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.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

M32C/83 Group

Concept of the Three-phase Motor Control Program

REJ05B0147-0120Z
Rev.1.20
Jul 25, 2003

1. Abstract

This application note describes the how to use three-phase motor control timer function, and application example.

2. Introduction

The explanation of this issue is applied to the following condition:

Applicable MCU: M32C/83 Group

3. 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.

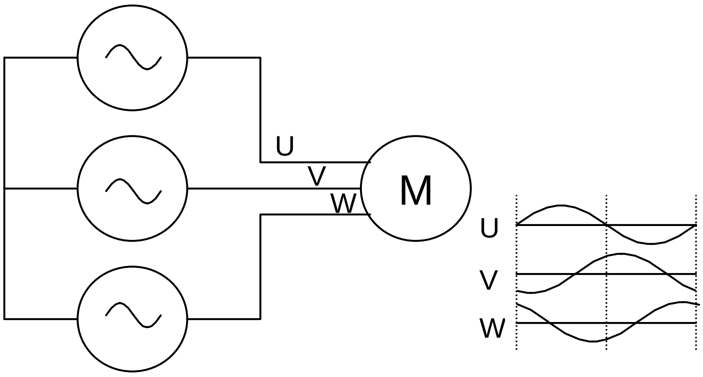


Figure 3.1 Three-phase Motor Drive

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.

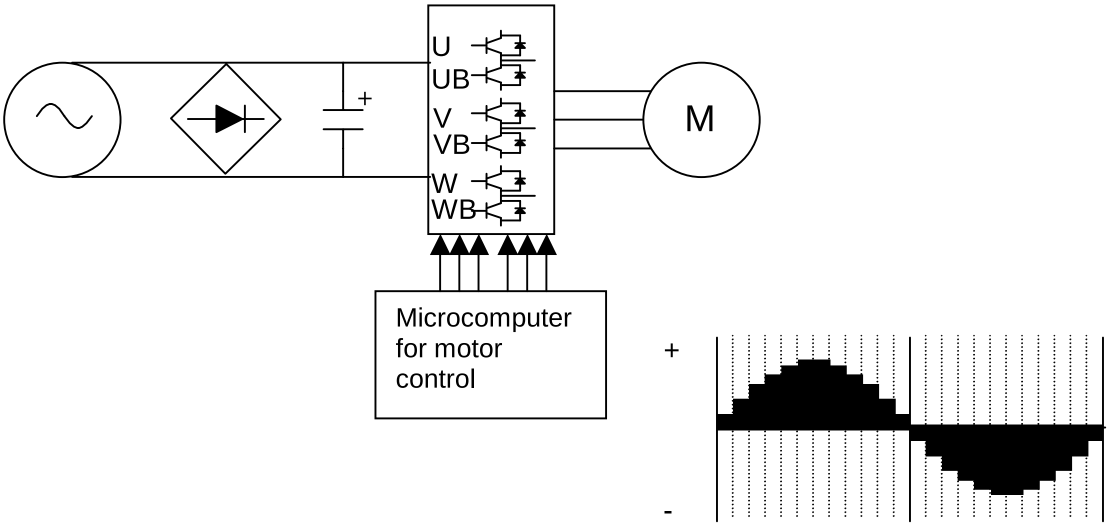


Figure 3.2 Example of Inverter Control Using a Microcomputer

3.2 Waveforms 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, W, UB, VB and WB in this diagram accept as input the signals output from the microcomputer pins.

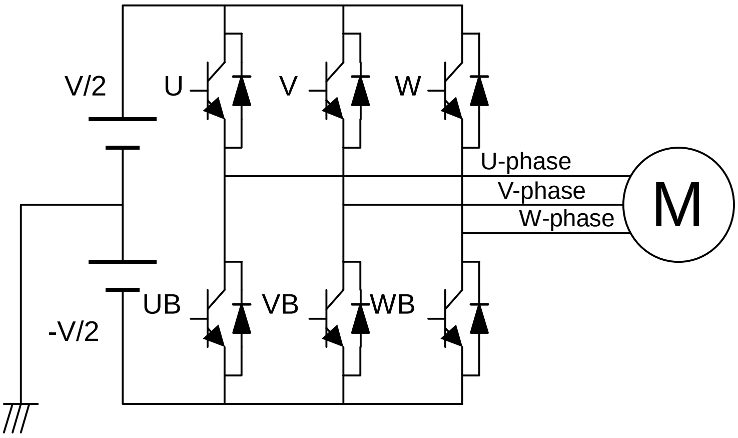


Figure 3.3 Power Transistor Circuit

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.

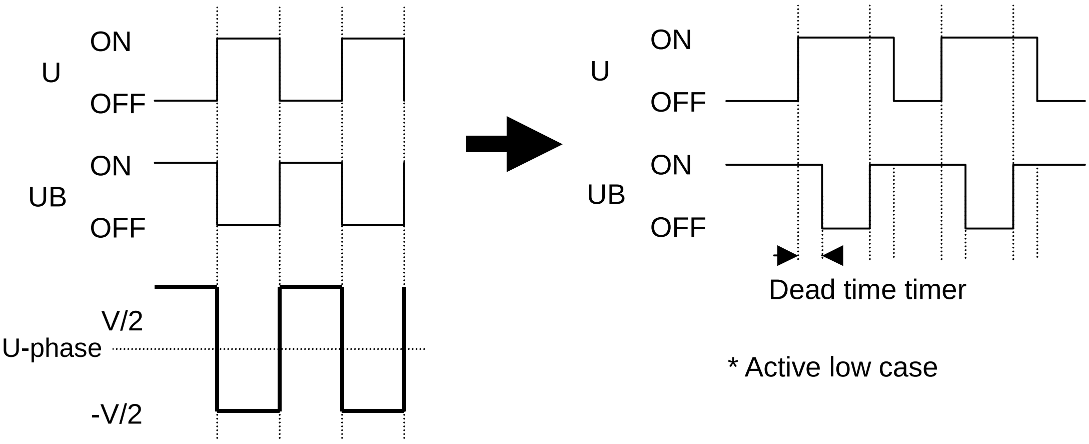
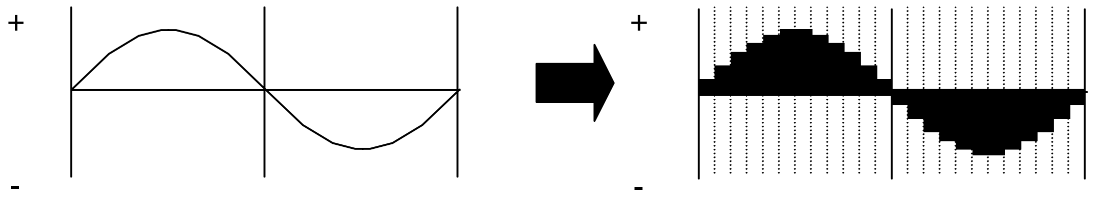


Figure 3.4 Microcomputer Output Waveform and Generated Waveform

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 current flows, causing the d.c. power supply to be shorted. To solve this problem, the timers used for three-phase motor control produce differential switchover timing to prevent simultaneous turn-on. This differential time is referred to as the shorting prevention time. By only setting a value in the shorting prevention timer during initialize processing in a program, it is possible to produce a shorting-free output waveform.

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. sine wave. Therefore, 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.

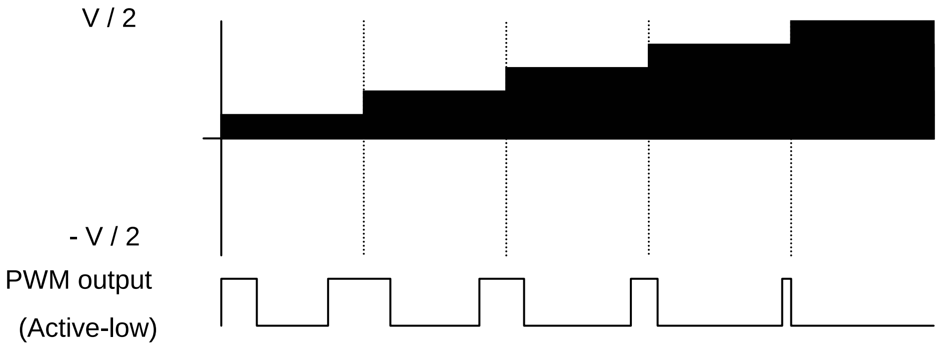


Figure 3.5 Changing an a.c. sine wave to a square wave by dividing it in time

4. Using Three-phase Motor Control Timer Functions

4.1 How to Output a Three-phase Waveform

This section describes the basic method for producing three-phase output waveform.

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.

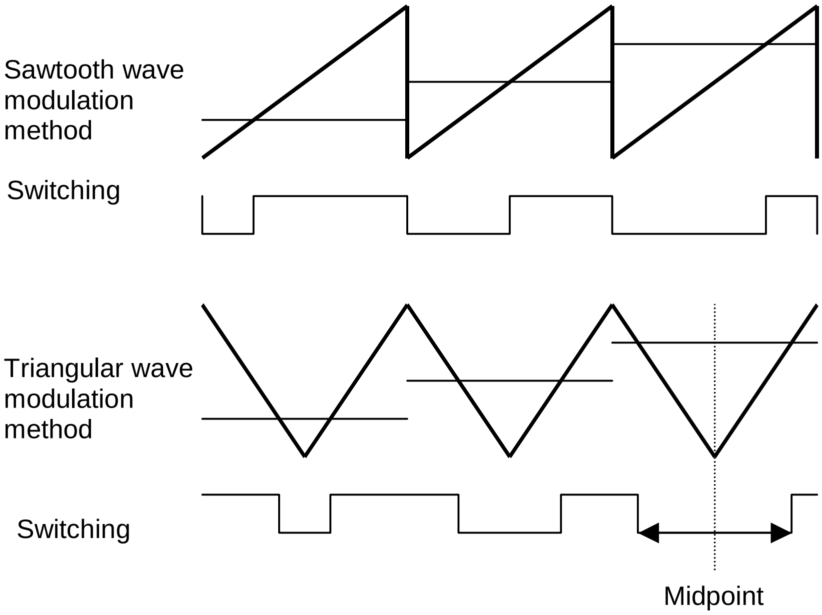

Sawtooth wave modulation method
Switching
Triangular wave modulation method
Switching
Midpoint

Figure 4.1 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

The three-phase motor control timer functions are thought of with respect to one carrier cycle for the sawtooth wave modulation and a 1/2 carrier cycle for the triangular wave modulation.

The carrier cycle peak timing is generated by TB2. An underflow trigger of this TB2 activates one-shot mode of TAi (i = 4, 1, 2). This TAi determines the PWM duty cycle.

Selection of modulation method between sawtooth wave and triangular wave modulation is determined by whether the shorting prevention time is triggered by the rising or the falling transition of TAi.

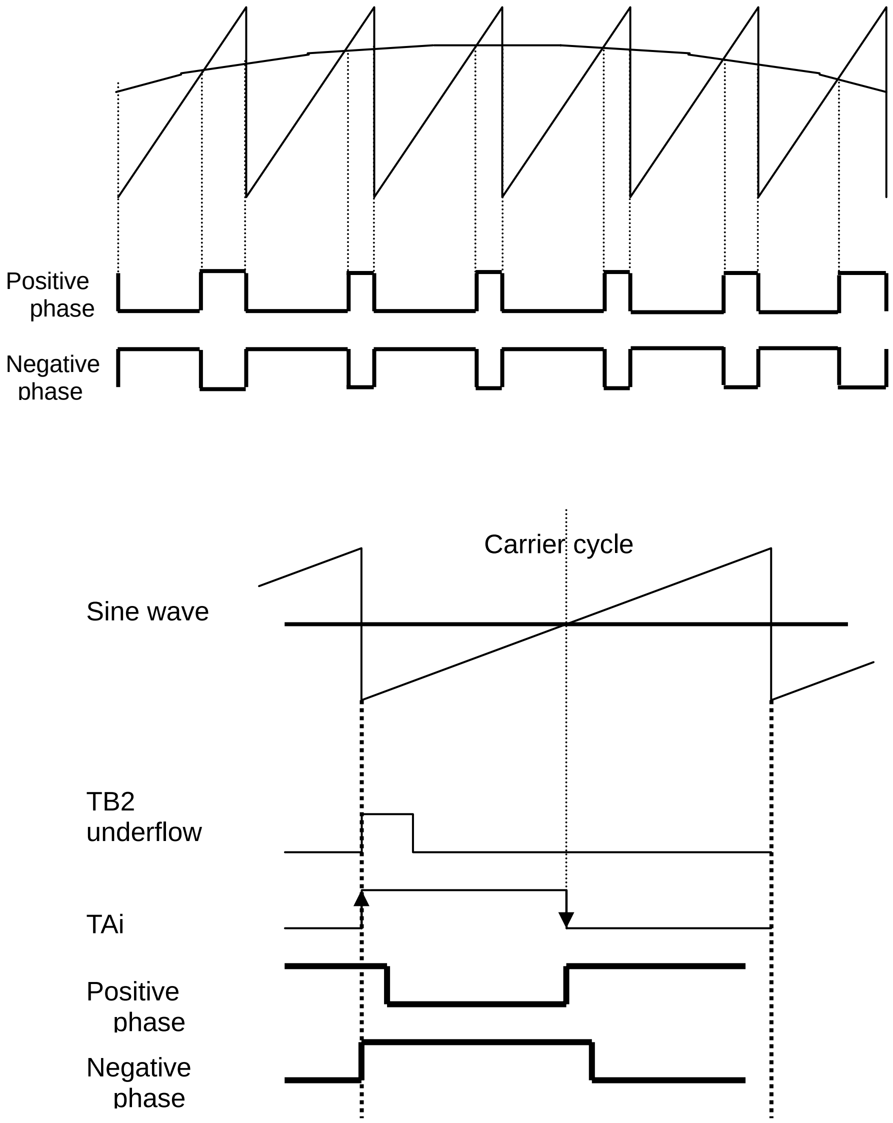
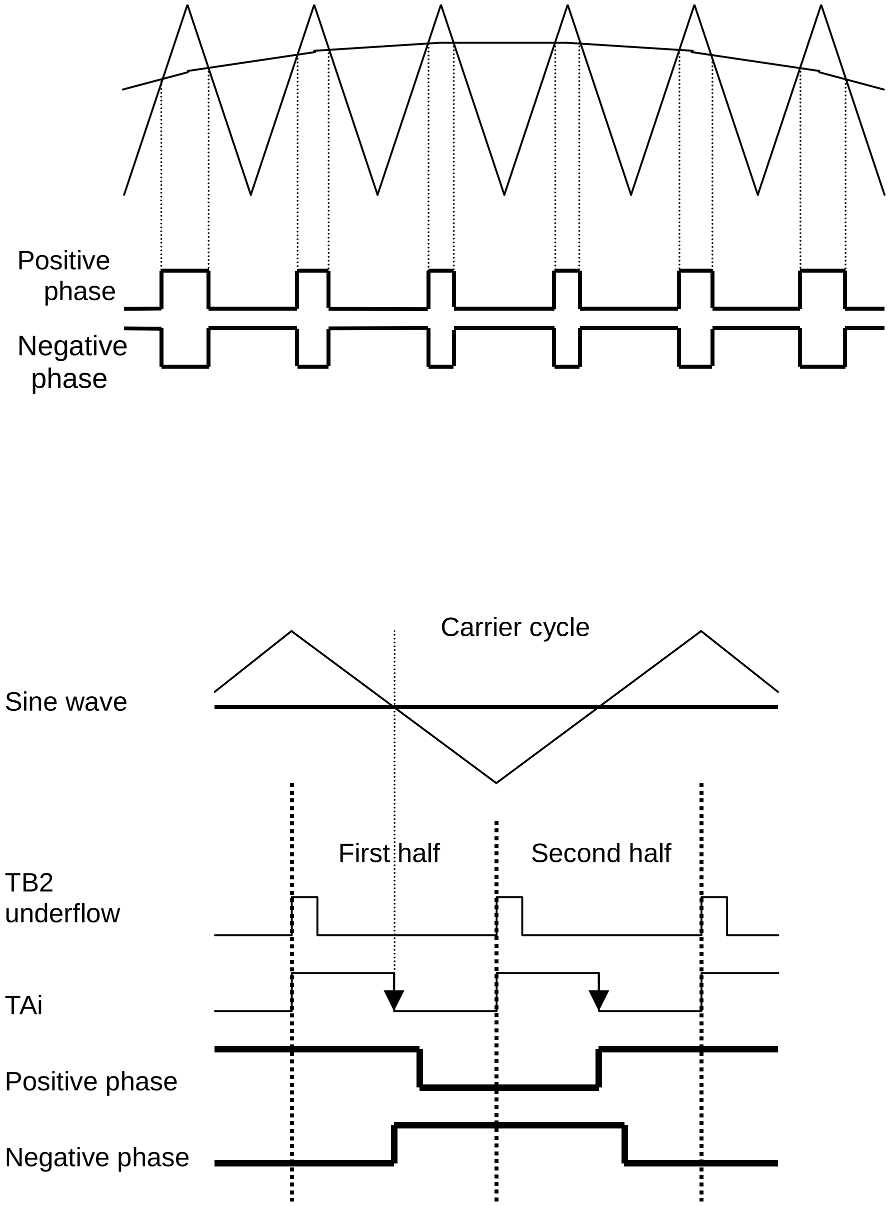


Figure 4.2 Relationship between Timer and PWM during Sawtooth Wave Modulation



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 Relationship between Timer and PWM during Triangular Wave Modulation

If a triangular wave modulated waveform needs to be output using the method described above, it is necessary to generate timing and set TAi back again every 1/2 carrier cycle, e.g., by means of an interrupt.

To reduce this software (interrupt) burden, the timers have a function to set the first and second half TAi values in one carrier cycle. This function is known as three-phase mode 1.

In three-phase mode 1, the values to be set in TAi are set alternately from two registers.

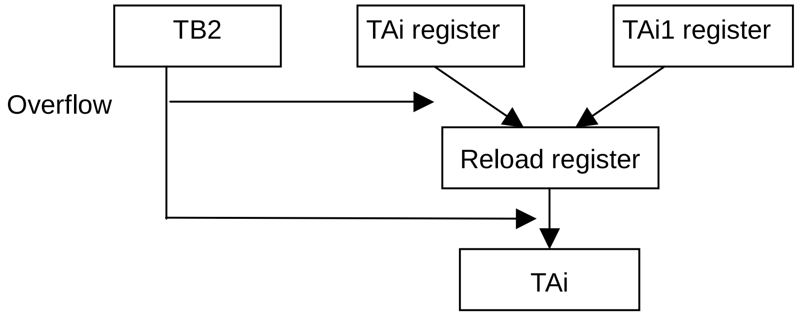
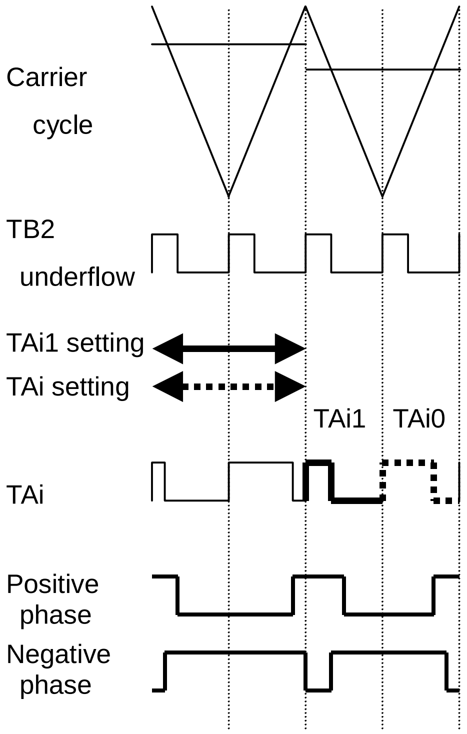
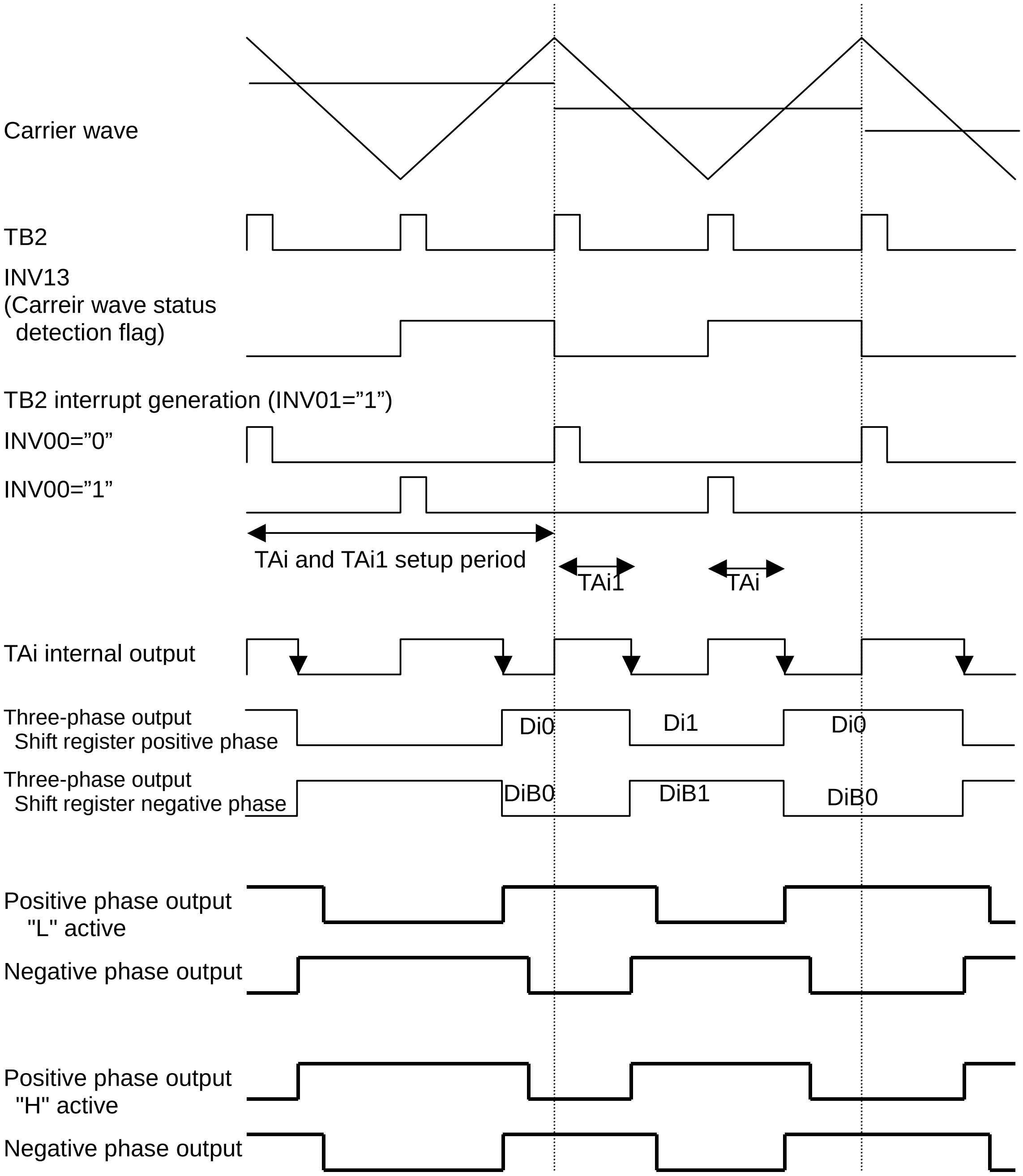


Figure 4.4 Three-phase Mode 1 Setting and Output Timing



* The TAi1 register cannot be rewritten at any point in time near a TB2 underflow. For details, see the user’s manual.

Figure 4.5 Three-phase Mode 1



* If the initial value (INV01 = 0) is specified for interrupt enable output and the interrupt frequency setting counter (ICTB2) is set to "2," the TB2 interrupt in the initial state is generated with the same timing as when INV00 = 1 in the above diagram. If the interrupt needs to be generated with the same timing as when INV00 = 0, set an odd number in the interrupt frequency setting counter (ICTB2) first time only.

Figure 4.6 Relationship between Three-phase Mode 1 and Other Settings
(invc0=*0**11**B; idb0=00010101B; idb1=00101010B)

4.4 Precautions about TAi Settings

When setting values in the TAi and TAi1 registers, pay attention to the following.

(1) Do not set data “0” in the TAi register

If data “0” is set in the TAi register (if the shorting prevention timer count source = f2, data “0,” “1”), the TAi timer neither counts nor produces a falling edge and hence the output waveform retains the current level intact. Therefore, do not set data “0” in the TAi register unless so intended.

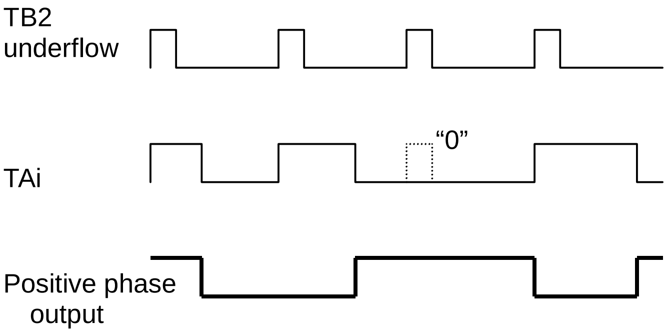


Figure 4.7 Positive Phase Waveform when TAi = 0

(2) Do not set any value in the TAi register that is larger than the TB2 set value.

If any value larger than the TB2 set value is set in the TAi register (if the shorting prevention timer count source = f2, a value larger than the TB2 set value – 1), the TAi timer stops after counting within the TB2 cycle, without producing a falling edge and, hence, the output waveform remains held at the current level. As a result, the waveform is inverted as shown below. Therefore, do not set any value in the TAi register that is larger than the TB2 set value unless so intended.

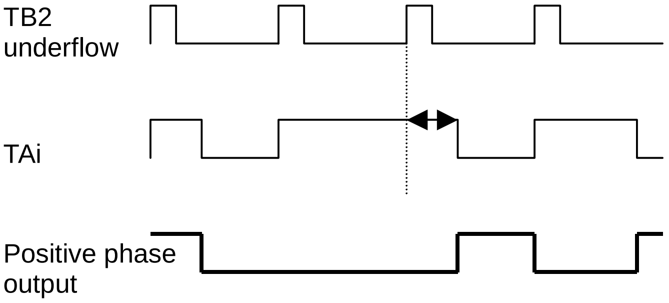


Figure 4.8 Output Waveform when TAi > TB2

(3) Restarting the shorting prevention timer

If an event occurs that causes the shorting prevention timer to start while the shorting prevention timer is counting for reasons of the set TAi data, the shorting prevention timer cannot be restarted.

If the conditions below hold true

For triangular wave modulation mode: Shorting prevention timer count source = f1
((TB2 set value + 1) – Tai1 set value) + TAi set value < shorting prevention timer set value
TAi1 set value + ((TB2 set value + 1) – TAi set value) < shorting prevention timer set time

For sawtooth wave modulation mode: Shorting prevention timer count source = f1
((TB2 set value + 1) – TAi set value) – 1 < shorting prevention timer set value
TAi set value – 1 < shorting prevention timer set value

Be aware that the shorting prevention timer cannot be restarted.

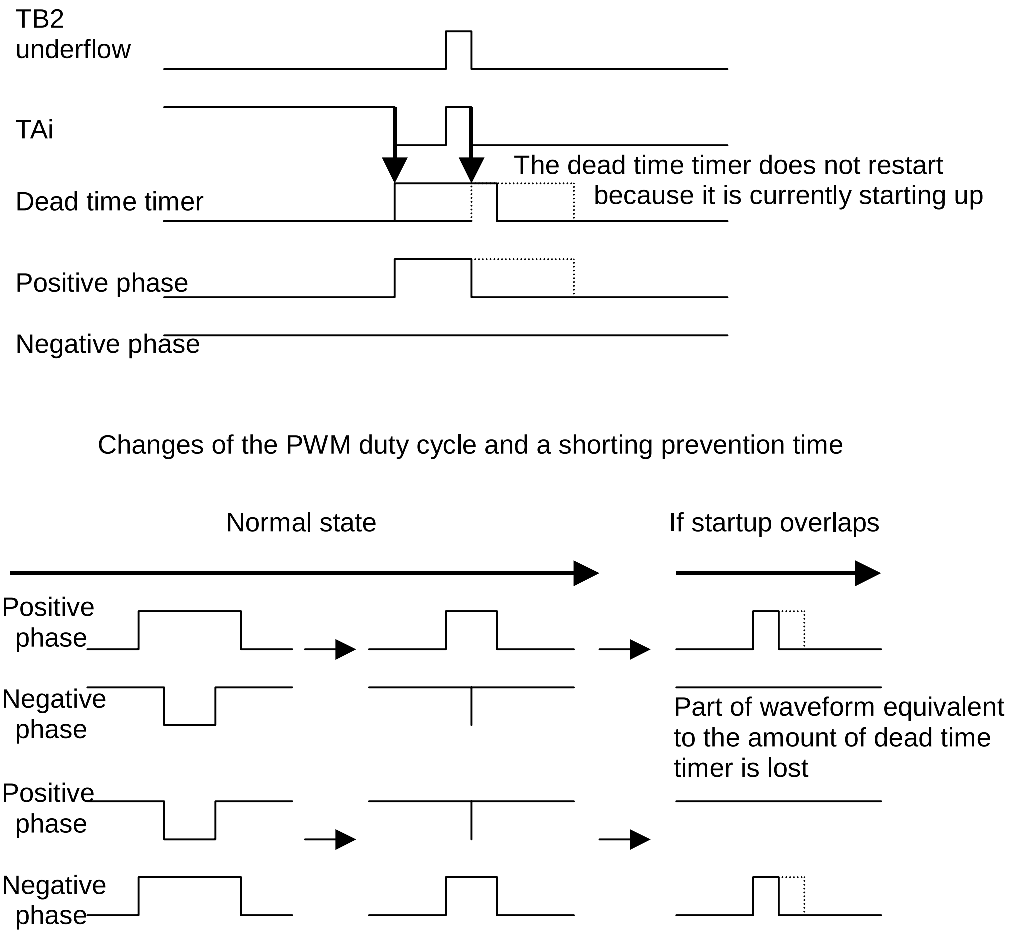

TB2
underflow
TAi
Dead time timer
The dead time timer does not restart
because it is currently starting up
Positive phase
Negative phase
Changes of the PWM duty cycle and a shorting prevention time
Normal state
If startup overlaps
Positive
phase
Negative
phase
Part of waveform equivalent
to the amount of dead time
timer is lost
Positive
phase
Negative
phase

Figure 4.9 Restarting of the Shorting Prevention Timer

4.5 Altering the Carrier Frequency

To alter the carrier frequency, change the TB2 interrupt cycle. For triangular wave modulation, however, caution must be used because two TB2 underflows comprise one carrier cycle.

For control to be made every carrier cycle using triangular wave modulation (invc0 = *0******B and inv0 = 0*****1*B), set the TB2 to be reloaded at odd-numbered occurrences of timer A output (tb2sc = 00000001B)

For control to be made every 1/2 carrier cycle using triangular wave modulation (invc0 = *0******B and inv0 = 1*****1*B or invc0 = ********B and invc0 = ******0*B), set the TB2 to be reloaded at a TB2 underflow (tb2sc = 00000000B).

That way, alteration of the carrier cycle can be timed to occur in synchronism with TAi settings.

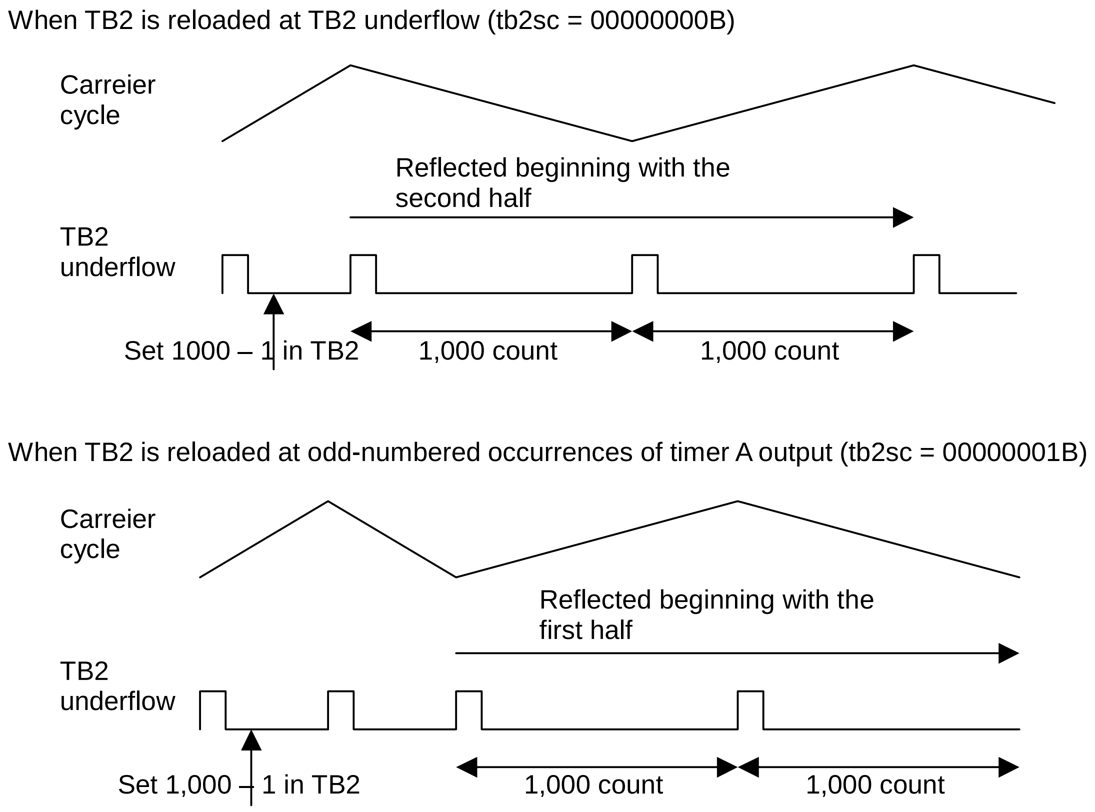

When TB2 is reloaded at TB2 underflow (tb2sc = 00000000B)
Carreier cycle
Reflected beginning with the second half
TB2 underflow
Set 1000 – 1 in TB2
1,000 count
1,000 count
When TB2 is reloaded at odd-numbered occurrences of timer A output (tb2sc = 00000001B)
Carreier cycle
Reflected beginning with the first half
TB2 underflow
Set 1,000 – 1 in TB2
1,000 count
1,000 count

Figure 4.10 Carrier Cycle Settings and Refection

4.6 Using Three-phase Output Buffer Register

By rewriting the value of the three-phase output buffer register, it is possible to fix the output level (continuously on or continuously off) for a given period or produce an output waveform whose positive and negative phases are at the same level.

This function allows producing two-phase modulation output (see Section 5) or 120-degree turn-on output (see Section 6).

* The content of the three-phase output buffer register is transferred to the shift register at the first TB2 underflow after being rewritten.

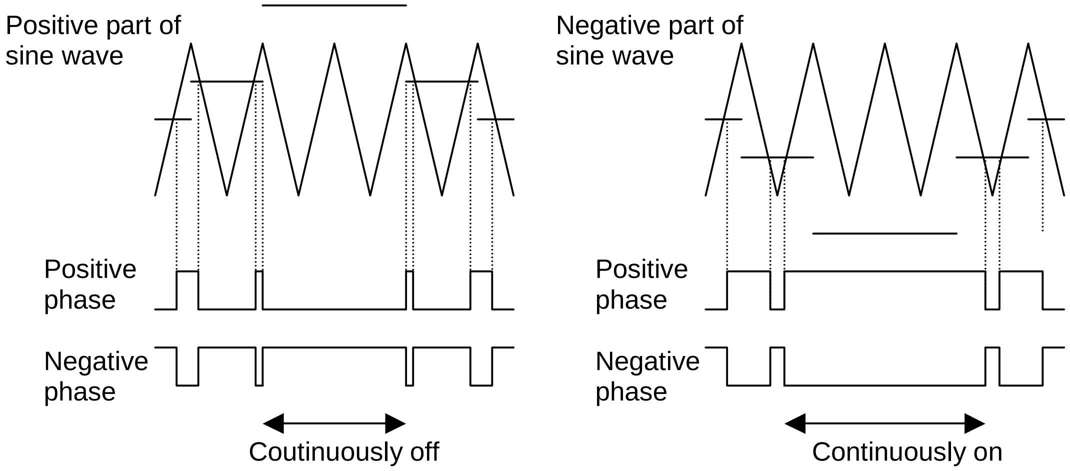


Figure 4.11 Conceptual Waveform of Two-phase Modulation Output (see Section 5 for details)

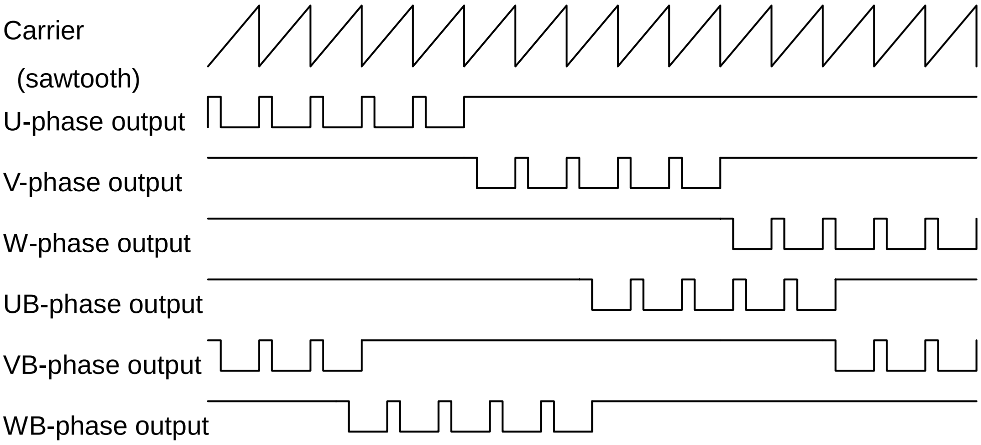


Figure 4.12 Conceptual Waveform of 120-degree Turn-on Output (see Section 6 for details)

5. 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 the three-phase motor control timer functions.

5.2 Using the Three-phase Motor Control Timer Functions

Of the three-phase motor control timer functions, select the triangular wave modulation method and three-phase mode 1.

5.3 Calculating PWM Data

The following shows how to calculate the values to be set in TAi.

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

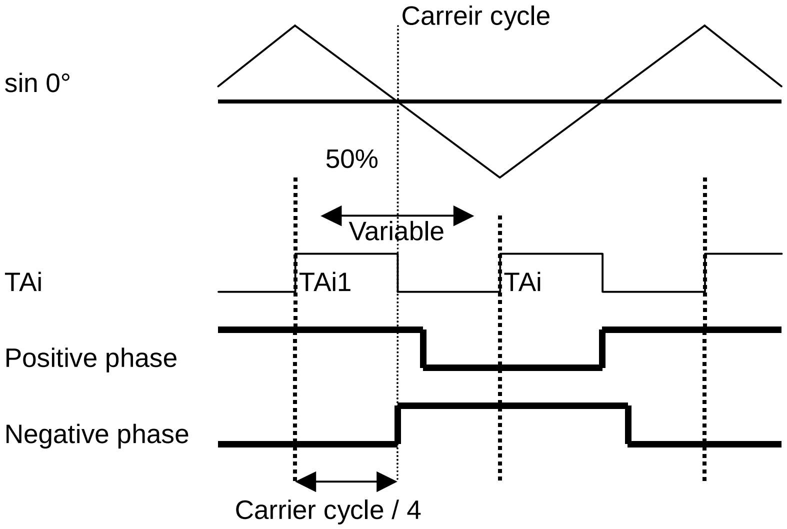


Figure 5.1 Relationship between TAi and PWM Data

Because 50% duty cycle = carrier cycle / 4,

TAi1 data = carrier cycle / 4 – variable value.

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

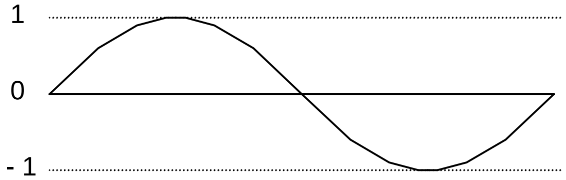


Figure 5.2 Sine Wave

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

Variable value = 50% x sinN°

= Carrier cycle / 4 x sinN°

Thus, the width setting values can be obtained from the equations below

TAi1 data = carrier cycle / 4 – carrier cycle / 4 x sinN°

TAi data = carrier cycle / 2 – TAi1 data

When using the current instruction values obtained by vector calculation, etc.,

TAi1 data = carrier cycle / 4 – calculated current instruction value x constant

TAi data = carrier cycle / 2 – TAi1 data

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.

TAi1 data = carrier cycle / 4 – carrier cycle / 4 x sinN° 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).

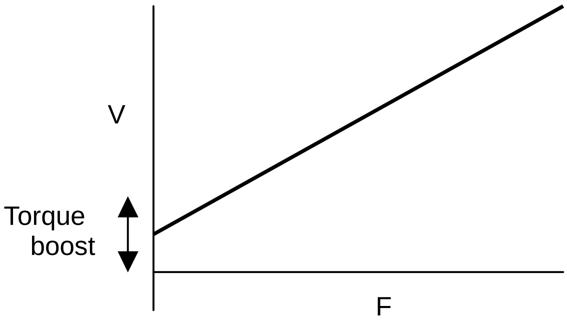


Figure 5.3 V/F Control

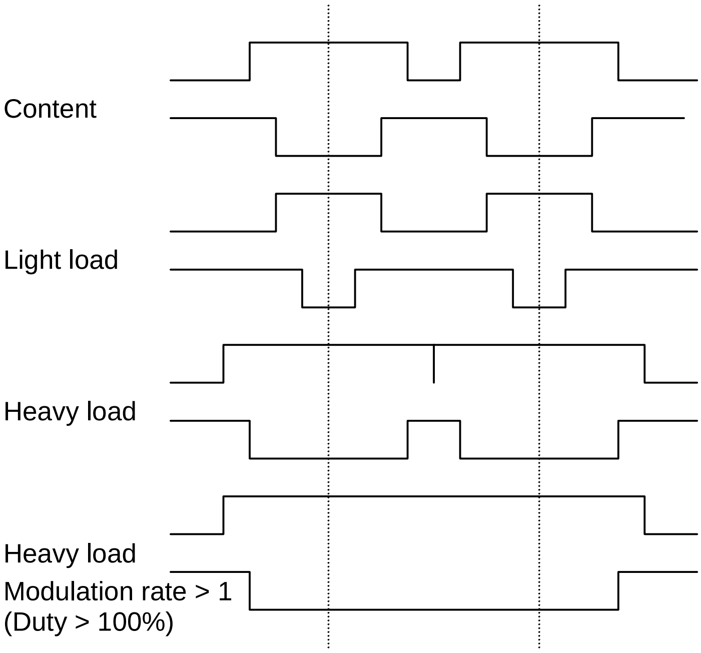


Figure 5.4 Relationship between Modulation Rate and Output Waveform

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.

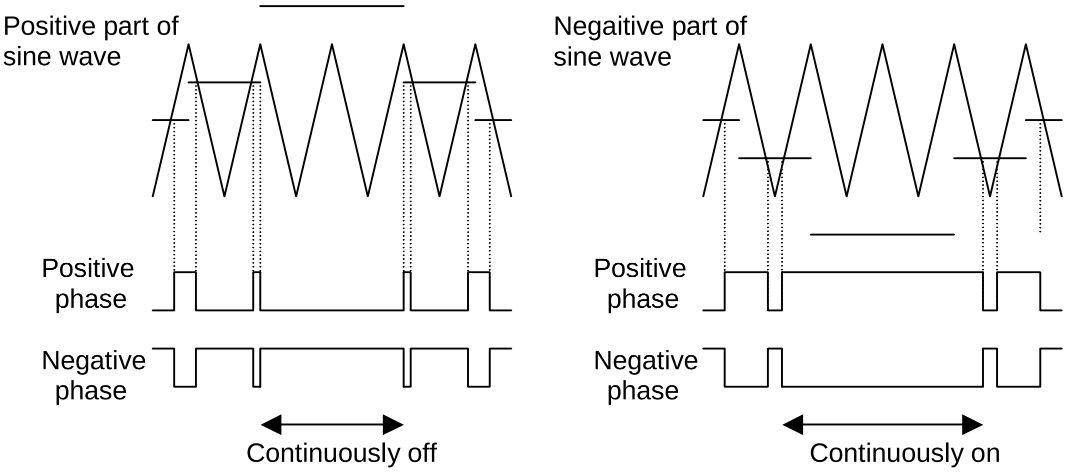
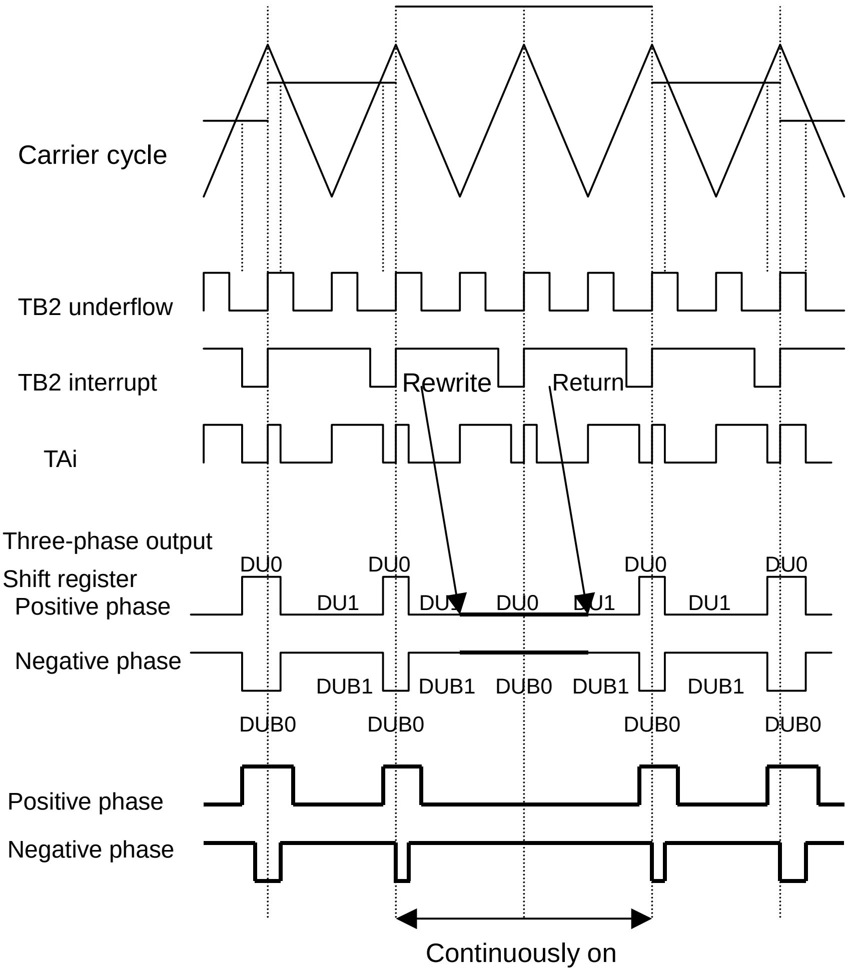


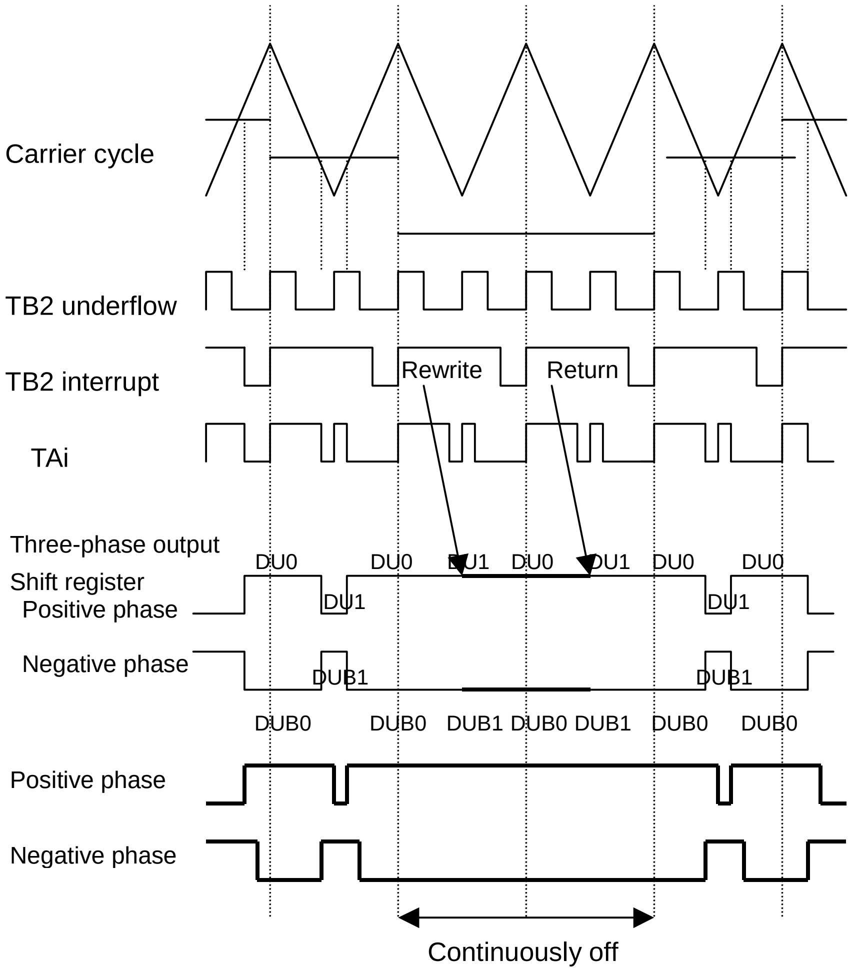
Figure 5.5 Conceptual Waveform of Two-phase Modulation Output

5.5.1 Rewriting the Three-phase Output Buffer Register to Produce Continuously On/Off Waveform Output

By rewriting the content of the three-phase output buffer register with the appropriate timing as shown below, it is possible to produce continuously on/off waveform output.

Because this method involves directly rewriting the three-phase output buffer register, there is a possibility of producing shorting laden waveform output unless data are written to the correct register bits. Therefore, always make sure that the simultaneous positive-negative active output disable bit (invc0 bit 4) is set to 1.





* The content of the three-phase output buffer register is transferred to the shift register at the first TB2 underflow after being rewritten.

Figure 5.6 Three-phase Output Buffer Register Rewrite Timing

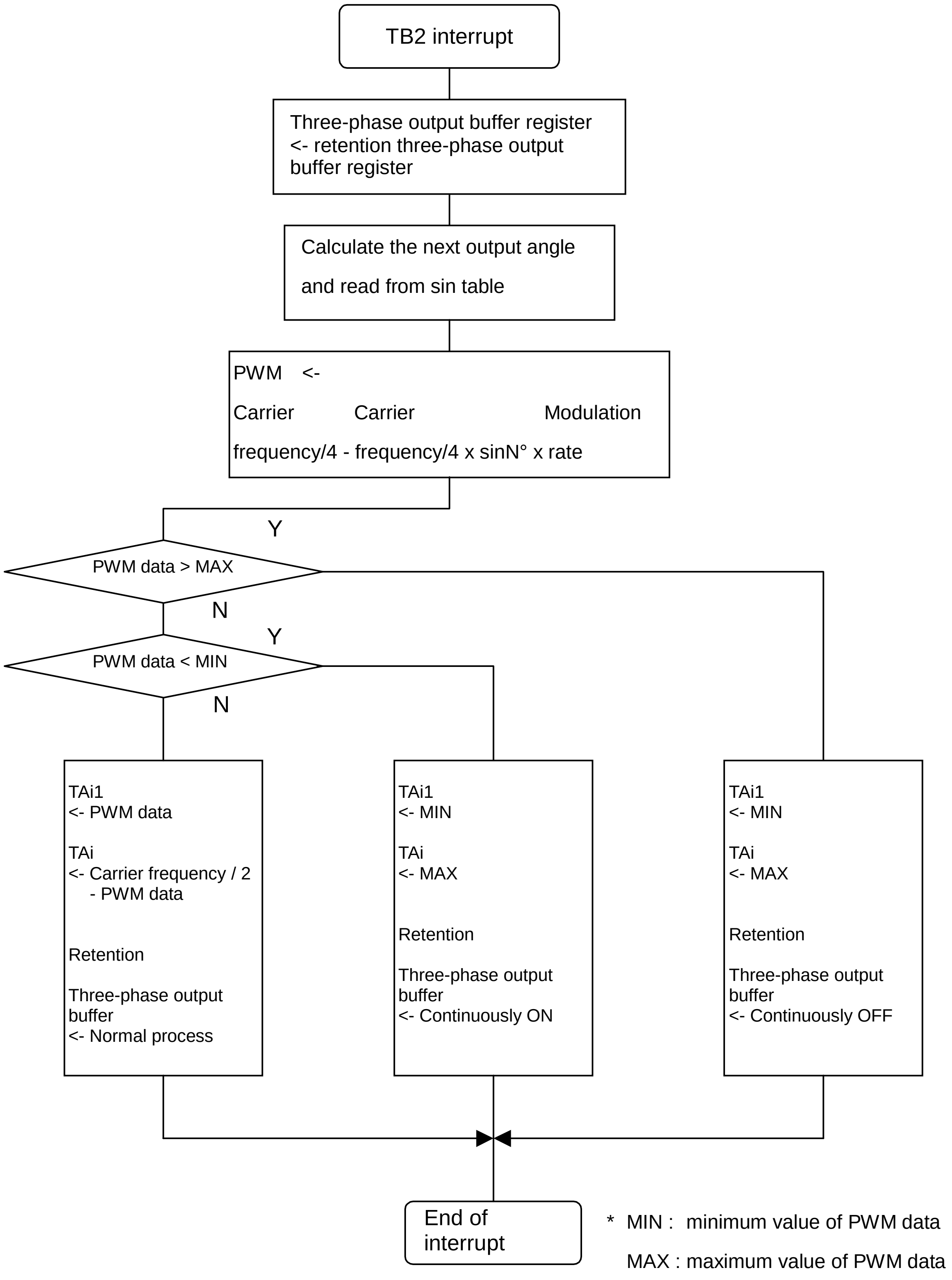

TB2 interrupt
Three-phase output buffer register <- retention three-phase output buffer register
Calculate the next output angle and read from sin table
PWM <- Carrier frequency/4 - Carrier frequency/4 x sinN° x Modulation rate
Y
PWM data > MAX
N
Y
PWM data < MIN
N
TAi1 <- PWM data
TAi <- Carrier frequency / 2 - PWM data
Retention Three-phase output buffer <- Normal process
TAi1 <- MIN
TAi <- MAX
Retention Three-phase output buffer <- Continuously ON
TAi1 <- MIN
TAi <- MAX
Retention Three-phase output buffer <- Continuously OFF
End of interrupt
* MIN : minimum value of PWM data
MAX : maximum value of PWM data

Figure 5.7 Example of a TB2 Interrupt Processing Flowchart (Method 1)

5.5.2 Using Timer Set Values to Produce Continuously On/Off Waveform Output

This is accomplished by setting 0 in the TAi and TAi1 registers to create a condition under which the shift register content cannot be shifted, as shown below.

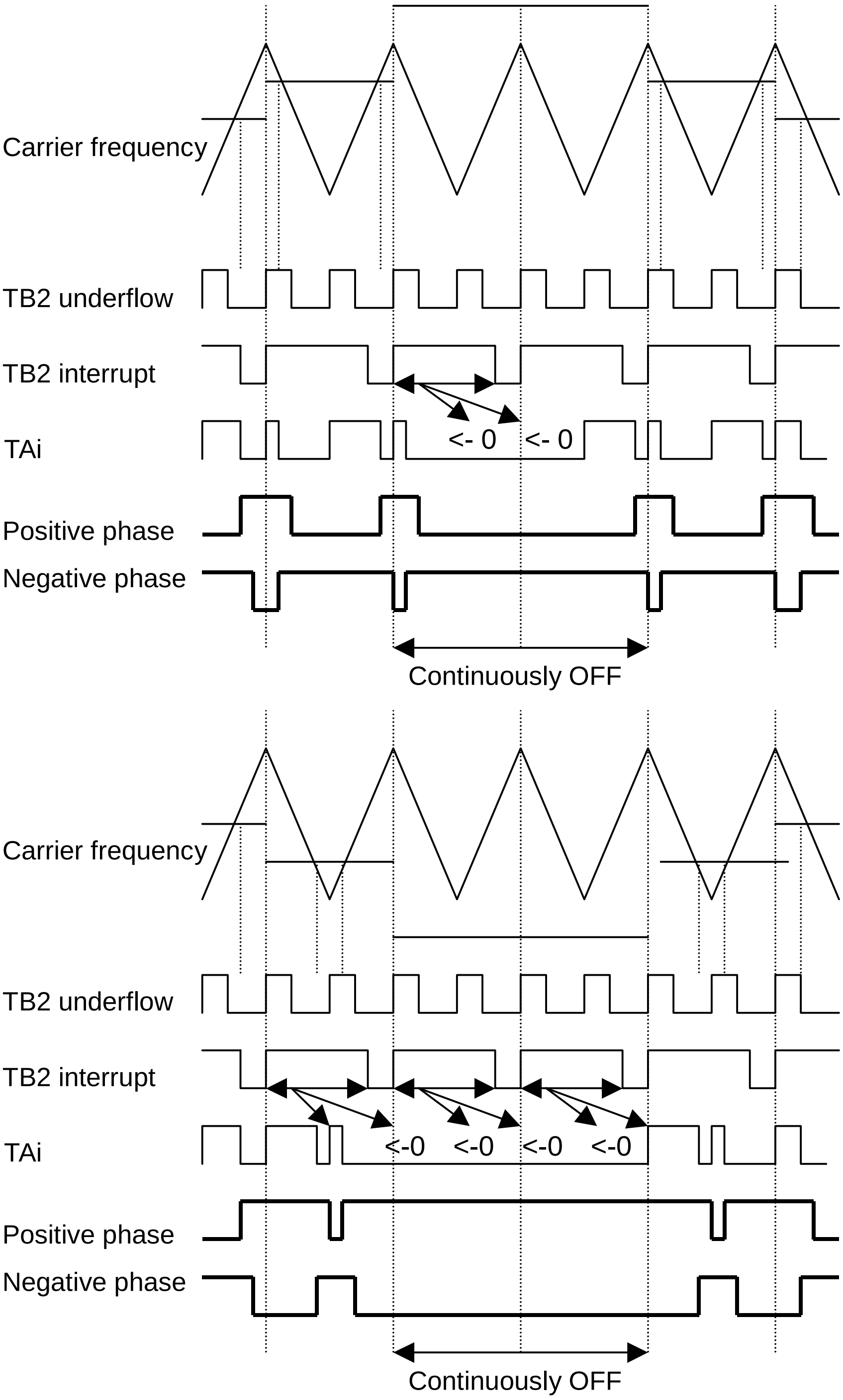


Figure 5.8 Relationship between TAi Data Setup Timing and Reflection

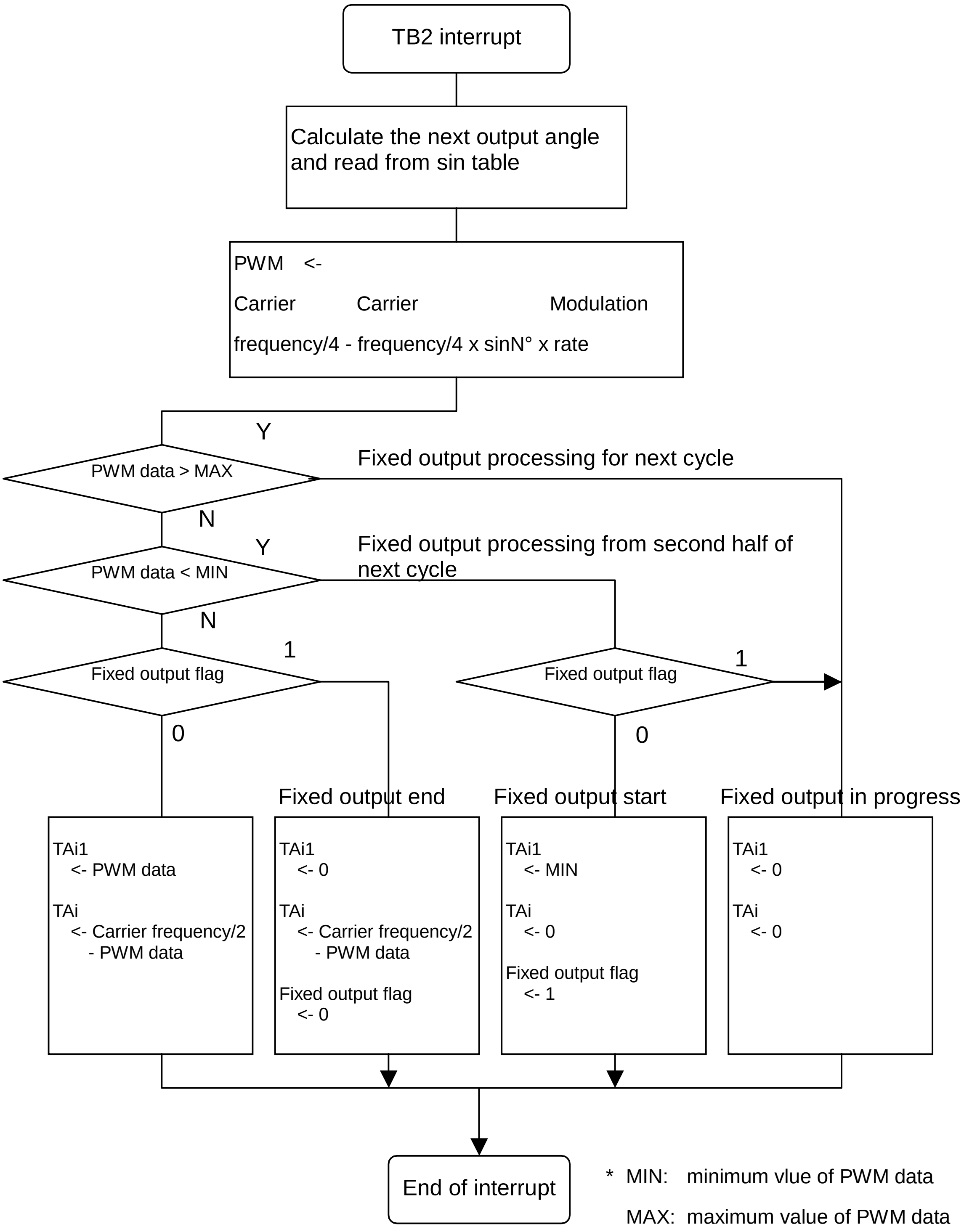

TB2 interrupt
Calculate the next output angle and read from sin table
PWM <-
Carrier frequency/4 - Carrier frequency/4 x sinN° x Modulation rate
Y
PWM data > MAX
Fixed output processing for next cycle
N
Y
PWM data < MIN
Fixed output processing from second half of next cycle
N
1
Fixed output flag
Fixed output flag
1
0
0
Fixed output end
Fixed output start
Fixed output in progress
TAi1 <- PWM data
TAi <- Carrier frequency/2 - PWM data
TAi1 <- 0
TAi <- Carrier frequency/2 - PWM data
Fixed output flag <- 0
TAi1 <- MIN
TAi <- 0
Fixed output flag <- 1
TAi1 <- 0
TAi <- 0
End of interrupt
* MIN: minimum vlue of PWM data
MAX: maximum value of PWM data

Figure 5.9 Example of a TB2 Interrupt Processing Flowchart (Method 2)

5.6 Altering the Output Frequency

Alteration of the output frequency can be accomplished by manipulating the sine wave output angle by, for example, skipping through sine wave tables. The following describes how to manipulate the output angle for an induction motor as an example.

Assume the case where the carrier frequency is 3.6 kHz and 360 data tables are available, one for 1°. If the output frequency in this case is 10 Hz, read one data table at a time for each carrier. To change it to 5 Hz, because the sine wave cycle is doubled, read each data table twice.

	(Reference)	
Output angle	sinN° at 10MHz	sinN° at 5MHz
0°	0	0
1°	0.0175	0
2°	0.0349	0.0175
3°	0.0523	0.0175
4°	0.0698	0.0349
5°	0.0872	0.0349
6°	0.1045	0.0523
7°	0.1219	0.0523
8°	0.1392	0.0698
9°	0.1564	0.0698
10°	0.1736	0.0872

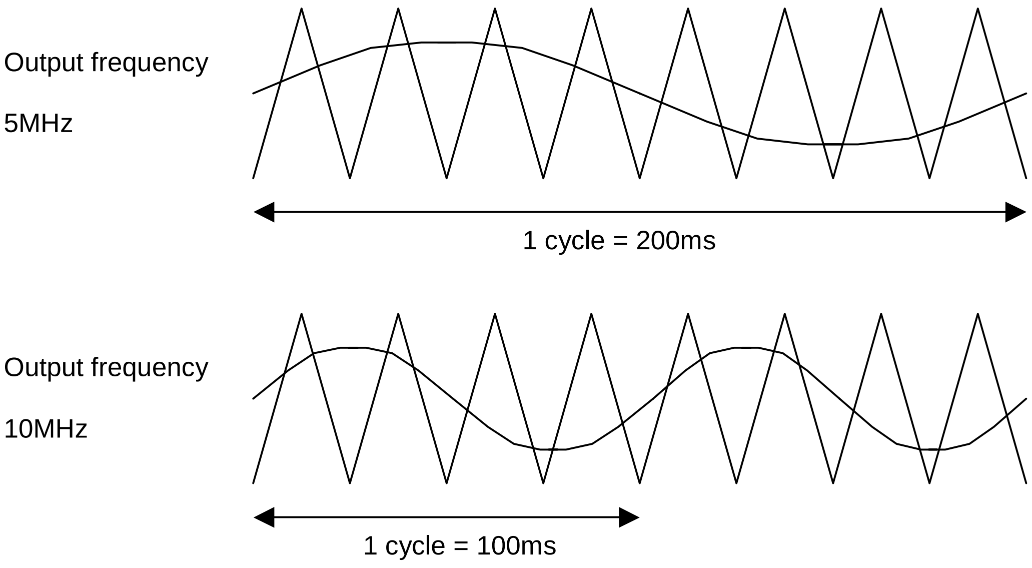


Figure 5.10 Altering the Output Frequency

6. 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 three-phase motor control timer functions.

6.2 Using the Three-phase Motor Control Timer Functions to Produce Waveform Output

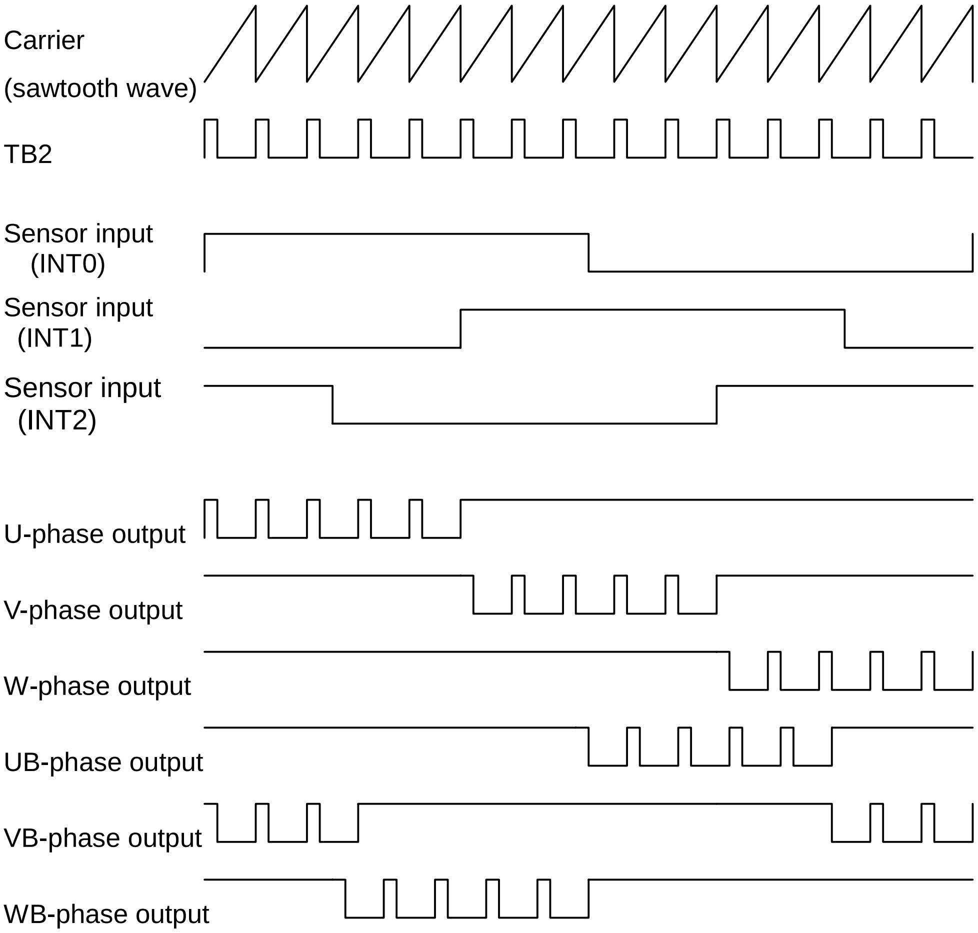
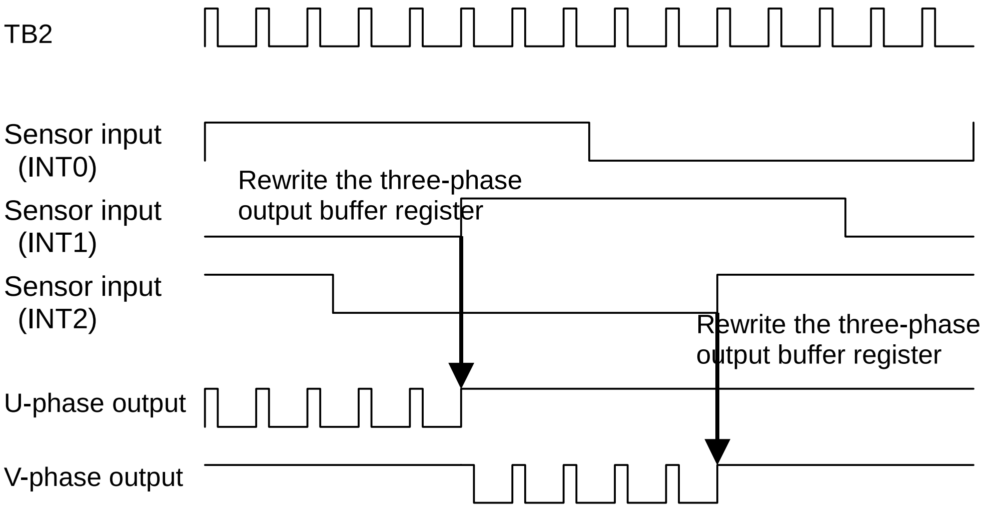


Figure 6.1 Relationship between 120-degree Turn-on Control Sensor Input and Waveform Output

Select triangular wave modulation mode for the modulation method and use three-phase mode 0. Switch active phases from one to another by rewriting the three-phase output buffer register within an INT interrupt.
(invc0=*1**1100; invc1=00000*0*; idb0=00******; idb1=00******;)

* Active phase switchover can also be accomplished by changing to general-purpose ports by rewriting the three-phase output buffer register.


TB2
Sensor input (INT0)
Sensor input (INT1)
Sensor input (INT2)
U-phase output
V-phase output
Rewrite the three-phase output buffer register
Rewrite the three-phase output buffer register

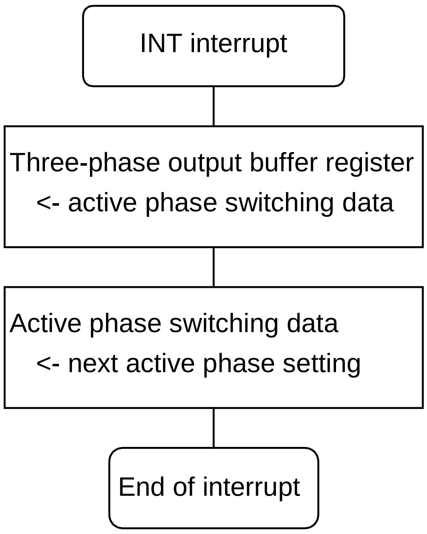

INT interrupt
Three-phase output buffer register <- active phase switching data
Active phase switching data <- next active phase setting
End of interrupt

Figure 6.2 Sensor 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 TA4, TA1 and TA2 each time the torque instruction value changes.

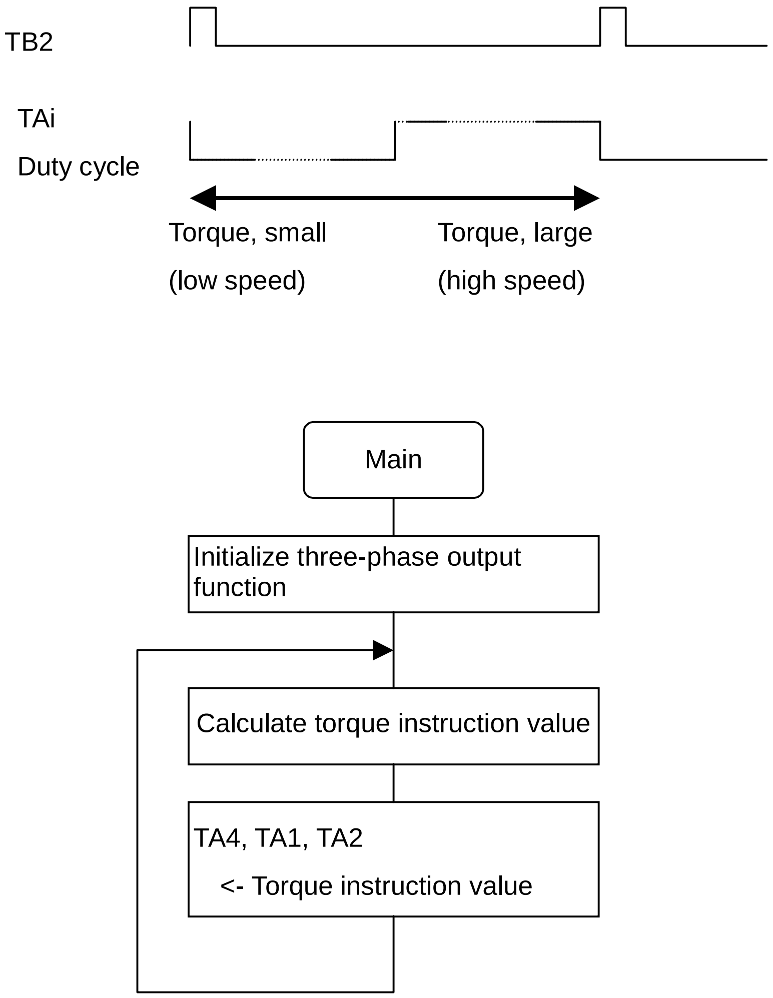

TB2
TAi
Duty cycle
Torque, small
(low speed)
Torque, large
(high speed)
Main
Initialize three-phase output function
Calculate torque instruction value
TA4, TA1, TA2
<- Torque instruction value

Figure 6.3 Relationship between Torque Instruction Value and TAi

7. Reference program example

An example to do three-phase motor wave form 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 wave form output is shown.

```
/*******************************************************************************
*      A sine wave form output reference program
*
*   (C)2003. Renesas Technology Corp. and Renesas Solutions Corp.,
*   All rights reserved.
******************************************************************************/

/******************************************************************************/
/*                                                                            */
/*   SFR setting                                                              */
/*                                                                            */
/******************************************************************************/
volatile char invc0;
#pragma ADDRESS      invc0  0308h         /* Three-phase PWM control register 0 */
volatile char invc1;
#pragma ADDRESS      invc1  0309h         /* Three-phase PWM control register 1 */
volatile char ictb2;
#pragma ADDRESS      ictb2  030dh
                     /* Timer B2 interrupt occurrences frequency set counter */
volatile char idb0;
#pragma ADDRESS      idb0   030ah  /* Three-phase output buffer register 0 */
volatile char idb1;
#pragma ADDRESS      idb1   030bh  /* Three-phase output buffer register 1 */
volatile char  ta1mr;
#pragma ADDRESS      ta1mr  0357h         /* Timer A1 mode register */
volatile char ta2mr;
#pragma ADDRESS      ta2mr  0358h         /* Timer A2 mode register */
volatile char ta4mr;
#pragma ADDRESS      ta4mr  035ah         /* Timer A4 mode register */
volatile char tb2mr;
#pragma ADDRESS      tb2mr  035dh         /* Timer B2 mode register */
volatile char tb2sc;
#pragma ADDRESS      tb2sc  035eh         /* Timer B2 special mode register */
volatile char trgsr;
#pragma ADDRESS      trgsr  0343h         /* Trigger select register */
volatile short tb2;
#pragma ADDRESS      tb2    0354h         /* Timer B2 register */
```

```
volatile char dtt;
#pragma ADDRESS      dtt    030ch                /* Dead time timer */
volatile short ta4;
#pragma ADDRESS      ta4    034eh                /* Timer A4 register */
volatile short ta1;
#pragma ADDRESS      ta1    0348h                /* Timer A1 register */
volatile short ta2;
#pragma ADDRESS      ta2    034ah                /* Timer A2 register */
volatile short ta41;
#pragma ADDRESS      ta41   0306h                /* Timer A4-1 register */
volatile short ta11;
#pragma ADDRESS      ta11   0302h                /* Timer A1-1 register */
volatile short ta21;
#pragma ADDRESS      ta21   0304h                /* Timer A2-1 register */*/
volatile char ps1;
#pragma ADDRESS      ps1    03b1h                /* Function select register A1 */
volatile char ps2;
#pragma ADDRESS      ps2    03b4h                /* Function select register A2 */
volatile char psl1;
#pragma ADDRESS      psl1   03b3h                /* Function select register B1 */
volatile char psl2;
#pragma ADDRESS      psl2   03b6h                /* Function select register B2 */
volatile char psc;
#pragma ADDRESS      psc    03afh                /* Function select register C */
volatile char tb2ic;
#pragma ADDRESS      tb2ic  0096h                /* Timer B2 interrupt control
register */
volatile char tabsr;
#pragma ADDRESS      tabsr  0340h                /* Count start flag */
volatile char prcr;
#pragma ADDRESS      prcr   000ah                /* Protect register */

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

#define CLK 20000000                      /* Microcomputer's frequency (Hz) */
#define CARR 20000                        /* Carrier frequency (Hz) */
#define DTT_TM 40                         /* Dead time timer (x 0.1µs) */

#define carr_set_2 ((CLK/CARR)/2)         /* Carrier cycle 1/2 */
#define carr_set_4 ((CLK/CARR)/4)         /* Carrier cycle 1/4 */
#define dtt_set ((CLK*DTT_TM)/10000000)   /* Dead time timer's value */
```

```
void main_ini()
{
       ictb2=1;       /* 1 time interruption at TB2 underflow 2 times arises */

       prcr=0x02;           /* Protection release */
       invc0=0x16;          /* Three-phase PWM output mode, Output disabled */
       invc1=0x42;          /* Three-phase mode 1 */
       prcr=0x00;           /* Protection */

       tb2sc=0x01;          /* Timer B2 reload timing = Synchronized rising edge of
                            triangular wave */

       idb0=0x15;           /* Three-phase output buffer register 0 setting */
       idb1=0x2a;           /* Three-phase output buffer register 1 setting */

       ta1mr=0x12;          /* Oneshot pulse mode */
       ta2mr=0x12;          /* Oneshot pulse mode */
       ta4mr=0x12;          /* Oneshot pulse mode */
       tb2mr=0x00;          /* Timer mode */
       trgsr=0x45;          /* TA1,TA2 and TA4's trigger is TB2 overflow */

       tb2=carr_set_2-1;                  /* Carrier cycle 1 / 2 */
       dtt=dtt_set;                       /* Dead time timer value */
       ta4=ta1=ta2=carr_set_4;            /* Duty 50% */
       ta41=ta11=ta21=carr_set_4;         /* Duty 50% */

       psc=0x04;                   /* Function select register setting */
       psl2=0x01;
       psl1=0x18;
       ps2=0x03;
       ps1=0x3c;

       tb2ic=0x07;                 /* Set "7" to TB2 interrupt priority level */

       asm("  FSET   I");          /* Enable interruption */

       tabsr=0x96;                 /* Start TA1,TA2,TA4 and TB2 timer */

       prcr=0x02;                  /* Protect release */
       invc0=0x1e;                 /* Three-phase output enabled */
       prcr=0x00;                  /* Protect */

}
```

```
/****************************************************************************/
/*                                                                          */
/*    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 value (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° x 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 Carrier / 4 */

}
```

```
/*****************************************************************************/
/*                                                                           */
/*     TB2 interrupt process                                                 */
/*                                                                           */
/*****************************************************************************/

void tb2_int(void);

void pwm_uvw_set(void);
void pwm_uvwa_set(void);
void pwm_uvwb_set(void);
void pwm_buf(void);
void i_con(void);
void angle(void);

unsigned char idb0_b=0x15;                /* Three-phase output buffer register 0
                                          (Temporary) */
unsigned char idb1_b=0x2a;                /* Three-phase output buffer register 1
                                          (Temporary) */
signed short tq_dat;                      /* Torque order value x Carrier / 4 */
signed short sinpt_sum;                   /* SIN pointer sum total counter */
signed short sin_pt;                      /* SIN tablr 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 valaue */
#define pwm_max      (carr_set_2-1)       /* PWM maximum value */
#define pwm_min 1                         /* PWM minimum value */
```

```
/*     TB2 interrupt (without two-phase modulation)            */

#pragma       INTERRUPT/B  tb2_int
void tb2_int(void)
{

       angle();                           /* Generate a SIN table pointer */
       i_con();                           /* Calcurate a PWM order value */
       pwm_uvw_set();/* PWM order value upper limit compansation ->Timer setting */

}


/*     TB2 interrupt (Two-phase modulation by timer value renewal)     */

#pragma       INTERRUPT/B  tb2_int
void tb2_int(void)
{

       angle();                           /* Generate a SIN table pointer */
       i_con();                           /* Calcurate a PWM order value */

       pwm_uvwa_set();/*PWM order vaalue upper limit compansation->Timer setting */

}


/*TB2 interrupt (Two-phase modulation by three-phase output buffer renewal )  */

#pragma       INTERRUPT/B  tb2_int
void tb2_int(void)
{

       pwm_buf();                  /* Three-phase output buffer setting */
       angle();                    /* Generate a SIN table pointer */
       i_con();                    /* Calcurate a PWM order value */

       pwm_uvwb_set();/* PWM order value upper limit compansation->Timer setting */

}
```

```
/******************************************************************************/
/*                                                                           */
/*Example of TB2 interrupt operation module ( in case of a guidance motor )  */
/*            Calcurate a SIN angle                                          */
/*            Calcurate 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 tatal is maximum ? (23040 =
                                   360° x 64)    */
       {
              sinpt_sum=sinpt_sum-23040;  /* SIN pointer sum total upper limit
                                          compansation */
       }
       else
       {
       }
       sin_pt=sinpt_sum>>6;               /* SIN pointer <- SIN pointer sum total
                                          / 64 */
}


void i_con()
{
/*     U-phase PWM order value = Carrier / 4 – (sinN° x (Torque order value x 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)° x (Torque order value
x 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)° x (Torque order value
x 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 TB2 interrupt                                                */
/*     PWM data setting                                                      */
/*                                                                           */
/*****************************************************************************/

/*                   Without two-phase modulation                            */

void pwm_uvw_set()
{

/* U-phase PWM compansation */
       if(pwm_max<pwm_u_w)                /* Duty maximum ? */
       {
              ta41=pwm_max;               /* The first half  <- Maximum value */
              ta41=pwm_max;               /* The first half  <- Maximum value */
              ta4=pwm_min;                /* The latter half <- Minimum value */
       }
       else
       {
              if(pwm_min>pwm_u_w)         /* Duty minimum ? */
              {
                     ta41=pwm_min;        /* The first half  <- Minimum value */
                     ta41=pwm_min;        /* The first half  <- Minimum value */
                     ta4=pwm_max;         /* The latter half <- Maximum value */
              }
              else                        /* Minimum < Duty < Maximum */
              {
                     ta41=pwm_u_w;        /* The first half  <- PWM order value */
                     ta41=pwm_u_w;        /* The first half  <- PWM order value */
                     ta4=carr_set_2-pwm_u_w;     /* The latter half <- Carrier cycle /
                                                 2 – U-phase PWM order value */
              }
       }

/* V-phase PWM compansation */
       if(pwm_max<pwm_v_w)                /* Duty maximum ? */
       {
              ta11=pwm_max;               /* The first half  <- Maximum value */
              ta11=pwm_max;               /* The first half  <- Maximum value */
              ta1=pwm_min;                /* The latter half <- Minimum value */
       }
       else
       {
              if(pwm_min>pwm_v_w)         /* Duty minimum ? */
              {
                     ta11=pwm_min;        /* The first half    <- Minimum value */
                     ta11=pwm_min;        /* The first half    <- Minimum value */
                     ta1=pwm_max;         /* The latter half   <- Maximum value */
              }
else                                /* Minimum < Duty < Maximum */
              {
                     ta11=pwm_v_w; /* The first half      <- PWM order value */
                     ta11=pwm_v_w; /* The first half      <- PWM order value */
                     ta1=carr_set_2-pwm_v_w;      /* The latter half      <- Carrier
                                                  cycle / 2 – V-phase PWM oeder value */
              }
       }

/* W-phase PWM compansation */
       if(pwm_max<pwm_w_w)          /* Duty maximum ? */
       {
              ta21=pwm_max;         /* The first half    <- Maximum value */
              ta21=pwm_max;         /* The first half    <- Maximum value */
              ta2=pwm_min;          /* The latter half   <- Minimum value */
       }
       else
       {
              if(pwm_min>pwm_w_w)   /* Duty minimum ? */
              {
                     ta21=pwm_min; /* The first half    <- Minimum value */
                     ta21=pwm_min; /* The first half    <- Minimum value */
                     ta2=pwm_max;  /* The latter half   <- Maximum value */
              }
              else                  /* Minimum < Duty < Maximum */
              {
                     ta21=pwm_w_w; /* The first half      <- PWM order value */
                     ta21=pwm_w_w; /* The first half      <- PWM order value */
                     ta2=carr_set_2-pwm_w_w;/* The latter half   <- Carrier cycle / 2
                                                 – W-phase PWM order value */
              }
       }
}




/*     Two-phase modulation by timer value "0" setup    */


struct tag{
              char   bit0:1;
              char   bit1:1;
              char   bit2:1;
       }flag_buf;
#define arm_u_flg flag_buf.bit0    /* U-phase continuity arm output flag */
#define arm_v_flg flag_buf.bit1    /* V-phase continuity arm output flag */
#define arm_w_flg flag_buf.bit2    /* W-phase continuity arm output flag */
```

```
void pwm_uvwa_set()
{

/* U-phase PWM compansation */
        if(pwm_max<pwm_u_w)             /* Duty maximum ? */
        {                               /* The fixation output start or inside */
                ta41=0;                 /* The first half    <- 0 */
                ta41=0;                 /* The first half    <- 0 */
                ta4=0;                  /* The latter half   <- 0 */
        }
        else
        {
                if(pwm_min>pwm_u_w)     /* Duty minimum ? */
                {
                        if(arm_u_flg==1)        /* Under fixation output */
                        {
                                ta41=0;         /* The first half    <- 0 */
                                ta41=0;         /* The first half    <- 0 */
                                ta4=0;          /* The latter half   <- 0 */
                        }
                        else                    /* Fixation output start */
                        {
                                ta41=pwm_min;   /* The first half    <- Minimum value */
                                ta41=pwm_min;   /* The first half    <- Minimum value */
                                ta4=0;          /* The latter half   <- 0             */
                                arm_u_flg=1;    /* Fixation output flag set */
                        }
                }
                else                            /* Minimum < Duty < Maximum */
                {
                        if(arm_u_flg==1)        /* Fixation output end */
                        {
                                ta41=0;         /* The first half      <- 0 */
                                ta41=0;         /* The first half      <- 0 */
                                ta4=carr_set_2-pwm_u_w;         /* The latter half     <-
                                                Carrier cycle / 2 – U-phase PWM order value */
                                arm_u_flg=0;    /* Fixation output flag clrar */
                        }
                        else                    /* Normal process */
                        {
                                ta41=pwm_u_w;   /* The first half      <- PWM order value */
                                ta41=pwm_u_w;   /* The first half      <- PWM order value */
                                ta4=carr_set_2-pwm_u_w;/* The latter half    <- Carrier
                                                        cycle / 2 – U-phase PWM order value */
                        }
                }
        }
```

```
/* V-phase PWM compansation */
       if(pwm_max<pwm_v_w)         /* Duty maximum ? */
       {                           /* The fixation output start or inside */
              ta11=0;              /* The first half    <- 0 */
              ta11=0;              /* The first half    <- 0 */
              ta1=0;               /* The latter half   <- 0 */
       }
       else
       {
              if(pwm_min>pwm_v_w)  /* Duty minimum ? */
              {
                     if(arm_v_flg==1)     /* Under fixation output */
                     {
                            ta11=0;       /* The first half    <- 0 */
                            ta11=0;       /* The first half    <- 0 */
                            ta1=0;        /* The latter half   <- 0 */
                     }
                     else                 /* Continuous arm start */
                     {
                            ta11=pwm_min; /* The first half    <- Minimum value */
                            ta11=pwm_min; /* The first half    <- Minimum value */
                            ta1=0;        /* The latter half   <- 0 */
                            arm_v_flg=1;  /* Fixation output flag set */
                     }
              }
              else                        /* Minimum < Duty < Maximum */
              {
                     if(arm_v_flg==1)     /* Fixation output end */
                     {
                            ta11=0;       /* The first half      <- 0 */
                            ta11=0;       /* The first half      <- 0 */
                            ta1=carr_set_2-pwm_v_w;/* The latter half    <- Carrier
                                                 cycle / 2 – V-phase PWM order value */
                            arm_v_flg=0;  /* Fixation output flag clear */
                     }
                     else                 /* Normal process */
                     {
                            ta11=pwm_v_w; /* The first half      <- PWM order value */
                            ta11=pwm_v_w; /* The first half      <- PWM order value */
                            ta1=carr_set_2-pwm_v_w;     /* The latter half     <-
                                          Carrier cycle / 2 – V-phase PWM order value */
                     }
              }
       }
```

```
/* W-phase PWM compansation */
       if(pwm_max<pwm_w_w)         /* Duty maximum ? */
       {                           /* The fixation output start or inside */
              ta21=0;              /* The first half    <- 0 */
              ta21=0;              /* The first half    <- 0 */
              ta2=0;               /* The latter half   <- 0*/
       }
       else
       {
              if(pwm_min>pwm_w_w)  /* Duty minimum ? */
              {
                     if(arm_w_flg==1)            /* Under fixation output */
                     {
                            ta21=0;              /* The first half    <- 0 */
                            ta21=0;              /* The first half    <- 0 */
                            ta2=0;               /* The latter half   <- 0 */

                     }
                     else                        /* Fixation output start */
                     {
                            ta21=pwm_min; /* The first half    <- Minimum value */
                            ta21=pwm_min; /* The first half    <- Minimum value */
                            ta2=0;        /* The latter half   <- 0 */
                            arm_w_flg=1;  /* Fixation output flag set */
                     }
              }
              else                        /* Minimum < Duty < Maximum */
              {
                     if(arm_w_flg==1)     /* Fixation output end */
                     {
                            ta21=0;       /* The first half    <- 0 */
                            ta21=0;       /* The first half    <- 0 */
                            ta2=carr_set_2-pwm_w_w;     /* The latter half    <-
                                          Carrier cycle / 2 – W-phase PWM order value */
                            arm_w_flg=0;  /* Continuous arm flag clear */
                     }
                     else                 /* Normal process */
                     {
                            ta21=pwm_w_w; /* The first half    <- PWM order value */
                            ta21=pwm_w_w; /* The first half    <- PWM order value */
                            ta2=carr_set_2-pwm_w_w;     /* The latter half    <-
                                          Carrier cycle / 2 – W-phase PWM order value */
                     }
              }
       }

}
```

```
/*     Two-phase modulation by the three-phase output buffer setting  */


void pwm_buf()
{
       idb0=idb0_b;         /* Three-phase output buffer 0 setting */
       idb1=idb1_b;         /* Three-phase output buffer 1 setting */
}


void pwm_uvwb_set()
{

/* U-phase PWM compansation */
       if(pwm_max<pwm_u_w)         /* Duty maximum ? */
       {
              idb0_b=idb0_b&0xfc;
              idb0_b=idb0_b|0x02;

              ta41=pwm_max;        /* The first half    <- Maximum value */
              ta41=pwm_max;        /* The first half    <- Maximum value */
              ta4=pwm_min;         /* The latter half   <- Minimum value */
       }
       else
       {
              if(pwm_min>pwm_u_w)  /* Duty minimum ? */
              {
                     idb1_b=idb1_b&0xfc;
                     idb1_b=idb1_b|0x01;
                     ta41=pwm_min;        /* The first half    <- Minimum value */
                     ta41=pwm_min;        /* The first half    <- Minimum value */
                     ta4=pwm_max;         /* The latter half   <- Maximum value */
              }
              else                        /* Minimum < Duty < Maximum */
              {
                     idb0_b=idb0_b&0xfc;
                     idb0_b=idb0_b|0x01;
                     idb1_b=idb1_b&0xfc;
                     idb1_b=idb1_b|0x02;

                     ta41=pwm_u_w;        /* The firsr half<- PWM order value */
                     ta41=pwm_u_w;        /* The firsr half<- PWM order value */
                     ta4=carr_set_2-pwm_u_w;/* The latter half <- Carrier cycle /
                                                  2 – U-phase PWM order value */
              }
       }
```

```
/* V-phase PWM compansation */
       if(pwm_max<pwm_v_w)         /* Duty maximum ? */
       {
              idb0_b=idb0_b&0xf3;
              idb0_b=idb0_b|0x08;

              ta11=pwm_max;        /* The firsr half    <- Maximum value */
              ta11=pwm_max;        /* The firsr half    <- Maximum value */
              ta1=pwm_min;         /* The latter half   <- Minimum value */
       }
       else
       {
              if(pwm_min>pwm_v_w)  /* Duty minimum ? */
              {
                     idb1_b=idb1_b&0xf3;
                     idb1_b=idb1_b|0x04;

                     ta11=pwm_min; /* The first half    <- Minimum value */
                     ta11=pwm_min; /* The first half    <- Minimum value */
                     ta1=pwm_max;  /* The latter half   <- Maximum value */
              }
              else                 /* Minimum < Duty < Maximum */
              {
                     idb0_b=idb0_b&0xf3;
                     idb0_b=idb0_b|0x04;
                     idb1_b=idb1_b&0xf3;
                     idb1_b=idb1_b|0x08;

                     ta11=pwm_v_w; /* The first half    <- PWM order value */
                     ta11=pwm_v_w; /* The first half    <- PWM order value */
                     ta1=carr_set_2-pwm_v_w;     /* The latter half   <- Carrier
                                          cycle / 2 – V-phase PWM order value */
              }
       }

/* W-phase PWM compansation */
       if(pwm_max<pwm_w_w)         /* Duty maximum ? */
       {
              idb0_b=idb0_b&0xcf;
              idb0_b=idb0_b|0x20;

              ta21=pwm_max;        /* The first half    <- Maximum value */
              ta21=pwm_max;        /* The first half    <- Maximum value */
              ta2=pwm_min;         /* The latter half   <- Minimum value */
       }
       else
       {
              if(pwm_min>pwm_w_w)  /* Duty minimum ? */
              {
                     idb1_b=idb1_b&0xcf;
                     idb1_b=idb1_b|0x10;

                     ta21=pwm_min;        /* The first half    <- Minimum value */
                     ta21=pwm_min;        /* The first half    <- Minimum value */
                     ta2=pwm_max;         /* The latter half   <- Maximum value */
              }
              else                        /* Minimum < Duty < Maximum */
              {
                     idb0_b=idb0_b&0xcf;
                     idb0_b=idb0_b|0x10;
                     idb1_b=idb1_b&0xcf;
                     idb1_b=idb1_b|0x20;

                     ta21=pwm_w_w; /* The first half    <- PWM order value */
                     ta21=pwm_w_w; /* The first half    <- PWM order value */
                     ta2=carr_set_2-pwm_w_w;/* The latter half <- Carrier cycle /
                                          2 – W-phase PWM order value */
              }
       }
}
```

7.2 120° energization reference program

Shows the 120° energization output example

```
/******************************************************************************
*      120° energization reference program
*
*   (C)2003. Renesas Technology Corp. and Renesas Solutions Corp.,
*   All rights reserved.
******************************************************************************/
/******************************************************************************/
/*                                                                           */
/*    SFR setting                                                            */
/*                                                                           */
/******************************************************************************/
volatile char invc0;
#pragma ADDRESS      invc0  0308h  /* Three-phase PWM control register 0 */
volatile char invc1;
#pragma ADDRESS      invc1  0309h  /* Three-phase PWM control register 1 */
volatile char ictb2;
#pragma ADDRESS      ictb2  030dh  /* Timer B2 interrupt occurrences frequency set
                                   counter */
volatile char idb0;
#pragma ADDRESS      idb0   030ah  /* Three-phase output buffer register 0 */
volatile char idb1;
#pragma ADDRESS      idb1   030bh  /* Three-phase output buffer register 1 */
volatile char  ta1mr;
#pragma ADDRESS      ta1mr  0357h  /* Timer A0 mode register */
volatile char ta2mr;
#pragma ADDRESS      ta2mr  0358h  /* Timer A2 mode register */
volatile char ta4mr;
#pragma ADDRESS      ta4mr  035ah  /* Timer A4 mode register */
volatile char tb2mr;
#pragma ADDRESS      tb2mr  035dh  /* Timer B2 mode register */
volatile char tb2sc;
#pragma ADDRESS      tb2sc  035eh  /* Timer B2 special mode register */
volatile char trgsr;
#pragma ADDRESS      trgsr  0343h  /* Torigger select register */
volatile short tb2;
#pragma ADDRESS      tb2    0354h  /* Timer B2 register */
volatile char dtt;
#pragma ADDRESS      dtt    030ch  /* Dead time timer */
volatile short ta4;
#pragma ADDRESS      ta4    034eh  /* Timer A4 register */
```

```
volatile short ta1;
#pragma ADDRESS      ta1    0348h         /* Timer A1 register */
volatile short ta2;
#pragma ADDRESS      ta2    034ah         /* Timer A2 register */
volatile char ps1;
#pragma ADDRESS      ps1    03b1h         /* Function select register A1 */
volatile char ps2;
#pragma ADDRESS      ps2    03b4h         /* Function select register A2 */
volatile char psl1;
#pragma ADDRESS      psl1   03b3h         /* Function select register B1 */
volatile char psl2;
#pragma ADDRESS      psl2   03b6h         /* Function select register B2 */
volatile char psc;
#pragma ADDRESS      psc    03afh         /* Function select register C */
volatile char tabsr;
#pragma ADDRESS      tabsr  0340h         /* Count start flag */
volatile char prcr;
#pragma ADDRESS      prcr   000ah         /* Protect register */
volatile char ifsr;
#pragma ADDRESS      ifsr   031fh         /* External interrupt request cause
                                          select register */
volatile char int0ic;
#pragma ADDRESS      int0ic 009eh         /* INT0 interrupt control register */
volatile char int1ic;
#pragma ADDRESS      int1ic 007eh         /* INT1 interrupt control register */
volatile char int2ic;
#pragma ADDRESS      int2ic 009ch         /* INT2 interrupt control register */



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


#define CLK 20000000                /* Microcomputer's frequency (Hz) */
#define CARR 20000                  /* Carrier frequency (Hz) */

#define carr_set (CLK/CARR)         /* Carrier cycle */
```

```
void main_ini()
{
       ictb2=1;      /* 1 time interruption at TB2 underflow 2 times arises */

       prcr=0x02;    /* Protect release */
       invc0=0x44;   /* Three-phase PWM output, Sawtooth wave form modulation, Output
                     disabled */
       invc1=0x60;   /* Three-phase mode 0, Dead time invalid */
       prcr=0x00;    /* Protect */

       tb2sc=0x00;   /* Timer B2 reload timimg = Next underflow */

       idb0=0x0ff;   /* Three-phase output buffer register 0 setting */
       idb1=0x0ff;   /* Three-phase output buffer register 1 setting */

       ta1mr=0x12;   /* Oneshot pulse mode */
       ta2mr=0x12;   /* Oneshot pulse mode */
       ta4mr=0x12;   /* Oneshot pulse mode */
       tb2mr=0x00;   /* Timer mode */
       trgsr=0x45;   /* TA1,TA2 and TA4's trigger is TB2 overflow */

       tb2=carr_set; /* Carrier cycle */
       dtt=1;        /* Dead time timer = Minimum */

       ta4=1;        /* Torque 0% */
       ta1=1;        /* Torque 0% */
       ta2=1;        /* Torque 0% */

       /* INT is used for input of a position. */
       ifsr=0x07;                  /* INT0,1,2 both edge */
       int0ic=7;                   /* INT0,1,2 interrupt priority level setting */
       int1ic=7;
       int2ic=7;

       psc=0x04;
       psl2=0x01;
       psl1=0x18;
       ps2=0x03;
       ps1=0x3c;

       asm("  FSET   I");   /* Enable interruption */

       tabsr=0x96;                 /* Start a TA1,TA2,TA4 and TB2 timer */

       prcr=0x02;                  /* Protect release */
       invc0=0x4c;                 /* Sawtooth wavemodulation mode, Output enavled
*/
       prcr=0x00;                  /* Protect */
}
```

```
/*****************************************************************************/
/*                                                                           */
/*   Operation in main process                                               */
/*      The setting of the torque order value (a PWM duty)                   */
/*            corresponding to the speed                                     */
/*                                                                           */
/*****************************************************************************/
void dc_pwm_set(void);

unsigned short       tk_dat_w;            /* Torque order value */

void dc_pwm_set(void)
{
       ta4=ta1=ta2=tk_dat_w;
}



/*****************************************************************************/
/*                                                                           */
/*     Angle detection interrupt ( INT use )                                 */
/*     Change by the three-phase output buffer register at active phase of   */
/*     each phase                                                            */
/*****************************************************************************/


#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 idb0_bufb=0x0ff;            /* Next times, output phase */
signed short         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)
{
       idb0=idb0_bufb;             /* Output phase change */
       if(edge0_flg==0)
       {
              edge0_flg=1;
              rad_datw=60;         /* Next times, angle setting */
       }
       else
       {
              edge0_flg=0;
              rad_datw=240;        /* Next times, angle setting */
       }
       rad_to_pwm();               /* Next times, output phase setting */
}

void int1_int(void)
{
       idb0=idb0_bufb;             /* Output phase change */
       if(edge1_flg==0)
       {
              edge1_flg=1;
              rad_datw=180;        /* Next times, angle setting */
       }
       else
       {
              edge1_flg=0;
              rad_datw=0;          /* Next times, angle setting */
       }
       rad_to_pwm();               /* Next times, output phase setting */
}

void int2_int(void)
{
       idb0=idb0_bufb;             /* Output phase change */
       if(edge2_flg==0)
       {
              edge2_flg=1;
              rad_datw=300;        /* Next times, angle setting */
              rad_to_pwm();
       }
       else
       {
              edge2_flg=0;
              rad_datw=120;        /* Next times, angle setting */
       }
       rad_to_pwm();               /* Next times, output phase setting */
}
```

```
void rad_to_pwm(void)
{
       switch(rad_datw)
       {
              case 0:
                     idb0_bufb=0x0de;     /* Next output phase = U     WB    */
                     break;
              case 60:
                     idb0_bufb=0x0db;     /* Next output phase = V    WB     */
                     break;
              case 120:
                     idb0_bufb=0x0f9;     /* Next output phase =  V    UB      */
                     break;
              case 180:
                     idb0_bufb=0x0ed;     /* Next output phase =  W   UB       */
                     break;
              case 240:
                     idb0_bufb=0x0e7;     /* Next output phase =  W     VB   */
                     break;
              case 300:
                     idb0_bufb=0x0f6;     /* Next output phase = U     VB    */
                     break;
              default:
                     idb0_bufb=0x0ff;     /* Next output phase = off         */
                     break;
       }
}
```

7.3 Feedback control reference profram

As program development example of vector control, following control module are shown.

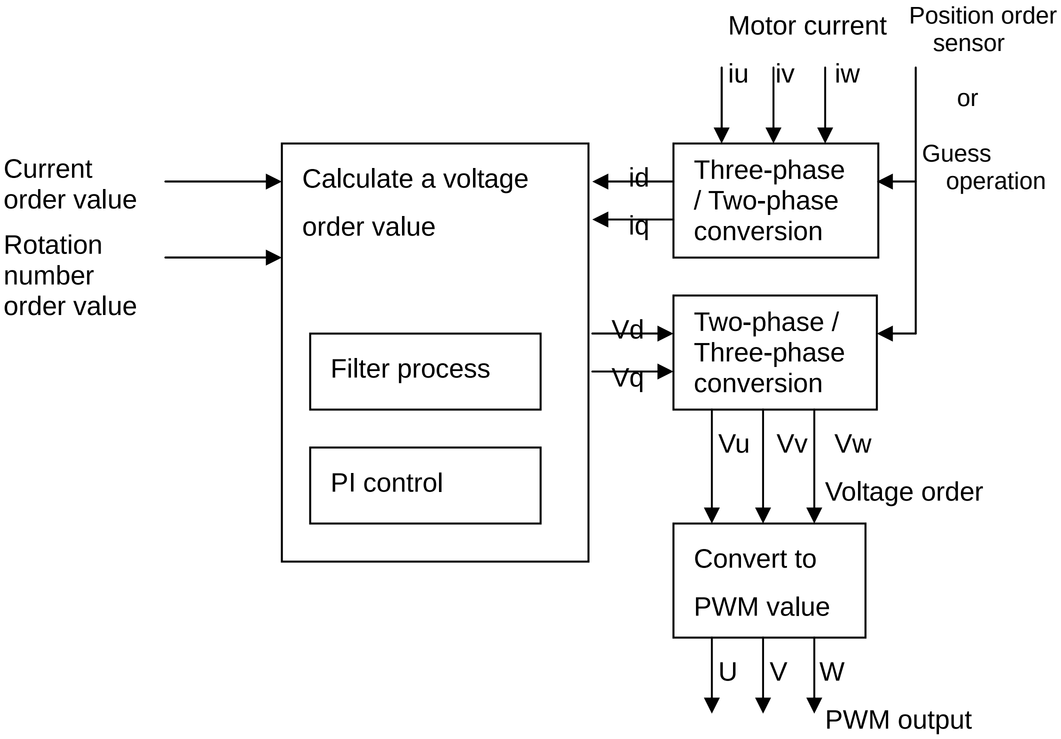
·Three-phase / Two-phase conversion

·Two-phase / Three-phase conversion

·PI control

·Filter process

About a PWM output, please refer to reference program to the preceding item.



7.3.1 Three-phase / Two-phase convert module

The example of three-phase / two-phase change is written in the following.

$$\begin{pmatrix} id \\ iq \end{pmatrix} = \sqrt{\frac{2}{3}} \begin{pmatrix} \cos\theta & \cos(\theta+240) & \cos(\theta+120) \\ -\sin\theta & -\sin(\theta+240) & \sin(\theta+120) \end{pmatrix} \begin{pmatrix} iu \\ iv \\ iw \end{pmatrix}$$

Calculated in the sample program of the rest as

iu	:U-phase current value	x 2^6
iv	:V-phase current value	x 2^6
iw	:W-phase current value	x 2^6
id	:V-phase current value	x 2^6
iq	:W-phase current value	x 2^6
sin table		x 2^6

to provide operation precision.

```
/*******************************************************************************
*      Reference program of Three-phase / Two-phase conversion
*
*   (C)2003. Renesas Technology Corp. and Renesas Solutions Corp.,
*   All rights reserved.
*******************************************************************************/
void dq_samp(void);

signed short iu_6w;         /* U-phase current value[2^6]             */
signed short iv_6w;         /* V-phase current value[2^6]             */
signed short iw_6w;         /* W-phase current value[2^6]             */
signed short work_6w;       /* work[2^6]         */
signed short id_6w;         /* d shaft current[2^6]                   */
signed short iq_6w;         /* q shaft current[2^6]                   */
signed short sin_pt;        /* sin table pointer                      */

void dq_samp()
{
       iw_6w=-(iu_6w+iv_6w);       /* W-phase current value[2^6] =-(U-phase current
                                   value[2^6]+V-phase current value[2^6]) */
       work_6w=(signed short)((
            (sin_tbl[sin_pt+90]*(long)iu_6w)
            +(sin_tbl[sin_pt+90+240]*(long)iv_6w)
             +(sin_tbl[sin_pt+90+120]*(long)iw_6w)
             )>>15);
                     /* work[2^6]=(                                       */
                     /*           cosθ x U-phase current value[2^6]       */
                     /*           +cos(θ+240) x V-phase current value[2^6]  */
                     /*           +cos(θ+120) x W-phase current value[2^6]) */
                     /*     *sin table is [2^15]                          */

       id_6w=(short)(((long)work_6w*26752)>>15); /* d shaft current[2^6]=work[2^6]
                                                 x √2/3       */

       work_6w=(signed short)((
            -(sin_tbl[sin_pt]*(long)iu_6w)
            -(sin_tbl[sin_pt+240]*(long)iv_6w)
             -(sin_tbl[sin_pt+120]*(long)iw_6w)
             )>>15);

                     /* work[2^6]=(                                       */
                     /*           -sinθ x U-phase current value[2^6]      */
                     /*           -sin(θ+240) x V-phase current value[2^6]  */
                     /*           -sin(θ+120) x W-phase current value[2^6]) */
                     /*     *sin table is [2^15]                          */
       iq_6w=(short)(((long)work_6w*26752)>>15); /* q-shaft
                                          current[2^6]=work[2^6] x √2/3     */
}
```

7.3.2 Two-phase / Three-phase convert module

The example of Two-phase / Three-phase change is written in the following.

$$\begin{pmatrix} V\,u \\ V\,v \\ V\,w \end{pmatrix} = \sqrt{\frac{2}{3}} \begin{pmatrix} \cos\theta & -\sin(\theta) \\ \cos(\theta+240) & -\sin(\theta+240) \\ \cos(\theta+120) & -\sin(\theta+120) \end{pmatrix} \begin{pmatrix} V\,d \\ V\,q \end{pmatrix}$$

Calculated in the sample program of the rest as

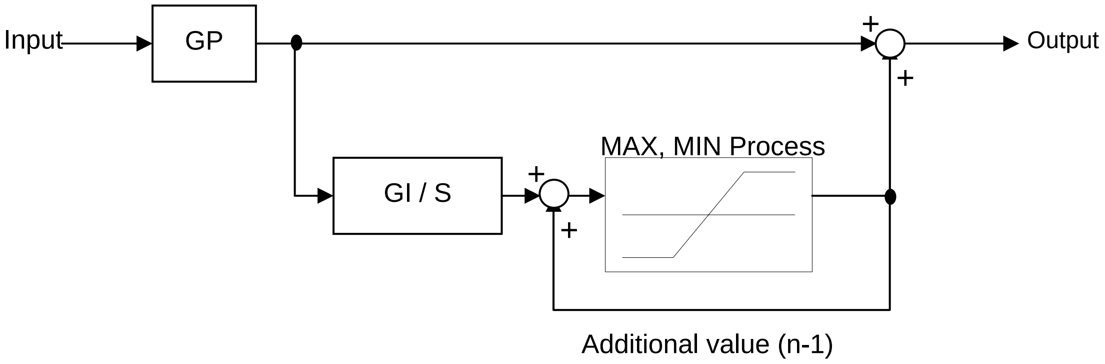
V d	:d-shaft voltage value	x 2^6
V q	:q-shaft voltage value	x 2^6
V u	:U-phase voltage order value	x 2^6
V v	:V-phase voltage order value	x 2^6
V w	:W-phase voltage order value	x 2^6
sin table		x 2^6

to provide operation precision.

```
/*******************************************************************************
*      Reference program of Two-phase / Three-phase conversion
*
*   (C)2003. Renesas Technology Corp. and Renesas Solutions Corp.,
*   All rights reserved.
******************************************************************************/
void uvw_samp(void);

signed short vd_6w;         /* d-shaft voltage[2^6]            */
signed short vq_6w;         /* q-shaft voltage[2^6]            */
signed short vu_6w;         /* U-phase voltage order[2^6]      */
signed short vv_6w;         /* V-phase voltage order[2^6]      */
signed short vw_6w;         /* W-phase voltage order[2^6]      */
signed short work_6w;       /* work[2^6]                       */
signed short sin_pt;        /* sin table pointer               */


void uvw_samp()
{
       work_6w=(signed short)((
              (sin_tbl[sin_pt+90]*(long)vd_6w)
              -(sin_tbl[sin_pt]*(long)vq_6w)
              )>>15);
                            /* work[2^6]=(                                   */
                            /*           cosθ x d-shaft voltage[2^6]        */
                            /*           -sinθ x q-shaft voltage[2^6])      */
                            /*     *sin table is[2^15]                       */

       vu_6w=(short)(((long)work_6w*26752)>>15); /* U-phase voltage
                                                 order[2^6]=work[2^6] x √2/3 */

       work_6w=(signed short)((
              (sin_tbl[sin_pt+90+240]*(long)vd_6w)
              -(sin_tbl[sin_pt+240]*(long)vq_6w)
              )>>15);
                            /* work[2^6]=(                                   */
                            /*           cos(θ+240) x d-shaft voltage[2^6]  */
                            /*           -sin(θ+240) x q-shaft voltage[2^6])*/
                            /*     *sin table is[2^15]                       */

       vv_6w=(short)(((long)work_6w*26752)>>15); /* V-phase voltage
                                                 order[2^6]=work[2^6] x √2/3 */

       vw_6w=-(vu_6w+vv_6w);
              /* W-phase voltage order[2^6]=                                 */
              /*-(U-phase voltage order[2^6] + V-phase voltage order[2^6])   */
}
```

7.3.3 PI control module

The example of the PI control is written in the following.



Calculated in the sample program of the rest as

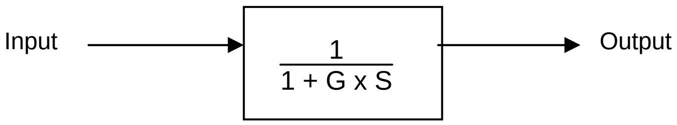
Input data	x 2^6
Output data	x 2^6
Gain P	x 2^{10}
Gain I / Control cycle	x 2^{18}

to provide operation precision.

```
/******************************************************************************
*       Reference program of PI control
*
*   (C)2003. Renesas Technology Corp. and Renesas Solutions Corp.,
*   All rights reserved.
******************************************************************************/
void    PI_samp(void);

signed short data_in_6w;            /* Input data[2^6]                  */
signed short data_out_6w;           /* Output data[2^6]                 */
signed short GP_10w;                /* Gain P[2^10]                     */
signed short GI_t_18w;              /* (Gain I / Control cycle t)[2^18]       */
signed long sum_16l;                /* Addtional value[2^16]                  */
signed long work1_16l;              /* work1[2^16]                      */
signed long work2_16l;              /* work2[2^16]                      */
#define work2_limit 500             /* work2 limit value 500(Temporary data)*/

void    PI_samp()
{

        work1_16l =(long)data_in_6w * GP_10w;       /*work1[2^16]=Input data[2^6] x
                                                    Gain-P[2^10]*/

        sum_16l = sum_16l + ((work1_16l >> 11) * GI_t_18w) >> 7;
                /* Additional value[2^16]=Additional value[2^16]+((work1[2^16] /
                2^11) x (Gain-I / t)[2^18]) / 2^7 */

        work2_16l = work1_16l + sum_16l;            /* work2[2^16]=work2[2^16]+
                                                    Additional value[2^16] */

        if(work2_16l > work2_limit)
        {
                work2_16l = work2_limit;
        }else
        {
                if(work2_16l < -work2_limit)
                {
                        work2_16l = -work2_limit;
                }
        }

        data_out_6w = (short)(work2_16l >> 10);     /* Output data[2^6]=work2[2^16] /
                                                    2^10 */

}
```

7.3.4 Filter control module

The example of the filter control is written in the following.



Calculated in the sample program of the rest as

Input data	x 2^{6}
Output data	x 2^{6}
Gain-P	x 2^{10}
Gain-I / control cycle	x 2^{18}

to provide operation precision.

```
/*******************************************************************************
*      Reference program of Filter control
*
*   (C)2003. Renesas Technology Corp. and Renesas Solutions Corp.,
*   All rights reserved.
*******************************************************************************/
void   fill_samp(void);

signed short data_in_6w;                  /* Input data[2^6]                  */
signed short fill_out_6w;                 /* Output data[2^6]                 */
signed short G_t21;                       /* 1 / (Gain x Control cycle)[2^21] */
signed long fill_22l;                     /* Additional value[2^22]          */

void   fill_samp()
{
       fill_22l=fill_22l
              +(long)((G_t21*(long)(data_in_6w-fill_out_6w))>>5);
                            /* Additional value[2^16]=Additional value[2^22]
                                   + (1 / (Gain x Control cycle))[2^21]
                                   x (Input data[2^6] – Output data[2^6])    ) /
                                   2^5    */
       fill_out_6w=(short)(fill_22l>>16);        /* Output data[2^6]=Additional
                                                        value[2^22]>>16 */
}
```

8. Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com

E-mail Support

E-mail: support_apl@renesas.com

Data Sheet

M32C/83 group Rev. 1.0

(Use the latest version on the home page: http://www.renesas.com)

REVISION HISTORY	M32C/83 Group Concept of the Three-phase Motor Control Program Application Note

Rev.	Date	Description	
		Page	Summary
1.20	Jul 25, 2003	-	First edition issued

Keep safety first in your circuit designs!

- 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

- 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.
- 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.
- 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).
- 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.
- 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.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- 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.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.