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 Super Low Power (SLP)シリーズ
ステッピングモータドライブ例

内容

本アプリケーションノートでは、H8/38024F SLP MCU を用いて 2 相/4 相ステッピングモータをドライブする例を説明します。2 種類のハードウェアドライバ回路を紹介します。

はじめに

ステッピングモータは、電流パルスをモータの回転に変換します。通常、ステッピングモータはコイルで構成されています。このコイルに電圧をかけると、1 ステップ毎にモータが回転します。

通常の動作では、2 つのコイルが同時に作動します。ステッピングモータは、作動する巻線を変化させるごとに、1 ステップずつ時計回りに回転します。逆の順序で作動させれば、モータは反時計回りに回転します。

回転速度はパルスの周波数で制御します。ステッピングモータに 1 パルス入力するたびに、モータは所定の角度を回転します。代表的な 1 ステップの回転角度は 18 度です。1 ステップで 18 度ずつ回転すると、モータが 1 回転する（360 度）には、20 ステップ必要です。時間の間隔を変えることでモータの速度を調整でき、ステップ数をカウントすることで回転角度を制御できます。

H8/38024 MCU の I/O ポート 5（ビット 3～0）を用いて 2 相/4 相ステッピングモータドライバ回路を駆動します。ここでは、2 種類のステッピングモータドライバ回路を紹介します。最初に紹介するのはステッピングモータドライバ IC（L298N）で、次はパワーMOSFET トランジスタ（2SK1095）です。

動作確認デバイス

H8/38024 シリーズ

目次

1. ハードウェアの概要

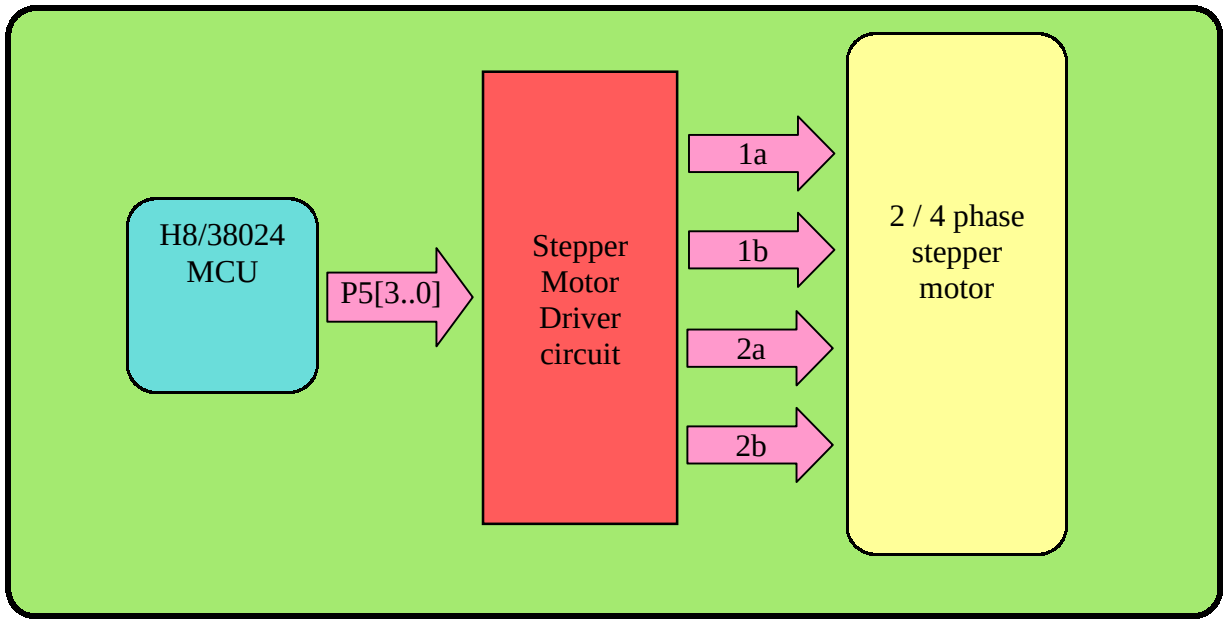

図 1　ハードウェアブロック図

基本的に、ステッピングモータ制御回路は 3 つのコンポーネントから成ります。

1. マイクロコンピュータ　　— ステッピングモータ制御波形を生成します。
2. モータドライバ　　— ステッピングモータを駆動します。
3. ステッピングモータ　　— MCU からの制御波形を評価します。

1.1 ハードウェア回路例 1

ハードウェア設計で最も複雑な部分はステッピングモータドライバ回路の設計です。このモータドライバの設計は、ステッピングモータの特性に依存します。ステッピングモータドライバの設計で最も一般的なのは、“ステッピングモータドライバ IC”を用いる方法です。この例ではステッピングモータとして 2 相または 4 相を接続できる SAIA-UAG2 を使用します。表 1 と表 2 に、本アプリケーションノートの例で使用できる 2 種類のステッピングモータの技術データを示します。

L298N ステッピングモータドライバの回路図（2 相）を図 2 に示します。R1 と R2 の値は、負荷電流に依存します。IC を L298N から L293D に変えると、保護ダイオードを削減できます。

表 1　SAIA-UAG2 ステッピングモータの技術データ

項目	値		
分解能あたりのステップ数	20		
巻線タイプ	ユニポーラ		
定格電圧	6V	12V	24V
巻線抵抗	35Ω	170Ω	700Ω
最大消費電力		0.4W	
巻線温度		130°C	
デューティサイクル		100%	
ホールディングトルク		0.5cNm	
ディテントトルク		0.14cNm	
回転方向		逆転可	
回転子イナーシャ		0.31gcm^2	

表 2　Step-Syn（型番 103H546-0440）ステッピングモータの技術データ

項目	値
分解能あたりのステップ数	200
定格電圧	3.15V
巻線抵抗	3.15Ω
相あたりの電流	1A
ホールディングトルク	1.5kg.cm
回転方向	逆転可
回転子イナーシャ	31gcm^2
重量	0.2kg

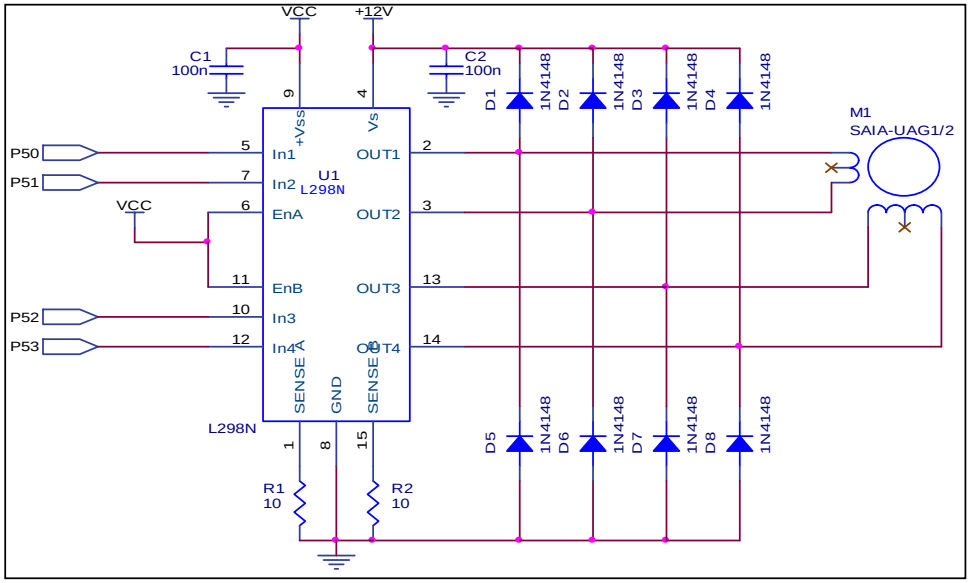

図 2　L298N ステッピングモータドライバ回路（2 相）

表 3　フルステップシーケンス

ステップ	P53	P52	P51	P50	PDR5
0	1	0	0	1	0x09
1	1	0	1	0	0x0A
2	0	1	1	0	0x06
3	0	1	0	1	0x05

表 4　ハーフステップシーケンス

ステップ	P53	P52	P51	P50	PDR5
0	1	0	0	0	0x08
1	1	0	1	0	0x0A
2	0	0	1	0	0x02
3	0	1	1	0	0x06
4	0	1	0	0	0x04
5	0	1	0	1	0x05
6	0	0	0	1	0x01
7	1	0	0	1	0x09

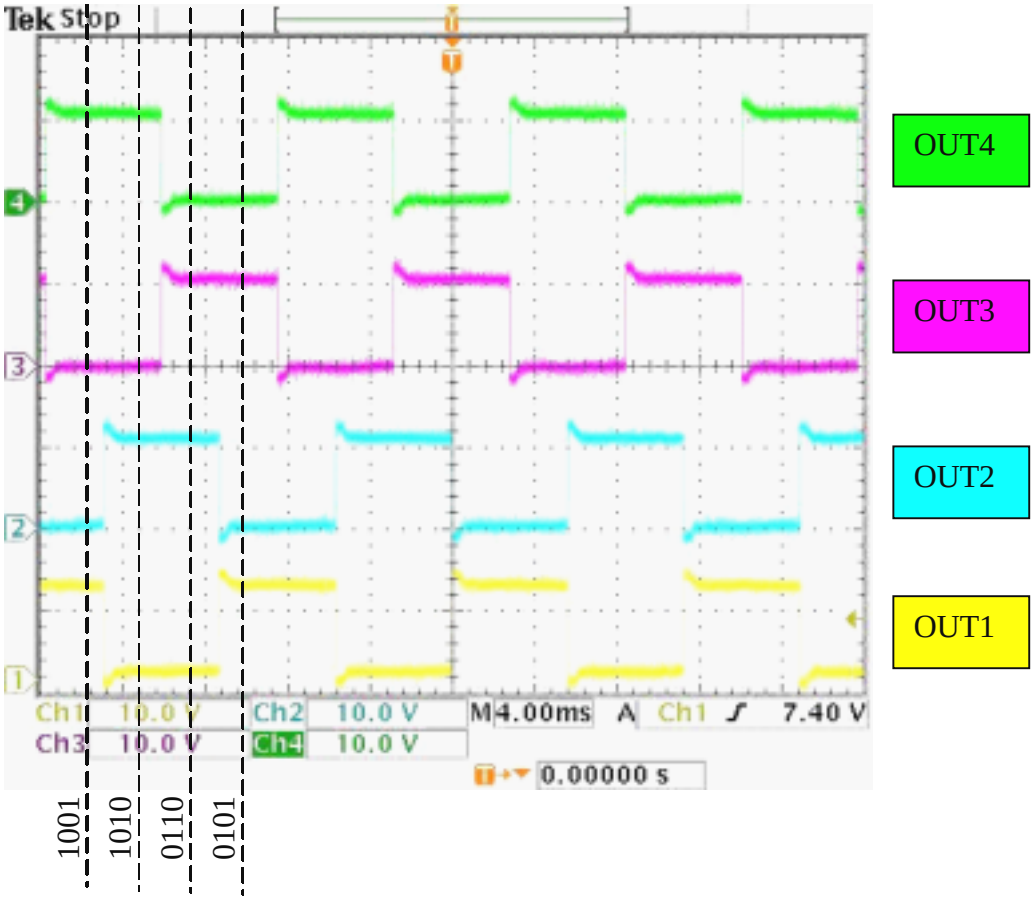

図 3　ハードウェア回路例 1 とプログラム例 1 による波形

1.2 ハードウェア回路例 2

ステッピングモータドライバ IC を使う方法のほかに、N チャネル MOSFET 2SK1095 を 4 つ用いてステッピングモータを駆動する方法もあります。図 4 に MOSFET を用いたステッピングモータドライバ回路（4 相）を示します。ステッピングモータの制御波形は、AND ゲートを通して入力します。

信号は 4 つの N チャネル MOSFET トランジスタゲート（型番：2SK1095）に入力し、それぞれを順にオン／オフしていきます。このトランジスタは、ステッピングモータのコイルで発生する電流のドレインとなります。保護ダイオード（型番：HRP-22）を 4 つ用いて、ドレイン－ソース間の電圧降下を 0.55V に保ちます。モータでは電気的なノイズが多く発生するので、C1 と C2 でノイズを抑えます。

最後に、+12V*電源から、コイル電流を制限する 2 つの抵抗を経由してモータ巻線に電力が供給されます。MOSFET のスイッチ時間は非常に速いため、フェーズ信号を反転する（A と A-、B と B-）間にデッドタイムを入れて、A と B がオンになる前に A-と B-が少し弱まるようにする必要があります。ハーフステップシーケンスを実現する場合も、このデッドタイムによる遅延を増加させることができます。MCU がスタンバイモードに入ると I/O ポートがハイインピーダンス状態になり、R6、R7、R8、R9 がプルダウンされているので、全フェーズが非動作状態になることに注意してください。

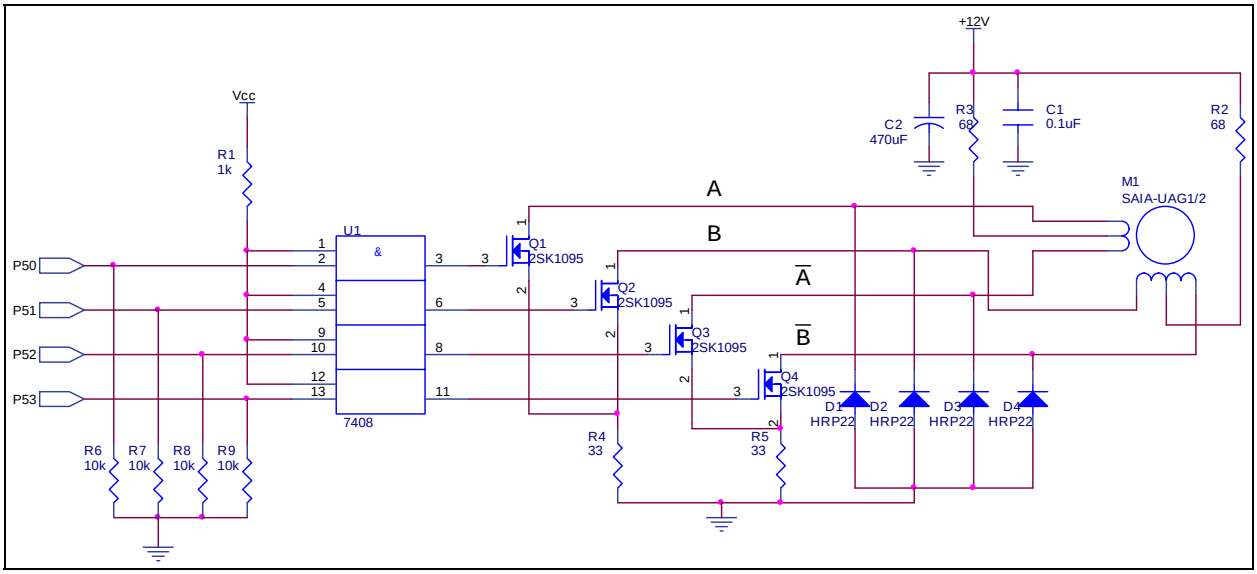

図 4　MOSFET ステッピングモータドライバ回路

【注】*使用するステッピングモータにあわせて+12V **電源および** R2, R3, R4, R5 **の値を変更できます。例えば、***Step-syn* **ステッピングモータで定格ホールディングトルクを実現するには、電源** = 3V**、**R2=R3=R4=R5=1Ω**とします。**

表 5　フルステップシーケンス

ステップ	P53	P52	P51	P50	PDR5
0	1	0	0	1	0x09
1	1	0	1	0	0x0A
2	0	1	1	0	0x06
3	0	1	0	1	0x05

表 6　デッドタイムシーケンス

ステップ	P53	P52	P51	P50	PDR5
0	0	0	0	1	0x01
1	1	0	0	0	0x08
2	0	1	0	0	0x02
3	0	0	1	0	0x04

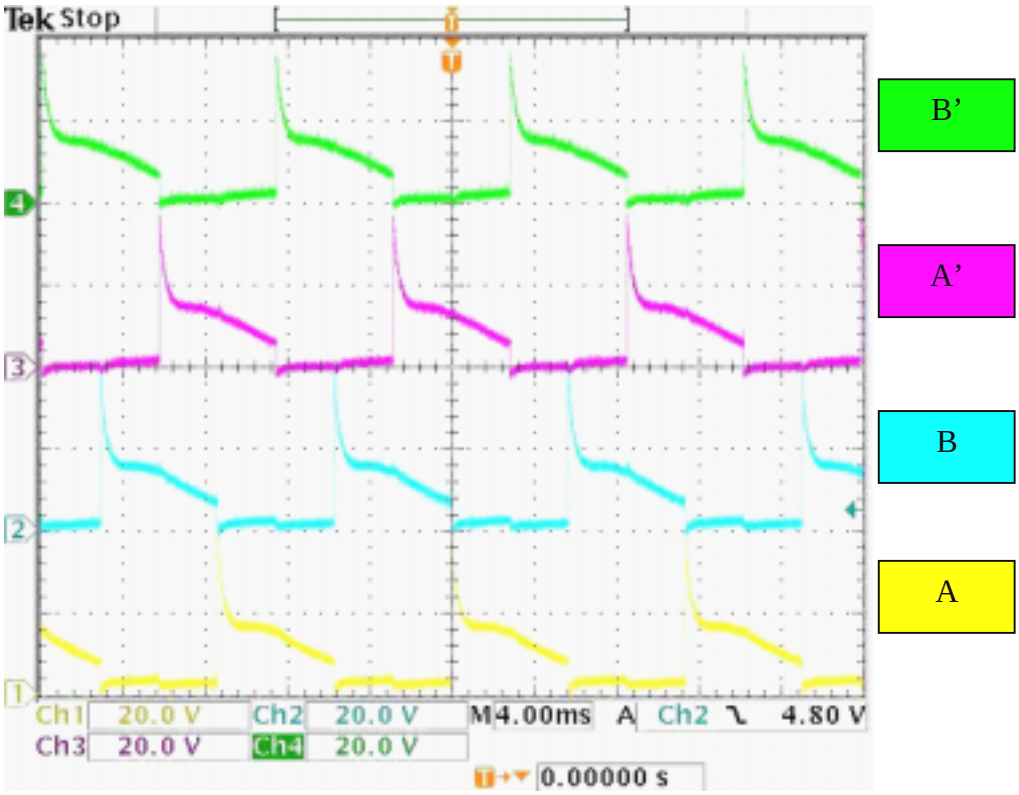

図 5　ハードウェア回路例 2 とプログラム例 2 による波形

2. ソフトウェアの概要

MCU の所定のポート端子をステッピングモータドライバ IC の入力端子に接続します。ステッピングシーケンスはソフトウェアで簡単に実現できます。ドライバ IC へ信号を入力するには、MCU のどの I/O ポートでも使うこともできます。この例では、ポート 5 の下位 4 ビットをドライバ IC の入力端子に接続しています（ハードウェア回路例 1）。巻線に通電するには、“In1”端子をハイレベル（2 進値 1）にし、“In2”端子をローレベル（2 進値 0）にします。図 6 のプログラムは、表 3 に記載のフルステップシーケンスでステッピングモータを連続して回転させる例です。

```
#include "iodefine.h"
#include <machine.h>

unsigned int loop_count, delay, max_step;
//an array of the positions used in the Full Stepping sequence
unsigned char motor_data[4] = {0x09,0x0a,0x06,0x05};
void main(void)
{
    P_IO.PCR5.BYTE = 0xFF;        //set port 5 as output port
    P_IO.PDR5.BYTE = 0x00;        //set port 5 data as 0000 0000
    loop_count = 0; max_step = 4; //Initialises the variable

    while(1)
    {
    //output stepping sequence
    P_IO.PDR5.BYTE = motor_data[loop_count++];
    //Coil energizing time
    for (delay=0 ; delay<0x300; delay++) ;
    //return to beginning
    if (loop_count==max_step) loop_count = 0;
    }
}
```

図 6　ステッピングモータ制御のソースプログラム例 1

2.1 ソースプログラム例 1 の説明

motor_data と max_step の値を変更するだけで、別のステップシーケンスを用いることができます。例えば、ハーフステップシーケンスでは、max_step を 8 に、motor_data[4]を motor_data[8]={0x08, 0x0a, …,0x01,0x09}に変更します（表 5 参照）。

MCU は各コマンドを数マイクロ秒で実行するので、ステップ間に遅延が必要です。これがないと、巻線に完全に通電されないので回転子が回転できません。この遅延は次に示すプログラム文で実現されています。

```
for (delay=0 ; delay<0x300; delay++) ;
```

ここで delay の値はモータの速度やトルクに影響します。delay の値を大きくすると、巻線に通電する時間が長くなり、モータのトルクは増加しますが速度は低下します。delay の値を小さくすると、モータの速度は向上しますがトルクが減少し、回転または制御ができなくなります。

したがって、ハードウェア回路例 2 では、各ステップシーケンスの前に遅延を挿入する必要があります。ハードウェア回路例 2 に適するようにソースプログラム例（図 6）を変更しなければなりません。図 7 のプログラムは、変更したステッピングモータ制御用ソースプログラムです。

```
#include "iodefine.h"
#include <machine.h>

unsigned int loop_count, delay, max_step;
//an array of the positions used in the Full Stepping sequence
unsigned char motor_data[8] = {  0x01, //Dead-time sequence 0
                                 0x09, //stepping sequence 0
                                 0x08, //Dead-time sequence 1
                                 0x0A, //stepping sequence 1
                                 0x02, //Dead-time sequence 2
                                 0x06, //stepping sequence 2
                                 0x04, //Dead-time sequence 3
                                 0x05  //stepping sequence 3
                               };
void main(void)
{
   P_IO.PCR5.BYTE = 0xFF;        //set port 5 as output port
   P_IO.PDR5.BYTE = 0x00;        //set port 5 data as 0000 0000
   loop_count = 0; max_step = 8; //Initialises the variable
   while(1)
   {
      P_IO.PDR5.BYTE = motor_data[loop_count++];  //output deadtime sequence
   for (delay=0 ; delay<0x10; delay++) ;          //small delay
      P_IO.PDR5.BYTE = motor_data[loop_count++];  //output stepping sequence
   for (delay=0 ; delay<0x300; delay++) ;         //Coil energizing time
      if (loop_count==max_step) loop_count = 0;
    //return to beginning
   }

}
```

図 7　ステッピングモータ制御のソースプログラム例 2

ソースプログラム例 1 と 2 は、ステッピングモータ制御を実現する簡単な一方法を示すものです。より効率良く MCU の負荷を抑えながらステッピングモータ制御のステップシーケンスを生成する方法として、タイマ割り込みを用いる方法があります。ソースプログラム例 2 をタイマ割り込み駆動方式に変更してみましょう。この例では、タイマ F を用いてステッピングモータ制御波形を生成することにします。

タイマ割り込みを用いる場合、プログラムは 2 つの部分に分かれます。

a. main プログラム　— void main(void)
b. 割り込み処理ルーチン　— __interrupt(vect=15) void INT_TimerFH(void)

【注】＊詳細なソースプログラムは図 8 と図 9 を参照してください。

```
#include "iodefine.h"
#include <machine.h>
void init_stepper_motor_io(void);
void init_timer_F(void);

unsigned int loop_count, delay, max_step;
//an array of the positions used in the Full Stepping sequence
unsigned char motor_data[8] = {  0x01, //Dead-time sequence 0
                                 0x09, //stepping sequence 0
                                 0x08, //Dead-time sequence 1
                                 0x0A, //stepping sequence 1
                                 0x02, //Dead-time sequence 2
                                 0x06, //stepping sequence 2
                                 0x04, //Dead-time sequence 3
                                 0x05  //stepping sequence 3
                              };
void main(void)
{
    init_stepper_motor_io();      //initialise I/O port
       init_timer_F();            //initialise Timer F with interrupt
    while(1)
    {
        //user program start here
    }
}
void init_stepper_motor_io(void)
{
    P_IO.PCR5.BYTE = 0xFF;
        //set port 5 as output port

    P_IO.PDR5.BYTE = 0x00;
        //set port 5 data as 0000 0000
}


void init_timer_F(void)
{
    set_imask_ccr(1);              //disable interrupt request
    P_TMRF.TCRF.BYTE = 0x04;       //set timer F 16bit mode, counting on TCFL
                                   //overflow signal
    P_TMRF.TCSRF.BIT.CCLRH = 1;    //TCF clearing by compare match


    P_TMRF.OCRF.BYTE.H = 0x00;
    P_TMRF.OCRF.BYTE.L =0x30;      //set output compare register value

    P_SYSCR.IENR2.BIT.IENTFH = 1;  //Enable Timer F interrupt
    P_SYSCR.IRR2.BIT.IRRTFH = 0;   //clear Timer F interrupt request flag
    set_imask_ccr(0);              //enable interrupt request
}
```

図 8　ステッピングモータ制御の main プログラム（タイマ割り込みを用いる場合）

```
#include "iodefine.h"
#include <machine.h>
extern unsigned char motor_data[8];
extern unsigned char loop_count;
#pragma section IntPRG
// vector 1 Reserved

// vector 2 Reserved

// vector 3 Reserved

// vector 4 IRQ0
__interrupt(vect=4) void INT_IRQ0(void) { /* sleep(); */ }
.
.
.
.

__interrupt(vect=14) void INT_TimerFL(void) {/* sleep(); */}
// vector 15 Timer FH Overflow
__interrupt(vect=15) void INT_TimerFH(void)
{
   if (P_TMRF.TCSRF.BIT.CMFH == 1)
   {
       P_TMRF.TCSRF.BIT.CMFH = 0;
       if (loop_count%2 == 0) //DEAD-TIME_DATA
       {
                P_IO.PDR5.BYTE = motor_data[loop_count++];
                P_TMRF.OCRF.BYTE.H = 0x00;
                P_TMRF.OCRF.BYTE.L = 0x30;
       }
       else


             //STEPPING_DATA
        {
             P_IO.PDR5.BYTE = motor_data[loop_count++];
             P_TMRF.OCRF.BYTE.H = 0x03;
             P_TMRF.OCRF.BYTE.L = 0x00;
        }

        if (loop_count==8) loop_count = 0;
   }

P_SYSCR.IRR2.BIT.IRRTFH= 0;
}
// vector 16 Timer G Overflow
__interrupt(vect=16) void INT_TimerG(void) {/* sleep(); */}
                                    // vector 17 Reserved
```

図９　ステッピングモータ制御の割り込みプログラム（タイマ割り込みを用いる場合）

2.2 タイマ割り込みを用いたソースプログラム例の説明

まず、MCU がステッピングモータ I/O ポートの出力値を 0x00 に初期化します。次に、タイマ F を 16 ビットフリーランカウンタに、タイマクロックを∅/32 に設定します。タイマ F カウンタ値（TCF）は、OCRF と TCF のコンペアマッチが発生したときに 0x0000 にクリアされるようにします。最後に、タイマ F 割り込みを許可して、コンペアマッチ割り込み要求を発生できるようにします。

処理は、タイマ初期化プログラムで初期カウントを OCRF にロードすることから開始します。この後、タイマ F 割り込み処理ルーチンが残りの処理を行ないます。割り込みルーチンは、OCRF の値が TCF の値と一致したときに実行されます。次に、デッドタイムシーケンスデータをポート 5 に出力し、OCRF をデッドタイム期間分だけ更新します。この後、OCRF が再度 TCF と一致すると、2 回目のタイマ割り込み要求が発生し、今回はステッピングモータシーケンスデータをポート 5 に出力し、OCRF を再度デッドタイム期間分だけ更新します。

割り込みルーチンは以下の処理を行ないます:

a. タイマ F 割り込み要求が発生すると、割り込み処理ルーチンが実行されます。
b. コンペアマッチフラグをチェックし、‘1’の場合は残りの処理を続けます。
c. CMFH フラグをクリアします。
d. loop_count 値をチェックし、偶数値であればデッドタイムシーケンスデータを出力し、OCRF 値をデッドタイム期間分だけ更新します。
e. loop_count 値が奇数であれば、ステッピングデータをポート 5 に出力し、OCRF 値をコイル通電期間分だけ更新します。
f. 8 つのシーケンスデータをすべてポート 5 に出力し終えたら、loop_count をリセットします。
g. 割り込み要求フラグを‘0’にクリアします。

3. その他の検討事項

アプリケーションの必要に応じて、ステッピングモータのドライブ処理をさらに強化できます。例えば、ステッピングモータをできるだけ短時間に回数を多く回転させたい場合があります。モータを起動するときは、モータ負荷を回転させるのに大きいトルクが必要なため、最初は遅延が長くなるかもしれません。しかし、この遅延は、モータの回転イナーシャにより、徐々に短縮することができます。このようにすれば、モータを高速に回転させることができます。こうした遅延時間は、モータの特性や負荷などから、計算で求めることができます。

本アプリケーションで取りあげたステッピングモータは、汎用低電力ステッピングモータ（最大 400mW）ですが、電力の大きいステッピングモータを使用する場合はドライバ回路に変更を加える必要があります。

参考文献

1. H8/38024 Series, H8/38024F-ZTAT Hardware Manual (ADE-602-231A)
2. H8/300 Using a H8/300 to control a stepper motor Application Note (AE-0057)

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.09.19	—	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。