#define  XTAL        9830400L
#define  Timer_clk   32L // main clock / 32

#define  C1          (XTAL / (Timer_clk*4L*523L))
#define  C1S         (XTAL / (Timer_clk*4L*554L))
#define  D1          (XTAL / (Timer_clk*4L*587L))
#define  D1S         (XTAL / (Timer_clk*4L*622L))
#define  E1          (XTAL / (Timer_clk*4L*659L))
#define  F1          (XTAL / (Timer_clk*4L*698L))
#define  F1S         (XTAL / (Timer_clk*4L*740L))
#define  G1          (XTAL / (Timer_clk*4L*784L))
#define  G1S         (XTAL / (Timer_clk*4L*830L))
#define  A1          (XTAL / (Timer_clk*4L*880L))
#define  A1S         (XTAL / (Timer_clk*4L*932L))
#define  B1          (XTAL / (Timer_clk*4L*987L))

#define  C2          (XTAL / (Timer_clk*4L*1046L))
#define  C2S         (XTAL / (Timer_clk*4L*1109L))
#define  D2          (XTAL / (Timer_clk*4L*1174L))
#define  D2S         (XTAL / (Timer_clk*4L*1244L))
#define  E2          (XTAL / (Timer_clk*4L*1318L))
#define  F2          (XTAL / (Timer_clk*4L*1396L))
#define  F2S         (XTAL / (Timer_clk*4L*1480L))
#define  G2          (XTAL / (Timer_clk*4L*1568L))
#define  G2S         (XTAL / (Timer_clk*4L*1661L))
#define  A2          (XTAL / (Timer_clk*4L*1760L))
#define  A2S         (XTAL / (Timer_clk*4L*1864L))
#define  B2          (XTAL / (Timer_clk*4L*1975L))

#define  C3          (XTAL / Timer_clk*4L)/(2093L)
#define  C3S         (XTAL / Timer_clk*4L)/(2217L)
#define  D3          (XTAL / Timer_clk*4L)/(2349L)
```

```
/*********************************************************/
/*  Function define                                      */
/*********************************************************/

void init_Tone(void);
void play_song(void);

/*********************************************************/
/*Constant Look up Table for Sine Wave value
/*********************************************************/
const unsigned char song1[]=
{
B2,  B2,  B2, A2S, G2S, A2S,
F2S, C2S, C2, F2S, F2, F2S,
A2S, G2S, B2, B2, A2S, G2S,
A2S, F2S, A1S, A1S, D2S, D2,
D2S, F2S, F2, F2, F2, F2S,
F2,  C2S, F2, D2S, B1, C2S,
D2S, C2S, D2S, F2, F2S, F2,
F2S, F2S, G2S, A2S, A2S, G2S,
G2S, G2S, 0xFF
};

/*********************************************************/
/*Global variable
/*********************************************************/
unsigned int i=0,j=0, count=0;

/*********************************************************/
/*  Main Program                                         */
/*********************************************************/
void main (void)
{  play_song();
    while (1)
   {
      //Write user program here
    }
}

/*********************************************************/
/*  Initialize Program                                   */
/*********************************************************/
//Initialize tone generation function
void init_Tone(void)
{
    set_imask_ccr(1);                    // Interrupt Disable

   //Init Timer F start

   // 8 bit timer F counter, Sub clock / 4 selected toggle output enable
   P_IO.PMR3.BYTE = 0x06;
   P_TMRF.TCRF.BYTE = 0xCE;
   P_TMRF.TCSRF.BYTE = 0x11;
```

```
    //TCF cleared when TCF and OCRF match
    if (P_TMRF.TCSRF.BIT.CMFH == 1) P_TMRF.TCSRF.BIT.CMFH = 0;
    if (P_TMRF.TCSRF.BIT.CMFL == 1) P_TMRF.TCSRF.BIT.CMFL = 0;

     set_imask_ccr(0);                      // Interrupt Enable

    //Init Timer F end
}


/************************************************************/
/*  play_song Program                                       */
/************************************************************/
void play_song(void)
{
    unsigned int i=0, j=0;

    init_Tone();
    while(1)
    {
       while (song1[i]!=0xFF)
       {
          P_TMRF.OCRF.BYTE.H = song1[i];
          P_TMRF.OCRF.BYTE.L = song1[i];
          i++;
          for (j=0; j<35000; j++) ;
       }
       for (j=0; j<35000; j++) ;
       i=0;
    }
    P_TMRF.TCRF.BYTE = 0x00;
}
```

5. 参考文献

1. PWM Sine Wave Generation, (Application Note ref. no: AN0303003, http://sg.renesas.com,)
2. Use PWM as A DAC, (Application Note ref. no: AN0303004, http://sg.renesas.com,)

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	04.08.06	—	初版発行

安全設計に関するお願い

1. 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

1. 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。
2. 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。
3. 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。
4. 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。
5. 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。
6. 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。
7. 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。
8. 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。