To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

M32C/83 Group

Concept of the Three-phase Motor Control Program using Intelligent I/O functions

1.0 Abstract

This application note describes the concept of the three-phase motor control waveform output program using the function 2 and 3 of intelligent I/O.

2.0 Introduction

The explanation of this issue is applied of the following condition.

Applicable MCU: M32C/83 Group

This program can also be used when operating other microcomputers within the M16C family, provided they have the same SFR (Special Function Registers) as the M32C/83 microcomputers. However, some functions may have been modified. Refer to the User's Manual for details. Use functions covered in this Application Note only after careful evaluation.

3.0 Outline of Inverter Control

3.1 About Inverter Control

Inverter control is a method of controlling motor drive by changing the applied frequency as necessary.
For example, three-phase motors are driven by applying a waveform that is 120 degrees out of phase, but even though a commercial power supply consisting of three phases is used, the applied frequency always depends on the frequency of the commercial power supply.

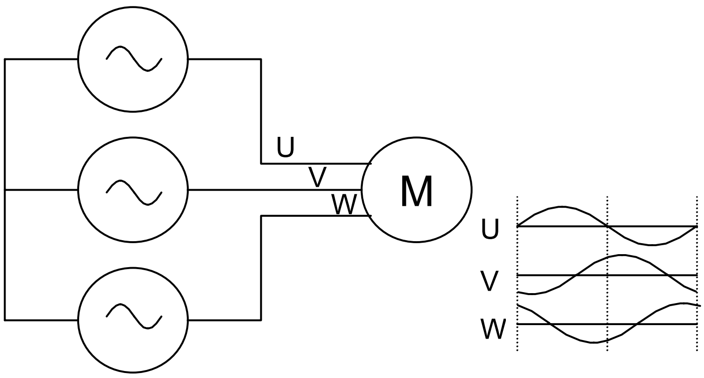


Figure 3.1 Three-phase Motor Drive

In inverter control, the commercial a.c. power supply is temporarily converted into a d.c. power supply, from which the frequency required for motor drive is produced by switching a transistor on and off.
Because this transistor switching is controlled by a microcomputer, any motor drive frequency can be produced by changing switching intervals.

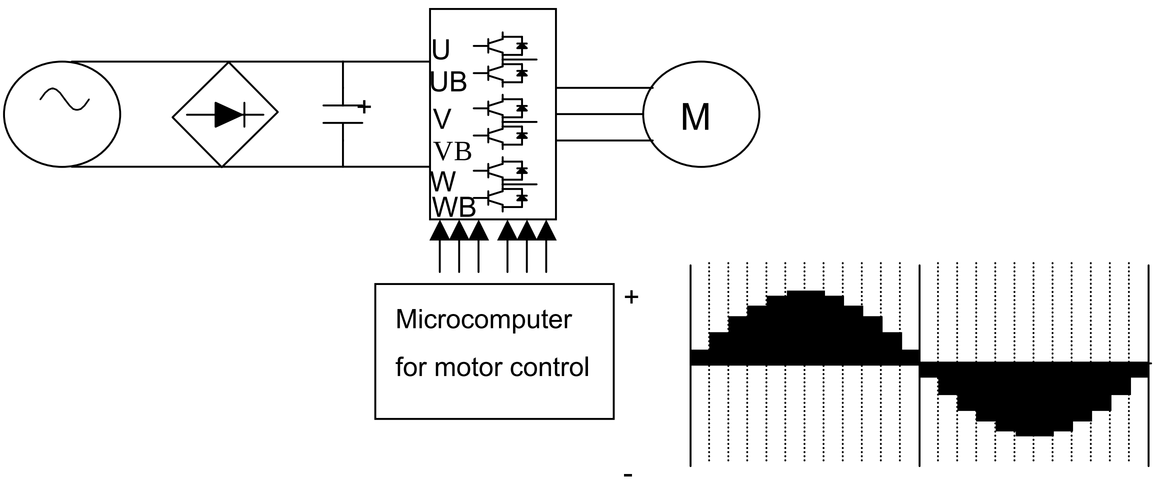


Figure 3.2 Example of Inverter Control using Microcomputer

3.2 Waveform Output by a Microcomputer

Because a.c. waveforms or motor drive high voltages cannot be output from a microcomputer port, a power transistor circuit like the one conceptually depicted in Figure 3.3 must be inserted between the microcomputer and the motor. The transistors U, V, UB, W, VB and WB in this diagram accept as input of the signals output from the microcomputer pins.

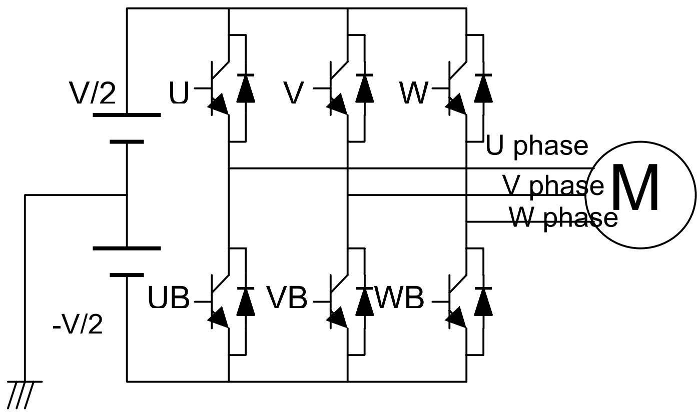
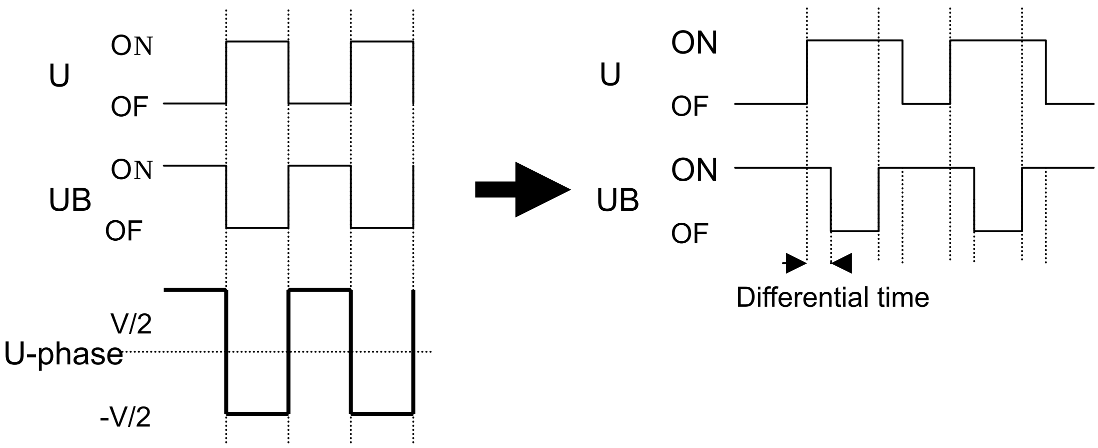


Figure 3.3 Power Transistor Circuit

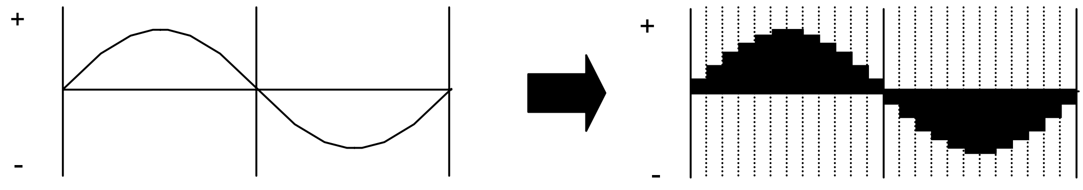
To explain motor control using only the U phase in Figure 3.3 for convenience's sake, if turn-on and off signals like those shown in Figure 3.4 are applied alternately to U and UB, the voltage levels also are inverted, producing an alternating (square) waveform in U phase.



*Active-low case
Figure 3.4 Microcomputer Output Wavefrom and Generated Waveform

One thing to be noted here is that if the positive and negative phases of two transistors turn on at the same time, a through correct flows, causing the d.c. power supply to be shorted. To solve this problem, a differential switchover timing must be produced to prevent simultaneous turn-on.

Application of a voltage with an equal amount of area in an equal duration of time to a motor has the same effect as applying a voltage that is approximated to an a.c. waveform output can be obtained by changing the widths of high and low outputs from a microcomputer.



* The smaller the division, the greater the proximity of a voltage waveform to a sine wave.

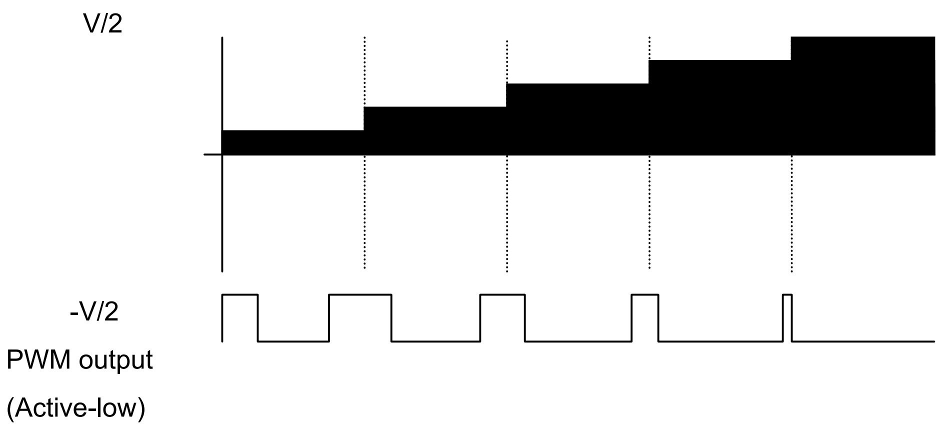


Figure 3.5 Changing an a.c. sine wave to a square wave by dividing it in time

4.0 Using Waveform Generation Function

4.1 How to Output a Three-phase Waveform

This section describes the basic method for producing three-phase output waveform using intelligent I/O.

4.2 Carrier Frequency

The reference frequency for the PWM pulse width with which transistors are switched on and off is known as the carrier frequency. When a sine wave is superimposed, this carrier frequency has intersecting points, at which the switching waveform has its levels inverted.

There are two types of carrier frequencies: sawtooth wave modulation method and triangular wave modulation method.

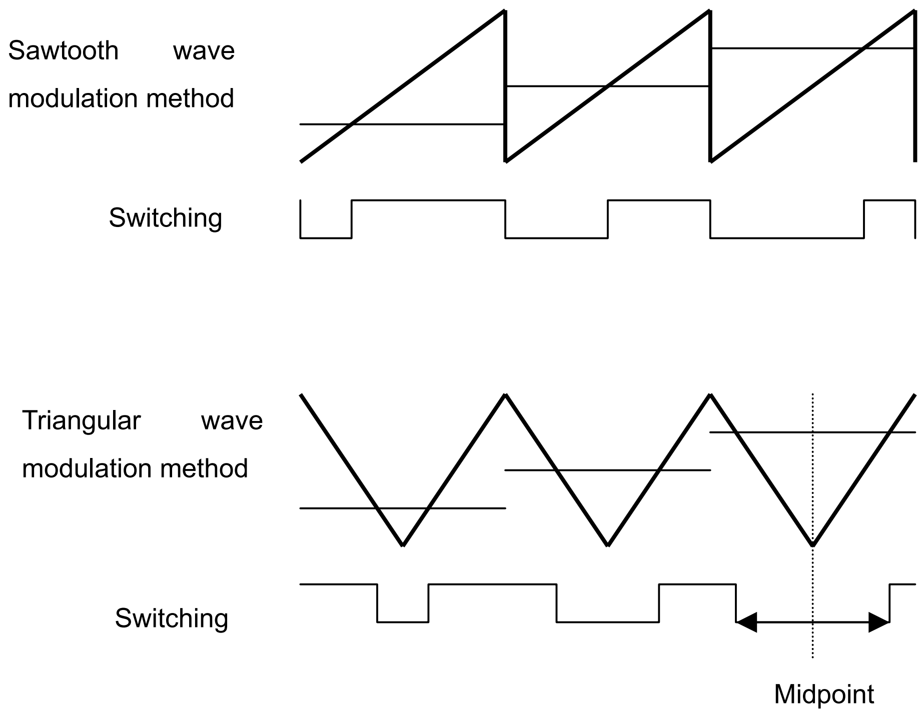


Figure 4.2 Carrier Frequency Modulation Method

In the sawtooth wave modulation method, the duty cycle is varied with respect to the beginning of the carrier cycle, whereas in the triangular wave modulation method, the duty cycle is varied with respect to the midpoint.

4.3 Method of Representation in PWM

In representing in the function to produce three-phase output waveform using intelligent I/O, a single PWM mode for the sawtooth wave modulation method and SR waveform output mode for the triangular wave modulation method are used.

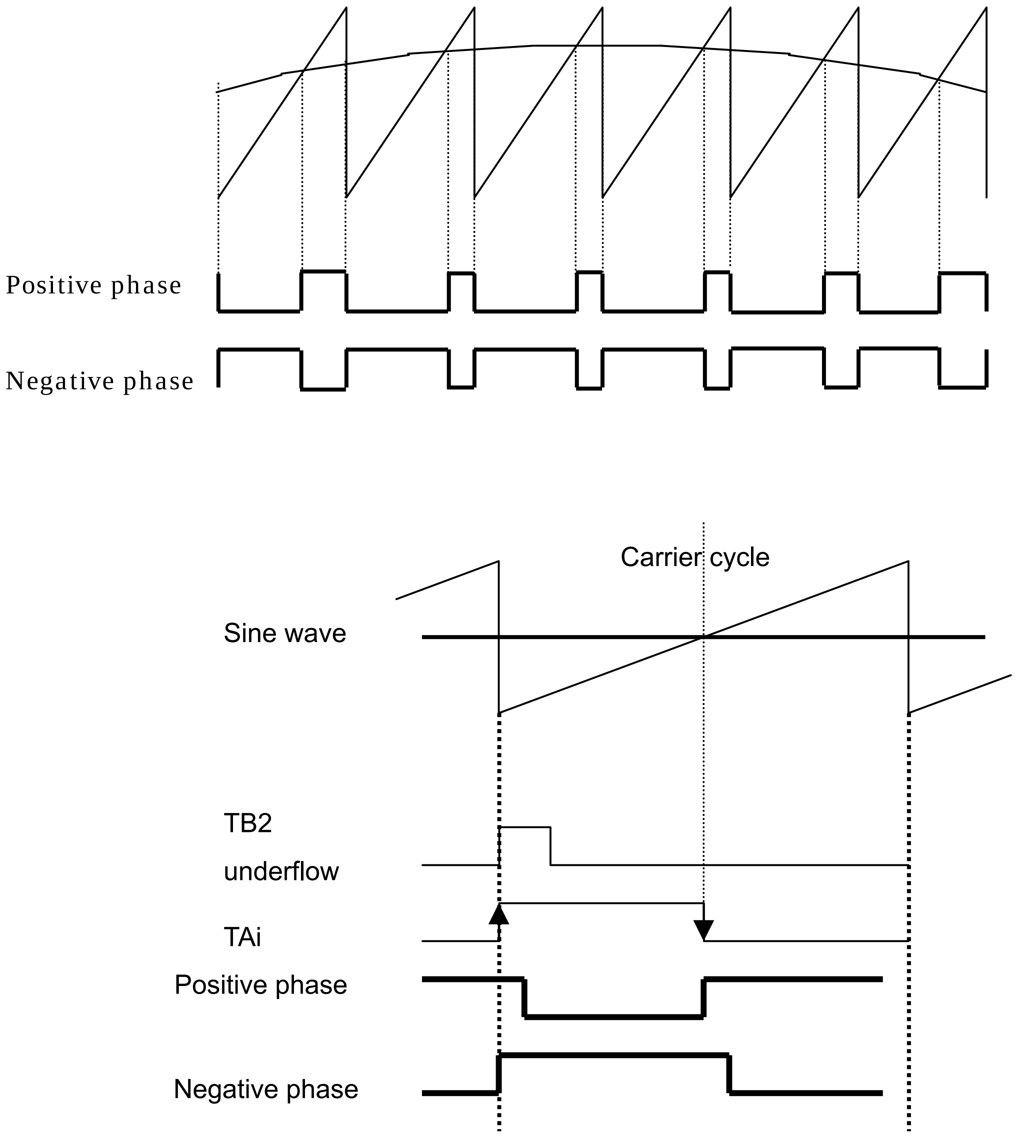
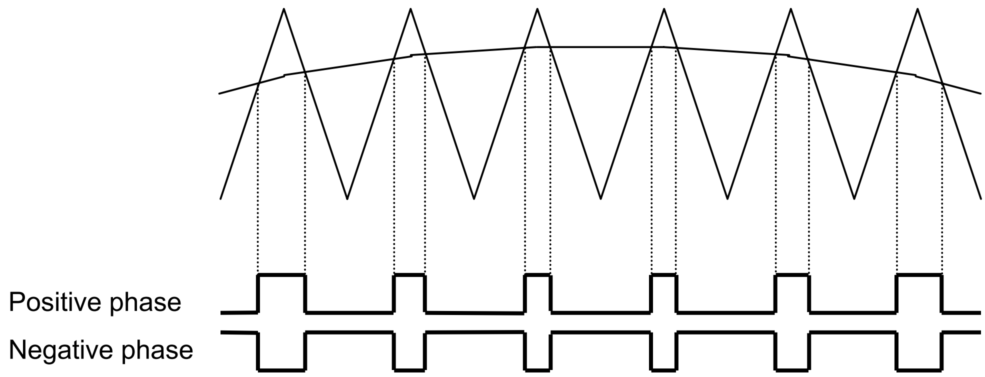
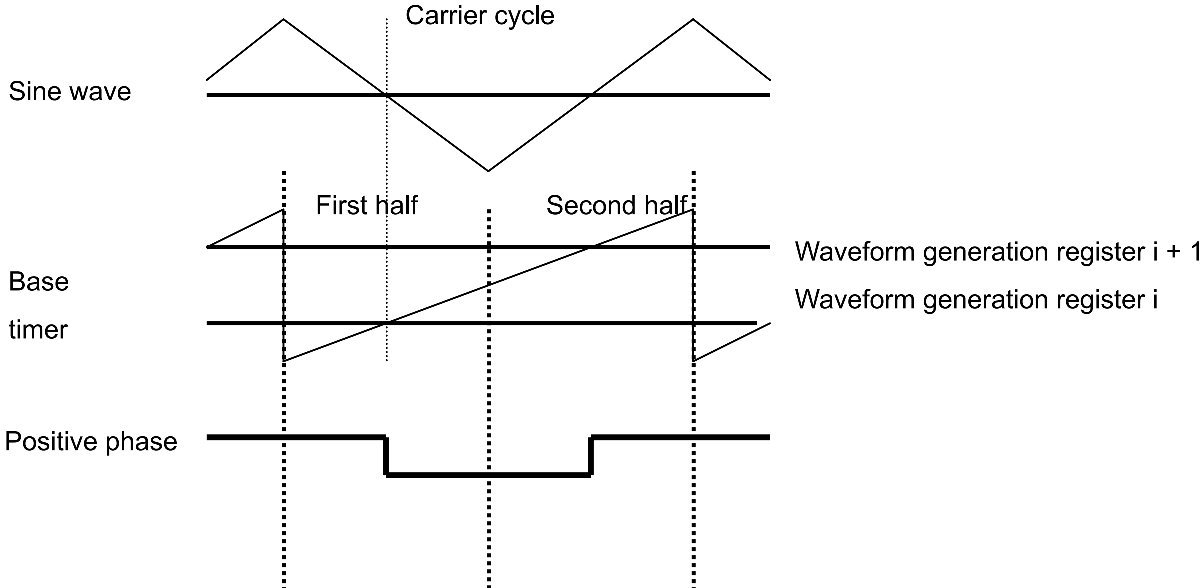


Figure 4.2 Relationship between Sawtooth Wave Modulation and single PWM mode





The first half and the second half basically are considered to be symmetrical.
(Actually, inclined and not symmetrical because they are arced parts of a sine wave)

Figure 4.3 Triangular wave modulation and SR waveform output mode

4.4 Procedures to Fix the Output Level

To fix the output level,

- Write “0” and “the value of larger than 2 + reload value of the base timer” in the wavefrom generation register
- Reset to a normal port in the function select register

These procedures allow producing two-phase modulation output or 120-degree turn-on output.

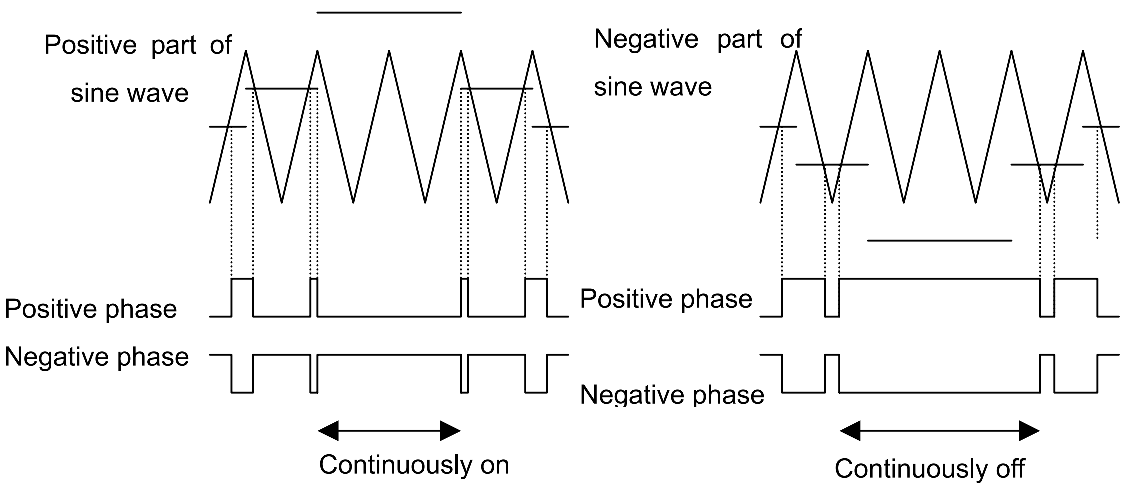


Figure 4.4 Conceptual Waveform of Two-phase Modulation Output

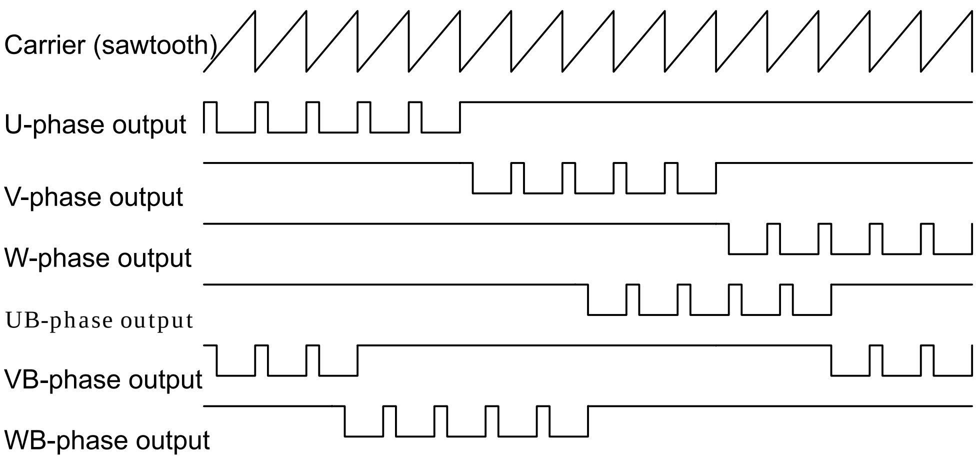


Figure 4.5 Conceptual Waveform of 120-degree Turn-on Output

5.0 Method for Producing Three-phase Sine Wave Output

5.1 To produce Three-phase Waveform Output

This section describes the method for producing three-phase sine wave output using intelligent I/O functions.

5.2 Using Intelligent I/O Functions

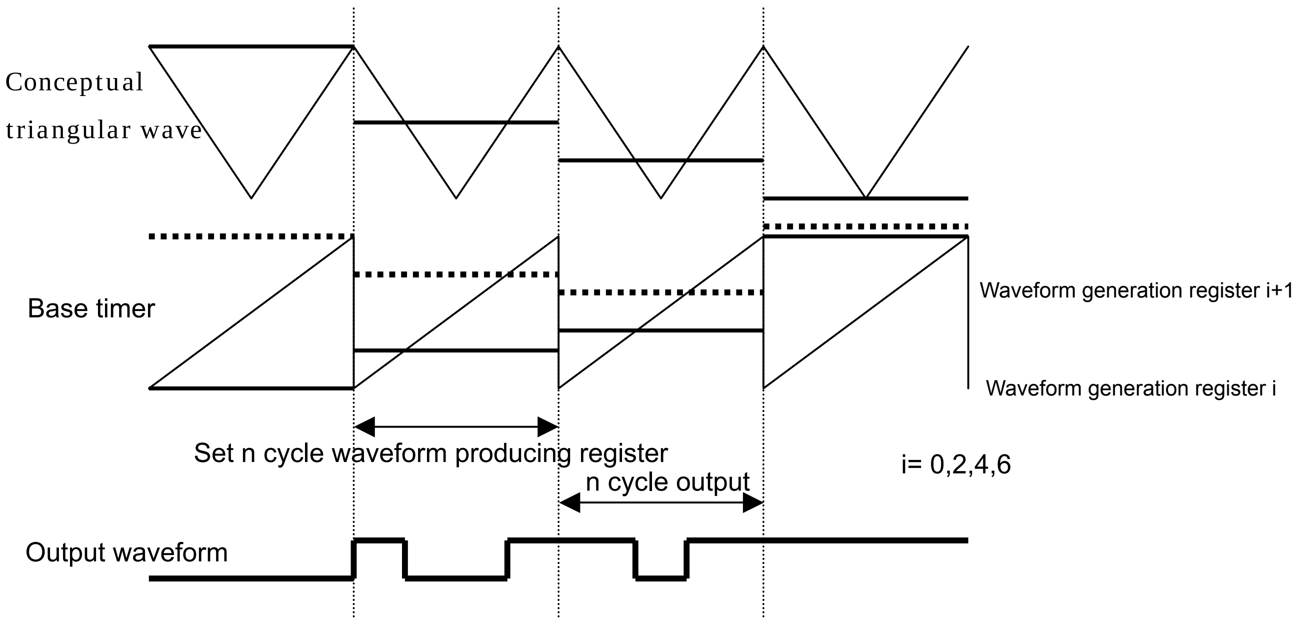


Figure 5.1 Relationship between conceptual triangle waveform and intelligent I/O

Base timer count is reload setting value + 2. The range of setting value in the waveform generation register is usually

Waveform generation register i = 1 to reload setting value Waveform generation register i +1 = 2 to reload setting value +1

And, waveform generation register i < waveform generation register i +1

1 cycle of continuous ON waveform output and continuous OFF waveform output can be produced after setting “0” or “a value of larger than 2 + reload setting value”, a value of larger than base timer count.

5.3 Calculating PWM Data

The following shows how to calculate the values to be set in the waveform generation register.

The output waveform consists of a 50% duty cycle around sin0°. The value to be set in the waveform generation register is obtained by adding or subtracting with respect to a 50% duty cycle.

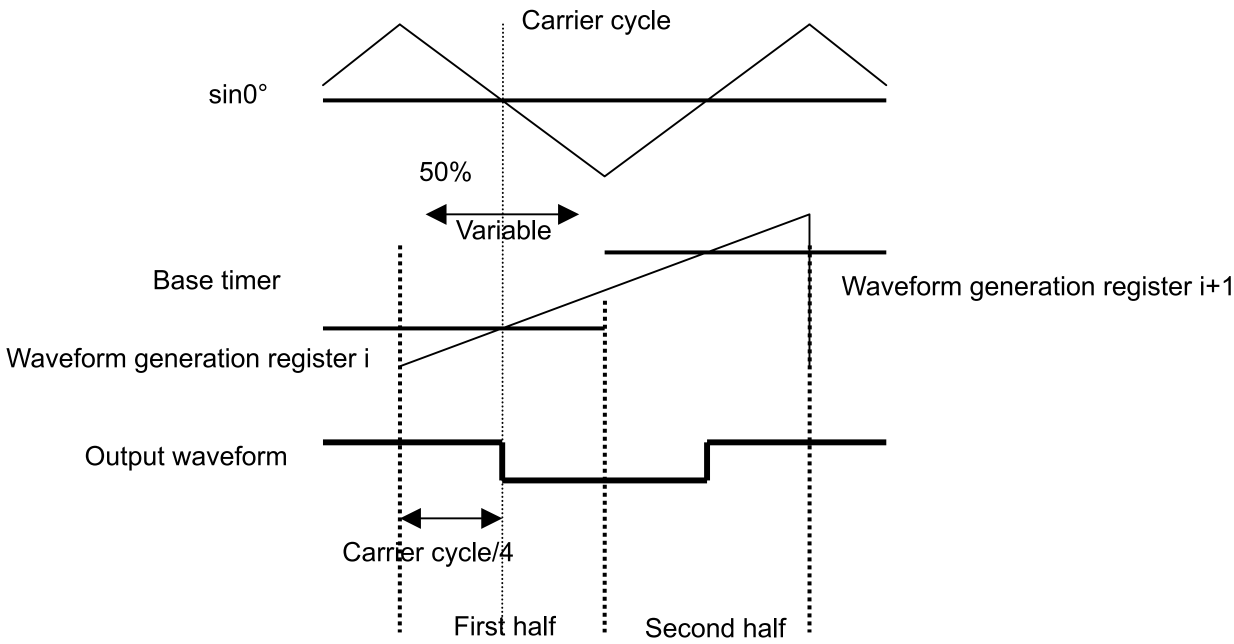


Figure 5.2 Relation between carrier frequency and output waveform

Because 50% of duty cycle = carrier cycle/4,

The first half data = carrier cycle/4 – variable value.

Furthermore, because the sine wave takes on values –1, 0 and +1

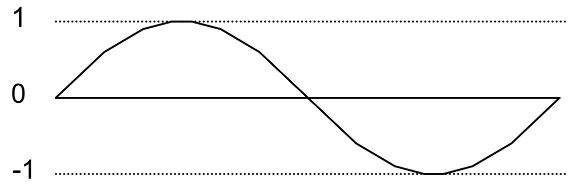


Figure 5.3 Sine Wave

To find the variable value necessary to produce a duty cycle consisting of –50%, 0 and 50%,

Variable value = 50% X sinN^{o}

= carrier cycle/4 X sinN^{o}

Thus, the setting values can be obtained from the equation below

The first half data = carrier cycle/4 – carrier cycle/4 X sinN^{o}

When putting this equation in terms of the relationship between base timer and waveform generation register, the following equation can be obtained.

Base timer reload value /4 = carrier cycle/4

Thus, the following equations can be obtained.

Waveform generation register i = base timer reload value/4 – base timer reload value/4 X sinN^{o}

Waveform generation register i +1 = base timer reload value – value in the waveform generation register i

5.4 Altering the Output Waveform Relative to Load

Information on PWM load is reflected by multiplying the PWM waveform that is output for each phase by a modulation rate.

Waveform generation register i = carrier cycle/4 – carrier cycle/4 X sinN^{o} X modulation rate

If V/F control is desired, for example, alter the value of the modulation rate in accordance with alteration of the output frequency to control the relationship between PWM wavelength (V) and output waveform (F).

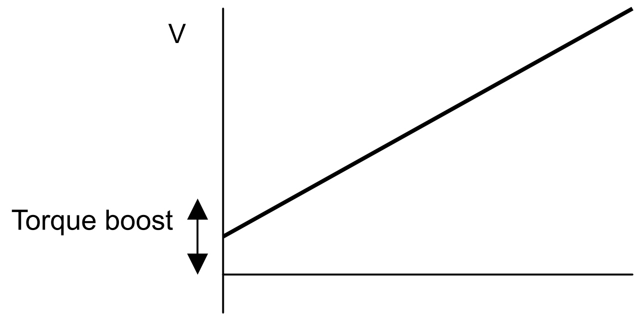


Figure 5.4 V/F Control

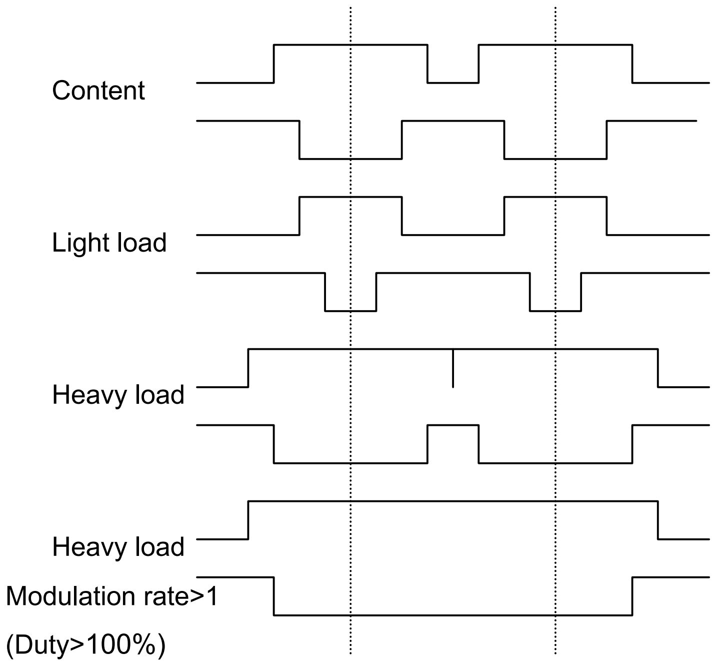


Figure 5.5 Relationship between Modulation Rate and Output Waveform

5.5 Two-phase Modulation Output (Continuously On, Continuously Off)

To prevent part of waveform equivalent to the amount of shorting prevention time from being lost when the PWM duty cycle is expanded or narrowed from a given width, there is a method of control to keep the waveform turned on or off for a duration greater than the carrier cycle.

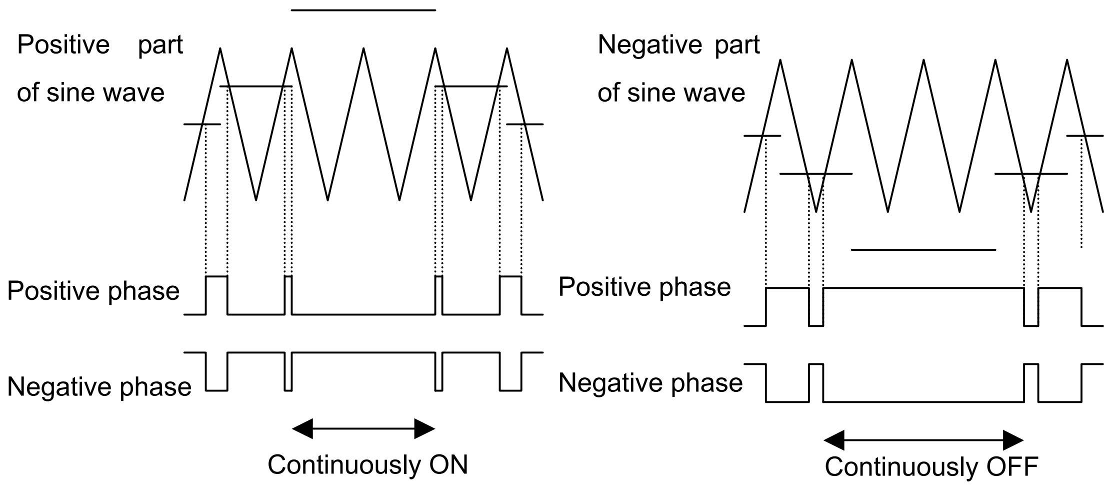


Figure 5.6 Conceptual Waveform of Two-phase Modulation Output

When producing continuously 1 cycle of ON waveform output (in low active) using intelligent I/O function, write "0" in the waveform generation register i. When producing continuously 1 cycle of OFF waveform (in low active) output, write " value larger than 2 + reload value" in the waveform generation register i.

For example, when reload value = 1000 – 2 (base timer counts = 1000 in this example), the set value should be as follows.

Continuously ON waveform output

waveform generation register i = 0 waveform generation register i +1 = larger than 1000

Content waveform generation register i =1 to 998 waveform generation register i +1 = 2 to 999

Must be waveform generation register i < waveform generation register i + 1

Continuously OFF waveform output

waveform generation register i = larger than 1000

waveform generation register i + 1 = larger than 1001

Must be waveform generation register i < waveform generation register i + 1

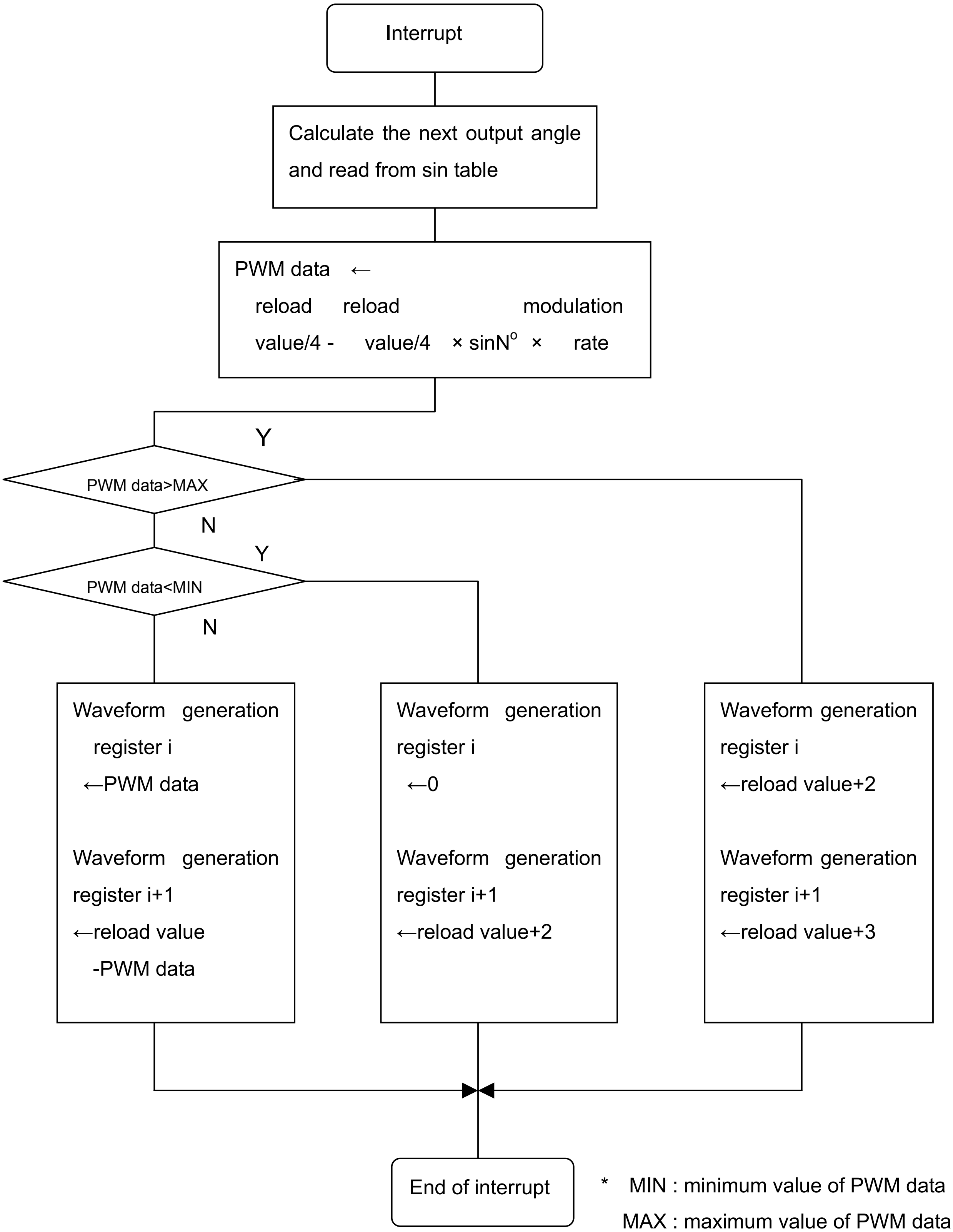
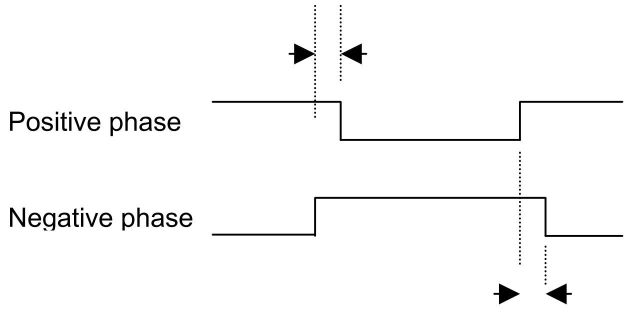


Figure 5.7 Example of a base timer reload interrupt Processing Flowchart

5.6 Producing the shorting prevention time

When producing negative-phase and positive-phase waveform output using intelligent I/O function, a shorting-free output waveform should be produced. A method of producing a shorting-free output waveform is shown below.



* The negative phase waveform output can be produced in the reverse output function select bit.

Figure 5.8 A shorting-free output waveform (active low)

The positive phase waveform generation register i = The positive phase waveform generation register i + a shorting prevention time

The positive phase waveform generation register i +1 = The positive phase waveform generation register i + 1

The positive phase waveform generation register i = The positive phase waveform generation register i

The positive phase waveform generation register i +1 = The positive phase waveform generation register i +1 + a shorting prevention time

6.0 Materializing 120-degree Turn-on Control

6.1 To produce 120-degree Turn-on Waveform Output

This section shows an example for producing 120-degree turn-on waveform output using the intelligent I/O functions.

6.2 Using Intelligent I/O Functions to Produce Waveform Output

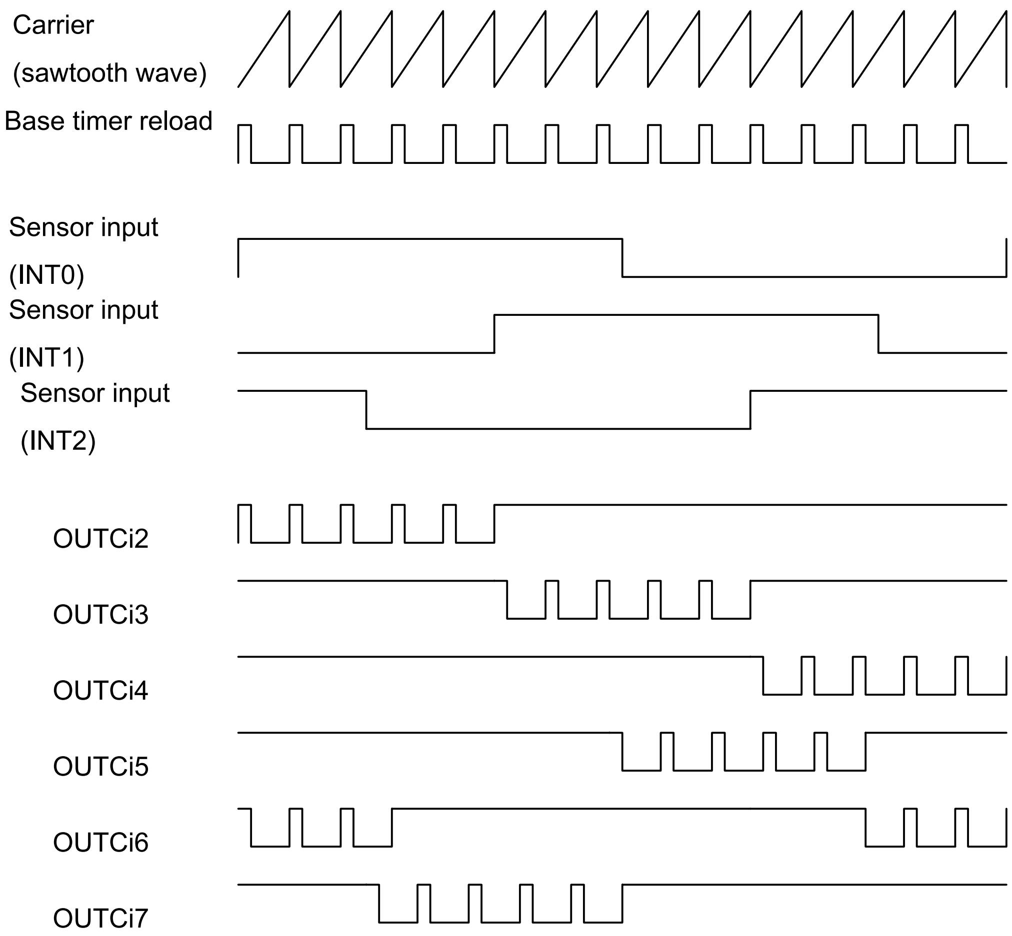


Figure 6.1 Relationship between 120-degrre Turn-on Control Sensor Input and Waveform Output

A single waveform output mode is used. Switch active phases from one to another by

-rewriting “0” or “value of larger than 2+ base timer reload value” in the waveform generation register

-changed to general-purpose ports by writing function select register

within an INT interrupt.

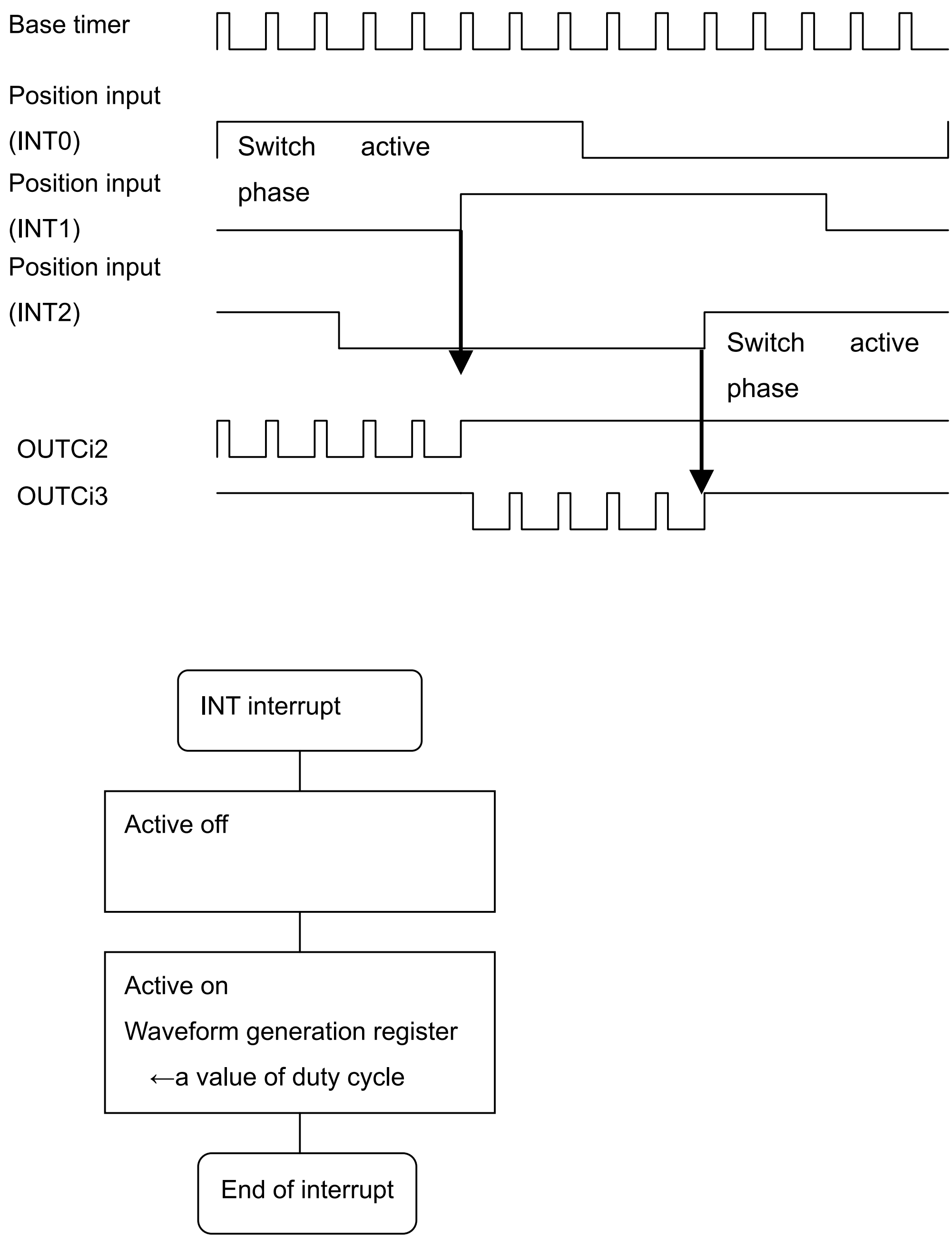
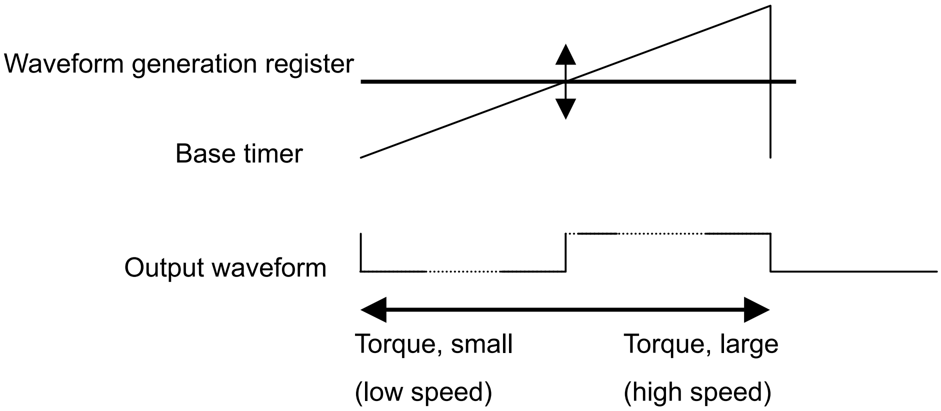


Figure 6.2 Position Interrupt and Output Phases

6.3 Speed Control

In 120-degree turn-on control, the relationship between torque and speed basically is proportional. More specifically, this control is accomplished by rewriting the values of the waveform generation register each time the torque instruction value changes.



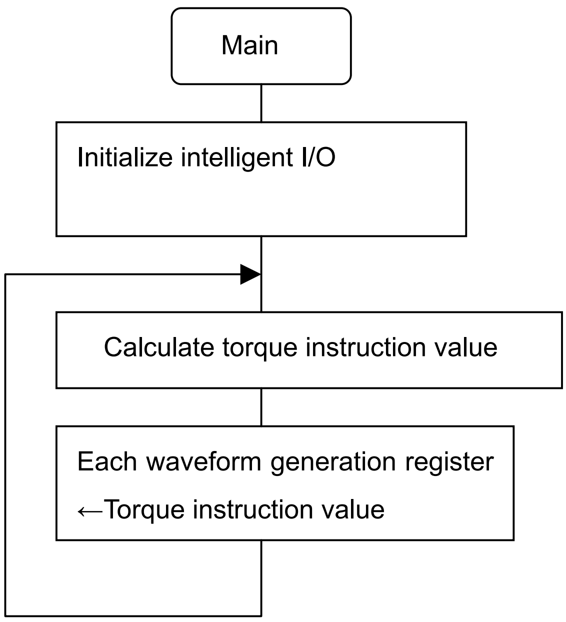


Figure 6.3 Relationship between Torque Instruction Value and Duty Cycle

7.0 Reference Program Example

An example to do three-phase motor waveform output is shown. Changes and adjustment are necessary in proportion to the each user application for the application program example.

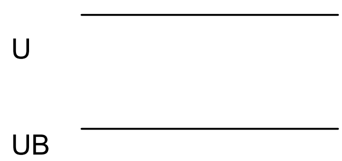
7.1 A sine wave form output reference program

An example to do three-phase sine waveform output is shown.
In this example, Group 2 registers are for the positive phase waveform output, and Group 3 for the negative phase waveform output. Output waveform shown below is a shorting-free waveform output. Just one example of a method of producing a shorting-free waveform output is shown here. See if it is necessary for the user system.

The relationship between the setting value and output waveform in the program example to produce waveform output is shown below.

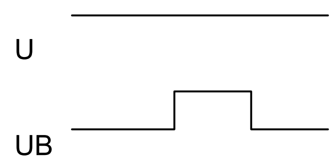
Pattern1:

```
g2po2=c_dat;
g2po3=c_dat+1;
g3po2=c_dat;
g3po3=c_dat+1;
```



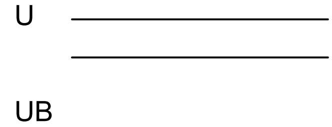
Pattern2:

```
g2po2=c_dat;
g2po3=c_dat+1;
g3po2=pwm_u_w;
g3po3=c_dat+d_t-pwm_u_w;
```



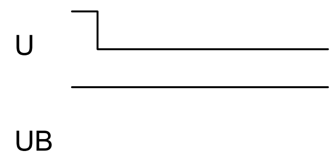
Pattern3:

```
g2po2=0;
g2po3=c_dat;
g3po2=0;
g3po3=c_dat;
```



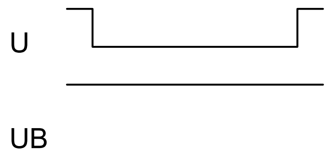
Pattern4:

```
g2po2=d_t;
g2po3=c_dat;
g3po2=0;
g3po3=c_dat;
```



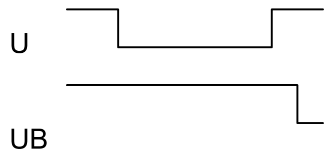
Pattern5:

```
g2po2=pwm_u_w+d_t;
g2po3=c_dat-pwm_u_w;
g3po2=0;
g3po3=c_dat;
```



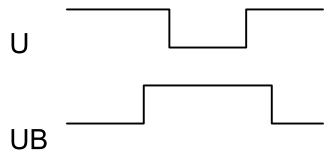
Pattern6:

```
g2po2=pwm_u_w+d_t;
g2po3=c_dat-pwm_u_w;
g3po2=0;
g3po3=c_dat+d_t-pwm_u_w;
```



Pattern7:

```
g2po2=pwm_u_w+d_t;
g2po3=c_dat-pwm_u_w;
g3po2=pwm_u_w;
g3po3=c_dat+d_t-pwm_u_w;
```



```
/*********************************************************************************
 *          sine waveform output reference program
 *
 *********************************************************************************/
/*-----------------------------------------------------------------------------------*/
/*     Copyright (C) 2003 Renesas Technology Corporation          */
/*     Copyright (C) 2003 Renesas Solutions Corporation           */
/*     All rights reserved.                                       */
/*******************************************************************************/

/*********************************************************************************/
/*                                                                */
/*      SFR setting                                               */
/*                                                                */
/*********************************************************************************/

volatile char iio6ir;
#pragma ADDRESS     iio6ir      00A6h     /* interrupt request register 6 */
volatile char int6ie;
#pragma ADDRESS     int6ie      00B6h     /* interrupt enable register 6 */
volatile char iio6ic;
#pragma ADDRESS     iio6ic      007Bh     /* intellingent I/O interrupt control register6 */
volatile int g2po0;
#pragma ADDRESS     g2po0       0140h     /* Group 2 waveform generation register 0 */
volatile int g2po1;
#pragma ADDRESS     g2po1       0142h     /* Group 2 waveform generation register 1 */
volatile int g2po2;
#pragma ADDRESS     g2po2       0144h     /* Group 2 waveform generation register 2 */
volatile int g2po3;
#pragma ADDRESS     g2po3       0146h     /* Group 2 waveform generation register 3 */
volatile int g2po4;
#pragma ADDRESS     g2po4       0148h     /* Group 2 waveform generation register 4 */
volatile int g2po5;
#pragma ADDRESS     g2po5       014Ah     /* Group 2 waveform generation register 5 */
volatile int g2po6;
#pragma ADDRESS     g2po6       014Ch     /* Group 2 waveform generation register 6 */
volatile int g2po7;
#pragma ADDRESS     g2po7       014Eh     /* Group 2 waveform generation register 7 */
volatile char g2pocr0;
#pragma ADDRESS     g2pocr0     0150h     /* Group 2 waveform generation cotrol register 0 */
volatile char g2pocr1;
#pragma ADDRESS     g2pocr1     0151h     /* Group 2 waveform generation control register 1
*/
volatile char g2pocr2;
#pragma ADDRESS     g2pocr2     0152h     /* Group 2 waveform generation control register 2 */
volatile char g2pocr3;
#pragma ADDRESS     g2pocr3     0153h     /* Group 2 waveform generation control register 3 */
volatile char g2pocr4;
#pragma ADDRESS     g2pocr4     0154h     /* Group 2 waveform generation control register 4 */
volatile char g2pocr5;
#pragma ADDRESS     g2pocr5     0155h     /* Group 2 waveform generation control register 5 */
volatile char g2pocr6;
#pragma ADDRESS     g2pocr6     0156h     /* Group 2 waveform generation control register 6 */
```

```
volatile char g2pocr7;
#pragma ADDRESS     g2pocr7     0157h    /* Group 2 waveform generation control register7 */
volatile char g2bcr0;
#pragma ADDRESS     g2bcr0      0162h    /* Group 2 base timer control register0 */
volatile char g2bcr1;
#pragma ADDRESS     g2bcr1      0163h    /* Group 2 base timer control register1 */
volatile char btsr;
#pragma ADDRESS     btsr        0164h    /* Group3 function enable flag */
volatile char g2fe;
#pragma ADDRESS     g2fe        0166h    /* Group3 function enable flag */
volatile int g3po0;
#pragma ADDRESS     g3po0       0180h    /* Group 3 waveform generation register 0 */
volatile int g3po1;
#pragma ADDRESS     g3po1       0182h    /* Group 3 waveform generation register 1 */
volatile int g3po2;
#pragma ADDRESS     g3po2       0184h    /* Group 3 waveform generation register 2 */
volatile int g3po3;
#pragma ADDRESS     g3po3       0186h    /* Group 3 waveform generation register 3 */
volatile int g3po4;
#pragma ADDRESS     g3po4       0188h    /* Group 3 waveform generation register 4 */
volatile int g3po5;
#pragma ADDRESS     g3po5       018Ah    /* Group 3 waveform generation register 5 */
volatile int g3po6;
#pragma ADDRESS     g3po6       018Ch    /* Group 3 waveform generation register 6 */
volatile int g3po7;
#pragma ADDRESS     g3po7       018Eh    /* Group 3 waveform generation register 7 */
volatile char g3pocr0;
#pragma ADDRESS     g3pocr0     0190h    /* Group 3 waveform generation control register 0 */
volatile char g3pocr1;
#pragma ADDRESS     g3pocr1     0191h    /* Group 3 waveform generation control register 1 */
volatile char g3pocr2;
#pragma ADDRESS     g3pocr2     0192h    /* Group 3 waveform generation control register 2 */
volatile char g3pocr3;
#pragma ADDRESS     g3pocr3     0193h    /* Group 3 waveform generation control register 3 */
volatile char g3pocr4;
#pragma ADDRESS     g3pocr4     0194h    /* Group 3 waveform generation control register 4 */
volatile char g3pocr5;
#pragma ADDRESS     g3pocr5     0195h    /* Group 3 waveform generation control register 5 */
volatile char g3pocr6;
#pragma ADDRESS     g3pocr6     0196h    /* Group 3 waveform generation control register 6 */
volatile char g3pocr7;
#pragma ADDRESS     g3pocr7     0197h    /* Group 3 waveform generation control register 7 */
volatile int g3mk4;
#pragma ADDRESS     g3mk4       0198h    /* Group 3 waveform generation mask register4*/
volatile int g3mk5;
#pragma ADDRESS     g3mk5       019ah    /* Group 3 waveform generation mask register 5*/
volatile int g3mk6;
#pragma ADDRESS     g3mk6       019ch    /* Group 3 waveform generation mask register 6*/
volatile int g3mk7;
#pragma ADDRESS     g3mk7       019eh    /* Group 3 waveform generation mask register 7*/
volatile char g3bcr0;
#pragma ADDRESS     g3bcr0      01A2h    /* Group 3 base timer control register0 */
volatile char g3bcr1;
#pragma ADDRESS     g3bcr1      01A3h    /* Group 3 base timer control register 1 */
```

```
volatile char g3fe;
#pragma ADDRESS     g3fe            01A6h       /*Group 3 function enable flag*/
volatile char ps6;
#pragma ADDRESS     ps6                     03BCh       /* function select register A6 */
volatile char ps7;
#pragma ADDRESS     ps7                     03BDh       /* function select register A7 */


/****************************************************************************/
/*                                                                         */
/*      Initialization                                                     */
/*                                                                         */
/****************************************************************************/
void main_ini(void);

#define CLK 30000000            /* Frequency of microcontroller Hz*/
#define CARR 20000                      /*carrier frequency Hz*/
#define DTT_TM 40                       /*shorting prevention timer ×0.1μs */

#define carr_set (CLK/CARR)                                 /*carrier frequency         */
#define carr_set_2 ((CLK/CARR)/2)                   /*carrier cycle 1/2  */
#define carr_set_4 ((CLK/CARR)/4)                   /*carrier frequency 1/4        */
#define dtt_set ((CLK*DTT_TM)/10000000)             /*a value of shorting prevention timer  */


void main_ini()
{

        g2bcr0=0x7f;        /*Group 2 base timer control register 0*/
        btsr=0x00;                  /*Group 2,3 base timer stop */
        g2bcr1=0x02;        /*Group 2 base timer control register1*/

        g2pocr0=0x00;       /*base timer reset factor */
        g2pocr2=0x01;       /*SR waveform output */
        g2pocr3=0x01;       /*SR waveform output */
        g2pocr4=0x01;       /*SR waveform output */
        g2pocr5=0x01;       /*SR waveform output */
        g2pocr6=0x01;       /*SR waveform output */
        g2pocr7=0x01;       /*SR waveform output */

        g2po0=carr_set-2; /* Group 2 waveform output register*/

        g2po2=carr_set;             /* Group 2 waveform generation register */
        g2po3=carr_set+1;           /* Group 2 waveform generation register */
        g2po4=carr_set;             /*Group 2 waveform generation register */
        g2po5=carr_set+1;           /* Group 2 waveform generation register */
        g2po6=carr_set;             /* Group 2 waveform generation register */
        g2po7=carr_set+1;           /* Group 2 waveform generation register */
```

```
g2pocr0=0x00;     /*base timer reset factor */

        g2pocr2=0xa1;     /*SR waveform output   reload base */
        g2pocr3=0xa1;     /*SR waveform output   reload base */
        g2pocr4=0xa1;     /*SR waveform output   reload base */
        g2pocr5=0xa1;     /*SR waveform output   reload base */
        g2pocr6=0xa1;     /*SR waveform output   reload base */
        g2pocr7=0xa1;     /*SR waveform output   reload base */


        g3bcr0=0x7f;      /*Group 3 base timer control register 0*/
        g3bcr1=0x02;      /* Group 3 base timer control register 1*/

        g3mk4=0x0000;
        g3mk5=0x0000;
        g3mk6=0x0000;
        g3mk7=0x0000;

        g3pocr0=0x00;     /*Base timer reset factor */
        g3pocr2=0x01;     /*SR waveform output   reload base */
        g3pocr3=0x01;     /*SR waveform output   reload base */
        g3pocr4=0x01;     /*SR waveform output   reload base */
        g3pocr5=0x01;     /*SR waveform output   reload base */
        g3pocr6=0x01;     /*SR waveform output   reload base */
        g3pocr7=0x01;     /*SR waveform output   reload base */

        g3po0=carr_set-2; /* Group 3 waveform generation register*/

        g3po2=0;                  /* Group 3 waveform generation register */
        g3po3=carr_set;           /* Group 3 waveform generation register */
        g3po4=0;                  /* Group 3 waveform generation register */
        g3po5=carr_set;           /* Group 3 waveform generation register */
        g3po6=0;                  /* Group 3 waveform generation register */
        g3po7=carr_set;           /* Group 3 waveform generation register */

        g3pocr0=0x00;     /*base timer reset factor */
        g3pocr2=0x21;     /*SR waveform output   reload base */
        g3pocr3=0x21;     /*SR waveform output   reload base */
        g3pocr4=0x21;     /*SR waveform output   reload base */
        g3pocr5=0x21;     /*SR waveform output   reload base */
        g3pocr6=0x21;     /*SR waveform output   reload base */
        g3pocr7=0x21;     /*SR waveform output   reload base */


        int6ie=0x01;      /*interrupt request disabled */
        iio6ir=0x00;              /*interrupt request clear */
        int6ie=0x05;      /*interrupt request enable*/
        iio6ic=0x07;      /*select interrupt level */

        ps7=0x25;
        ps6=0x54;
```

```
asm("   FSET    I");        /* interrupt enabled */

        btsr=0x0c;                  /*Group 2,3 base start */

        g2fe=0xfd;                  /*Group 2 enabled */
        g3fe=0xfd;                  /*Group 3 enabled */

}




/******************************************************************************/
/*                                                                            */
/*      sin table                                                             */
/*                                                                            */
/*          A sin table is multiplied by FFFF/2, and it prepares to increase operation precision. */
/*                                                                            */
/******************************************************************************/

const short sin_tbl[610]=
{
          572,   1144,   1715,   2286,   2856,        3425,   3993,   4560,   5126,   5690,
         6252,   6813,   7371,   7927,   8481,        9032,   9580, 10126, 10668, 11207,
        11743, 12275, 12803, 13328, 13848,       14364, 14876, 15383, 15886, 16384,
        16876, 17364, 17846, 18323, 18795,       19260, 19720, 20174, 20621, 21063,
        21497, 21926, 22347, 22762, 23170,       23571, 23965, 24351, 24730, 25101,
        25465, 25821, 26169, 26509, 26842,       27165, 27481, 27788, 28087, 28377,
        28659, 28932, 29196, 29451, 29697,       29935, 30163, 30381, 30591, 30791,
        30982, 31164, 31336, 31498, 31651,       31794, 31928, 32051, 32165, 32270,
        32364, 32449, 32523, 32588, 32643,       32688, 32723, 32748, 32763, 32767,
        32763, 32748, 32723, 32688, 32643,       32588, 32523, 32449, 32364, 32270,
        32165, 32051, 31928, 31794, 31651,       31498, 31336, 31164, 30982, 30791,
        30591, 30382, 30163, 29935, 29697,       29451, 29196, 28932, 28659, 28378,
        28087, 27788, 27481, 27166, 26842,       26510, 26169, 25821, 25465, 25101,
        24730, 24351, 23965, 23571, 23170,       22762, 22347, 21926, 21497, 21063,
        20621, 20174, 19720, 19260, 18795,       18323, 17847, 17364, 16877, 16384,
        15886, 15383, 14876, 14364, 13848,       13328, 12803, 12275, 11743, 11207,
        10668, 10126,   9580,   9032,   8481,        7927,   7371,   6813,   6252,   5690,
         5126,   4560,   3993,   3425,   2856,        2286,   1715,   1144,    572,      0,
         -572, -1143, -1715, -2286, -2856,          -3425, -3993, -4560, -5126, -5690,
        -6252, -6813, -7371, -7927, -8481,       -9032, -9580,-10126,-10668,-11207,
      -11743,-12275,-12803,-13328,-13848,       -14364,-14876,-15383,-15886,-16384,
      -16876,-17364,-17846,-18323,-18795,       -19260,-19720,-20174,-20621,-21062,
      -21497,-21926,-22347,-22762,-23170,       -23571,-23965,-24351,-24730,-25101,
      -25465,-25821,-26169,-26509,-26841,       -27165,-27481,-27788,-28087,-28377,
      -28659,-28932,-29196,-29451,-29697,       -29935,-30163,-30381,-30591,-30791,
      -30982,-31164,-31336,-31498,-31651,       -31794,-31928,-32051,-32165,-32270,
      -32364,-32449,-32523,-32588,-32643,       -32688,-32723,-32748,-32763,-32767,
      -32763,-32748,-32723,-32688,-32643,       -32588,-32523,-32449,-32364,-32270,
      -32165,-32051,-31928,-31794,-31651,       -31498,-31336,-31164,-30982,-30791,
```

```
        -30591,-30382,-30163,-29935,-29698,     -29451,-29196,-28932,-28659,-28378,
        -28087,-27788,-27481,-27166,-26842,     -26510,-26169,-25821,-25465,-25101,
        -24730,-24351,-23965,-23571,-23170,     -22762,-22347,-21926,-21498,-21063,
        -20621,-20174,-19720,-19260,-18795,     -18323,-17847,-17364,-16877,-16384,
        -15886,-15384,-14876,-14364,-13848,     -13328,-12803,-12275,-11743,-11207,
        -10668,-10126, -9580, -9032, -8481,      -7927, -7371, -6813, -6252, -5690,
         -5126, -4561, -3994, -3425, -2856,      -2286, -1715, -1144,  -572,     0,
           572,  1144,  1715,  2286,  2856,       3425,  3993,  4560,  5126,  5690,
          6252,  6813,  7371,  7927,  8481,       9032,  9580, 10126, 10668, 11207,
         11743, 12275, 12803, 13328, 13848,      14364, 14876, 15383, 15886, 16384,
         16876, 17364, 17846, 18323, 18795,      19260, 19720, 20174, 20621, 21063,
         21497, 21926, 22347, 22762, 23170,      23571, 23965, 24351, 24730, 25101,
         25465, 25821, 26169, 26509, 26842,      27165, 27481, 27788, 28087, 28377,
         28659, 28932, 29196, 29451, 29697,      29935, 30163, 30381, 30591, 30791,
         30982, 31164, 31336, 31498, 31651,      31794, 31928, 32051, 32165, 32270,
         32364, 32449, 32523, 32588, 32643,      32688, 32723, 32748, 32763, 32767,
         32763, 32748, 32723, 32688, 32643,      32588, 32523, 32449, 32364, 32270,
         32165, 32051, 31928, 31794, 31651,      31498, 31336, 31164, 30982, 30791,
         30591, 30382, 30163, 29935, 29697,      29451, 29196, 28932, 28659, 28378,
         28087, 27788, 27481, 27166, 26842,      26510, 26169, 25821, 25465, 25101,
         24730, 24351, 23965, 23571, 23170,      22762, 22347, 21926, 21497, 21063,
         20621, 20174, 19720, 19260, 18795,      18323, 17847, 17364, 16877, 16384,
         15886, 15383, 14876, 14364, 13848,      13328, 12803, 12275, 11743, 11207,
         10668, 10126,  9580,  9032,  8481,       7927,  7371,  6813,  6252,  5690,
          5126,  4560,  3993,  3425,  2856,       2286,  1715,  1144,   572,     0,
          -572, -1143, -1715, -2286, -2856,      -3425, -3993, -4560, -5126, -5690,
         -6252, -6813, -7371, -7927, -8481,      -9032, -9580,-10126,-10668,-11207,
        -11743,-12275,-12803,-13328,-13848,     -14364,-14876,-15383,-15886,-16384,
        -16876,-17364,-17846,-18323,-18795,     -19260,-19720,-20174,-20621,-21062,
        -21497,-21926,-22347,-22762,-23170,     -23571,-23965,-24351,-24730,-25101,
        -25465,-25821,-26169,-26509,-26841,     -27165,-27481,-27788,-28087,-28377,
        -28659,-28932,-29196,-29451,-29697,     -29935,-30163,-30381,-30591,-30791
};
```

```
/*********************************************************************************/
/*                                                                            */
/*      An operation example (in case of a guidance motor) in the main process. */
/*   Unit value calculation of a sin table                                    */
/*   A torque order value computation corresponding to the output frequency */
/*                                                                            */
/*********************************************************************************/
void main_pro(void);


signed short out_bin=100;           /*output frequency value     temporary value*/
signed short torq=1500;             /*torque data     temporary value       */
signed short tq_dat;                        /*torque order value X carrier /4*/
signed short sin_cut;                       /*An unit value to make a SIN pointer */


void main_pro()
{

/*Unit value of sin 23040 X output frequency ÷ carrier frequency   23040=360°×64     */
            sin_cut=(signed short)(((signed long)out_bin*23040)/CARR);

            tq_dat=(signed short)(((signed long)torq*carr_set_4)/1000);        /*torque order value X carreir/4 */

}


/*********************************************************************************/
/*                                                                            */
/*                              base timer interrupt process                  */
/*                                                                            */
/*********************************************************************************/
void iic_int(void);

void pwm_uvw_set(void);
void i_con(void);
void angle(void);

signed short tq_dat;                /*torque order value X carrier/4*/
signed short sinpt_sum;             /*SIN pointer sum total counter */
signed short sin_pt;                /*SIN table pointer */
signed short pwm_u_w;               /*U-phase PWM order value*/
signed short pwm_v_w;               /*V-phase PWM order value */
signed short pwm_w_w;               /*W-phase PWM order value */
signed int pwm_u_old_w;             /*U-phase PWM previous order value */
signed int pwm_v_old_w;             /*V-phase PWM previous order value */
signed int pwm_w_old_w;             /*W-phase PWM previous order value */
```

```
/*          base timer interrupt               */

#pragma INTERRUPT/B    iic_int
void iic_int(void)
{

            iio6ir=0x00;                                   /*interrupt request clear*/

            angle();                          /*Generate SIN table pointer    */
            i_con();                          /*Calculate PWM order value                */
            pwm_uvw_set();                    /*PWM order value upper limit compensation→timer setting */
}




/*******************************************************************************/
/*                                                                             */
/*       Example of base timer interrupt operation module (in case of a guidance motor) */
/*             Calculate a SIN angle                                           */
/*              Calculate a PWM duty                                           */
/*                                                                             */
/*******************************************************************************/



void angle()
{
   sinpt_sum=sin_cut+sinpt_sum;               /*sin pointer sum total←sin unit value + SIN pointer sum total */
            if(sinpt_sum>23040)               /*SIN pointer sum total is maximum? 23040=360°×64*/
            {
                        sinpt_sum=sinpt_sum-23040;          /*SIN pointer sum total upper limit compensation */
            }
            else
            {
            }
            sin_pt=sinpt_sum>>6;                                    /*SIN pointer ←SIN pointer sum total÷64*/
}


void i_con()
{
/*    U-phase pwm order value= carrier/4-(sinN°×(torque order value×carrier/4))                  */
            pwm_u_w=carr_set_4-(signed   short)(((signed   long)sin_tbl[sin_pt]*(signed   long)(tq_dat*2))>>16);


/*    V-phase pwm order value=carrier/4-(sin(N+120) °×(torque order value×carrier 4))            */
            pwm_v_w=carr_set_4-(signed short)(((signed long)sin_tbl[sin_pt+120]*(signed long)(tq_dat*2))>>16);


/*    W-phase pwm order value = carrier /4-(sin(N+240) °×(torque order value×carrier /4))                 */
            pwm_w_w=carr_set_4-(signed               short)(((signed              long)sin_tbl[sin_pt+240]*(signed
long)(tq_dat*2))>>16);
}
```

```
/*****************************************************************************/
/*                                                                           */
/*     Module for base timer interrupt                                       */
/*            PWM data setting                                               */
/*                                                                           */
/*****************************************************************************/


void pwm_uvw_set(void)          /**********With a shoting prevention time ************/
{

/*U-phase PWM compensation*/

        if((carr_set_2-dtt_set)<pwm_u_w)                        /*Duty maximum?*/
        {
                g2po2=carr_set;         /*Group 2 waveform generation register*/
                g2po3=0;                /*Group 2 waveform generation register */

                if(carr_set_2<pwm_u_w)
                {
                        g3po2=carr_set;         /* Group 3 waveform generation register */
                        g3po3=0;                /* Group 3 waveform generation register */
                }
                else
                {
                        g3po2=pwm_u_w;                  /* Group 3 waveform generation register */
                        g3po3=carr_set+dtt_set-pwm_u_w; /* Group 3 waveform generation register */
                }
        }
        else
        {
                if(dtt_set>pwm_u_w)                     /*Duty minimum?*/
                {
                        if(0>pwm_u_w)
                        {
                                if(pwm_u_old_w<0)       /*previous data is also<0?*/
                                {
                                        g2po2=0;        /* Group 2 waveform generation register */
                                }
                                else
                                {
                                        g2po2=dtt_set;  /*Group 2 waveform generation register */
                                }
                                g2po3=carr_set;  /* Group 2 waveform generation register */
                        }
                        else
                        {
                                g2po2=pwm_u_w+dtt_set;  /* Group 2 waveform generation register */
                                g2po3=carr_set-pwm_u_w; /* Group 2 waveform generation register */
                        }

                        g3po2=0;                /* Group 3 waveform generation register */
                        g3po3=carr_set;         /* Group 3 waveform generation register */



        }
                else                                    /*minimum<duty<maximum*/
```

```
                {
                        g2po2=pwm_u_w+dtt_set;          /* Group 2 waveform generation register */
                        g2po3=carr_set-pwm_u_w;         /* Group 2 waveform generation register */

                        if(pwm_u_old_w<0)               /*previous data <0?*/
                        {
                                g3po2=0;                /* Group 3 waveform generation register */
                        }
                        else
                        {
                                g3po2=pwm_u_w;          /* Group 3 waveform generation register */
                        }
                        g3po3=carr_set+dtt_set-pwm_u_w; /* Group 3 waveform generation register */
                }
        }
        pwm_u_old_w=pwm_u_w;                            /*previous data update*/


/*V-phase PWM revision */

        if((carr_set_2-dtt_set)<pwm_v_w)                                /*Duty maximum?*/
        {
                g2po4=carr_set;                 /* Group 2 waveform generation register */
                g2po5=0;                        /* Group 2 waveform generation register */

                if(carr_set_2<pwm_v_w)
                {
                        g3po4=carr_set;                 /* Group 3 waveform generation register */
                        g3po5=0;                        /* Group 3 waveform generation register */
                }
                else
                {
                        g3po4=pwm_v_w;                  /* Group 3 waveform generation register */
                        g3po5=carr_set+dtt_set-pwm_v_w; /* Group 3 waveform generation register */
                }
        }
        else
        {
                if(dtt_set>pwm_v_w)                     /*Duty minimum?*/
                {
                        if(0>pwm_v_w)
                        {
                                if(pwm_v_old_w<0)       /*previous data is also<0?*/
                                {
                                        g2po4=0;        /* Group 2 waveform generation register */
                                }
                                else
                                {
                                        g2po4=dtt_set;  /* Group 2 waveform generation register */
                                }
                                g2po5=carr_set;    /* Group 2 waveform generation register */
                        }
                        else

{
                                g2po4=pwm_v_w+dtt_set;  /* Group 2 waveform generation register */
                                g2po5=carr_set-pwm_v_w; /* Group 2 waveform generation register */
                        }

                        g3po4=0;                /* Group 3 waveform generation register */
```

```
                                g3po5=carr_set;              /* Group 3 waveform generation register */

                        }
                        else                                        /*minimum<duty<maximum*/
                        {
                                g2po4=pwm_v_w+dtt_set;              /* Group 2 waveform generation register */
                                g2po5=carr_set-pwm_v_w;             /* Group 2 waveform generation register */

                                if(pwm_v_old_w<0)                   /*previous data <0?*/
                                {
                                        g3po4=0;                    /* Group 3 waveform generation register */
                                }
                                else
                                {
                                        g3po4=pwm_v_w;              /* Group 3 waveform generation register */
                                }
                                g3po5=carr_set+dtt_set-pwm_v_w;     /* Group 3 waveform generation register */
                        }
                }
                pwm_v_old_w=pwm_v_w;                                /*previous data update*/


        /*W-phase PWM compensation*/

                if((carr_set_2-dtt_set)<pwm_w_w)                            /*Duty maximum?*/
                {
                        g2po6=carr_set;              /* Group 2 waveform generation register */
                        g2po7=0;                     /* Group 2 waveform generation register */

                        if(carr_set_2<pwm_w_w)
                        {
                                g3po6=carr_set;              /* Group 3 waveform generation register */
                                g3po7=0;                     /* Group 3 waveform generation register */
                        }
                        else
                        {
                                g3po6=pwm_w_w;                      /* Group 3 waveform generation register */
                                g3po7=carr_set+dtt_set-pwm_w_w;     /* Group 3 waveform generation register */
                        }
                }
                else
                {
                        if(dtt_set>pwm_w_w)                         /*Duty minimum?*/
                        {
                                if(0>pwm_w_w)
                                {



                if(pwm_w_old_w<0)           /*previous data is also<0?*/
                                        {
                                                g2po6=0;            /* Group 2 waveform generation register */
                                        }
                                        else
                                        {
                                                g2po6=dtt_set;      /* Group 2 waveform generation register */
                                        }
                                        g2po7=carr_set;     /* Group 2 waveform generation register */
                                }
```

```
                    else
                    {
                              g2po6=pwm_w_w+dtt_set;  /* Group 2 waveform generation register */
                              g2po7=carr_set-pwm_w_w; /* Group 2 waveform generation register */
                    }

                    g3po6=0;                /* Group 3 waveform generation register */
                    g3po7=carr_set;         /* Group 3 waveform generation register */

          }
          else                                      /*minimum<duty<maximum*/
          {
                    g2po6=pwm_w_w+dtt_set;          /* Group 2 waveform generation register */
                    g2po7=carr_set-pwm_w_w;         /* Group 2 waveform generation register */

                    if(pwm_w_old_w<0)               /*previous data<0?*/
                    {
                              g3po6=0;              /* Group 3 waveform generation register */
                    }
                    else
                    {
                              g3po6=pwm_w_w;        /* Group 3 waveform generation register */
                    }
                    g3po7=carr_set+dtt_set-pwm_w_w; /* Group 3 waveform generation register */
          }
     }
     pwm_w_old_w=pwm_w_w;                           /*previous data update*/
}
```

7.2 120° Energization Reference Program

The 120° enegization output example will be shown below. This example uses Group 2.

```
/*********************************************************************************
 *          120° energization reference program
 *

 *********************************************************************************/
/*-----------------------------------------------------------------------------*
/*    Copyright (C) 2003 Renesas Technology Corporation           *
/*    Copyright (C) 2003 Renesas Solutions Corporation            *
/*    All rights reserved.                                        *
/******************************************************************************/

/*********************************************************************************/
/*                                                                               */
/*     SFR setting                                                               */
/*                                                                               */
/*********************************************************************************/

volatile char      int1ic;
#pragma ADDRESS    int1ic       007Eh    /* INT1 interrupt control register */
volatile char      int2ic;
#pragma ADDRESS    int2ic       009Ch    /* INT2 interrupt control register */
volatile char      int0ic;
#pragma ADDRESS    int0ic       009Eh    /* INT0 interrupt control register */
volatile char      btsr;
#pragma ADDRESS    btsr         0164h    /* base timer start register */
volatile int       g2po0;
#pragma ADDRESS    g2po0        0140h    /* Group 2 waveform generation register 0 */
volatile int       g2po1;
#pragma ADDRESS    g2po1        0142h    /* Group 2 waveform generation register 1 */
volatile int       g2po2;
#pragma ADDRESS    g2po2        0144h    /* Group 2 waveform generation register 2 */
volatile int       g2po3;
#pragma ADDRESS    g2po3        0146h    /* Group 2 waveform generation register 3 */
volatile int       g2po4;
#pragma ADDRESS    g2po4        0148h    /* Group 2 waveform generation register 4 */
volatile int       g2po5;
#pragma ADDRESS    g2po5        014Ah    /* Group 2 waveform generation register 5 */
volatile int       g2po6;
#pragma ADDRESS    g2po6        014Ch    /* Group 2 waveform generation register 6 */
volatile int       g2po7;
#pragma ADDRESS    g2po7        014Eh    /* Group 2 waveform generation register 7 */
volatile char      g2pocr0;
#pragma ADDRESS    g2pocr0      0150h    /* Group 2 waveform generation control register 0 */
volatile char      g2pocr1;
#pragma ADDRESS    g2pocr1      0151h    /* Group 2 waveform generation control register 1 */
volatile char      g2pocr2;
#pragma ADDRESS    g2pocr2      0152h    /* Group 2 waveform generation control register 2 */
volatile char      g2pocr3;
#pragma ADDRESS    g2pocr3      0153h    /* Group 2 waveform generation control register 3 */
```

```
volatile char        g2pocr4;
#pragma ADDRESS      g2pocr4          0154h     /* Group 2 waveform generation control register 4 */
volatile char        g2pocr5;
#pragma ADDRESS      g2pocr5          0155h     /* Group 2 waveform generation control register 5 */
volatile char        g2pocr6;
#pragma ADDRESS      g2pocr6          0156h     /* Group 2 waveform generation control register 6 */
volatile char        g2pocr7;
#pragma ADDRESS      g2pocr7          0157h     /* Group 2 waveform generation control register 7 */
volatile char        g2bcr0;
#pragma ADDRESS      g2bcr0           0162h       /* Group 2 base timer control register 0 */
volatile char        g2bcr1;
#pragma ADDRESS      g2bcr1           0163h       /* Group 2 base timer control register 1 */
volatile char        g2fe;
#pragma ADDRESS      g2fe             0166h       /* Group 2 function enable flag */
volatile char        ifsr;
#pragma ADDRESS      ifsr             031Fh     /* External interrupt request cause select register */
volatile char        ps7;
#pragma ADDRESS      ps7              03BDh       /* Function select register A7 */
volatile char        p13;
#pragma ADDRESS      p13              03CDh       /* port 13                 */
volatile char        pd13;
#pragma ADDRESS      pd13             03CFh       /* port 13 direction register */


/*******************************************************************************/
/*                                                                           */
/*      Initialization                                                       */
/*                                                                           */
/*******************************************************************************/
void main_ini(void);

#define CLK 30000000                    /*Frequency of microcontroller Hz*/
#define CARR 20000                      /*carrier frequency Hz*/

#define carr_set (CLK/CARR)         /*carrier frequency                 */


void main_ini()
{
        p13=0xaf;                   /*port 13 register   non-active*/
        pd13=0xaf;                  /*port 13 direction regiser */

        g2bcr0=0x7f;        /*Group 2 base timer control register 0*/
        btsr=0x00;                  /*Group 2,3 base timer stop */
        g2bcr1=0x02;        /*Group 2 base timer control register 1*/

        g2pocr0=0x00;       /*Base timer reset factor */
        g2pocr2=0x00;       /*Single-phase waveform output */
        g2pocr3=0x00;       /*Single-phase waveform output */
        g2pocr4=0x00;       /*Single-phase waveform output   */
        g2pocr5=0x00;       /*Single-phase waveform output   */
```

```
        g2pocr6=0x00;      /*Single-phase waveform output */
        g2pocr7=0x00;      /*Single-phase waveform output */

        g2po0=carr_set-2; /*Group 2 waveform generation register */
        g2po2=1;           /*Group 2 waveform generation register */
        g2po3=1;           /* Group 2 waveform generation register */
        g2po4=1;           /* Group 2 waveform generation register */
        g2po5=1;           /* Group 2 waveform generation register */
        g2po6=1;           /* Group 2 waveform generation register */
        g2po7=1;           /* Group 2 waveform generation register */

        g2pocr0=0x00;      /*Base timer reset factor */
        g2pocr2=0x20;      /* Single-phase waveform output reload base*/
        g2pocr3=0x20;      /* Single-phase waveform output reload base */
        g2pocr4=0x20;      /* Single-phase waveform output reload base */
        g2pocr5=0x20;      /* Single-phase waveform output reload base */
        g2pocr6=0x20;      /* Single-phase waveform output reload base */
        g2pocr7=0x20;      /* Single-phase waveform output reload base */

        btsr=0x04;                    /*Group 2 base timer start */
        g2fe=0xfd;                    /*Group 2 enabled */

        ps7=0xaf;


        /*Using INT for positional input */
        ifsr=0x07;                    /*INT0,1,2 both edges*/
        int0ic=7;          /*INT0,1,2 interrupt level selection*/
        int1ic=7;
        int2ic=7;

        asm("     FSET      I");      /* interrupt enable */


}



/*******************************************************************************/
/*                                                                            */
/*      Operation in main process                                             */
/*     1.The setting of the torque instruction value (a PWM duty cycle) */
/*                        corresponding to the speed                          */
/*******************************************************************************/
void dc_pwm_set(void);

unsigned int        tk_dat_w;                    /*torque order value*/

void dc_pwm_set(void)
{
        g2po2=g2po3=g2po4=g2po5=g2po6=g2po7=tk_dat_w;
}
```

```
/****************************************************************************/
/*                                                                          */
/*          Angle detection interrupt (INT use)                             */
/*        Change active phase of each phase                                 */
/*                  by function select register within INT interrupt        */
/*                                                                          */
/*                                                                          */
/****************************************************************************/


#pragma INTERRUPT/B   int0_int
void int0_int(void);
#pragma INTERRUPT/B   int1_int
void int1_int(void);
#pragma INTERRUPT/B   int2_int
void int2_int(void);
void rad_to_pwm(void);


unsigned char     ps7b=0x00;                          /*Next output phase*/
signed    int             rad_datw=0;             /*Input angle*/
struct tag{
                  char     bit0:1;
                  char     bit1:1;
                  char     bit2:1;
                  char     bit3:1;

        }dc_buf;
#define edge0_flg dc_buf.bit1
#define edge1_flg dc_buf.bit2
#define edge2_flg dc_buf.bit3



void int0_int(void)
{
        ps7=ps7b;                   /*Output phase change*/

        if(edge0_flg==0)
        {
                edge0_flg=1;
                rad_datw=60;        /*Next angle setting*/
        }
        else
        {
                edge0_flg=0;
                rad_datw=240;       /*Next angle setting*/
        }

        rad_to_pwm();               /*Next output phase setting*/

}
```

```
void int1_int(void)
{
        ps7=ps7b;                       /*Output phase change*/

        if(edge1_flg==0)
        {
                edge1_flg=1;
                rad_datw=180;                   /*Next angle setting*/
        }
        else
        {
                edge1_flg=0;
                rad_datw=0;                     /*Next angle setting*/
        }

        rad_to_pwm();                           /*Next output phase setting*/
}


void int2_int(void)
{
        ps7=ps7b;                       /*Output phase change*/

        if(edge2_flg==0)
        {
                edge2_flg=1;
                rad_datw=300;                   /*Next angle setting*/
                rad_to_pwm();
        }
        else
        {
                edge2_flg=0;
                rad_datw=120;                   /*Next angle setting*/
        }

        rad_to_pwm();                           /*Next output phase setting*/
}

void rad_to_pwm(void)
{
        switch(rad_datw)
        {
                case 0:
                        ps7b=0xa0;      /*Next output phase setting U                WB */
                        break;
                case 60:
                        ps7b=0x88;      /* Next output phase setting   V        WB */
                        break;
                case 120:
                        ps7b=0x0a;      /* Next output phase setting   V   UB      */
                        break;
```

```
        case 180:
                ps7b=0x03;      /* Next output phase setting    W UB      */
                break;
        case 240:
                ps7b=0x05;      /* Next output phase setting    W     VB          */
                break;
        case 300:
                ps7b=0x24;      /* Next output phase setting U        VB   */
                break;
        default:
                ps7b=0x00;      /* Next output phase setting off          */
                break;
    }
}
```

5.0 Reference

Datasheet

Refer to M32C/83 Group datasheet

(Acquire the most current version from Renesas web-site)

6.0 Web-site and contact for support

Renesas Web-site

http://www.renesas.com

Contact for Renesas technical support

Mail to : support_apl@renesas.com

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	Nov 28, 2003	-	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.