

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.

M16C/28 Group

Solutions of Three-Phase Motor Control Programming

1.0 Abstract

This application note discusses how to use 3-phase motor control timer function.

2.0 Introduction

The explanation of this issue is applicable to M16C/28 Group (M30280M4A-XXXFP).

3.0 Inverter Control

3.1 What Is Inverter Control?

Inverter control voluntarily changes the frequency to drive a motor. For example, 3-phase motors can be driven with waveform that is out of phase by 120 degrees, however, when the motor is powered with 3-phases of commercial power supply, its motor frequency depends on the frequency of commercial power supply .

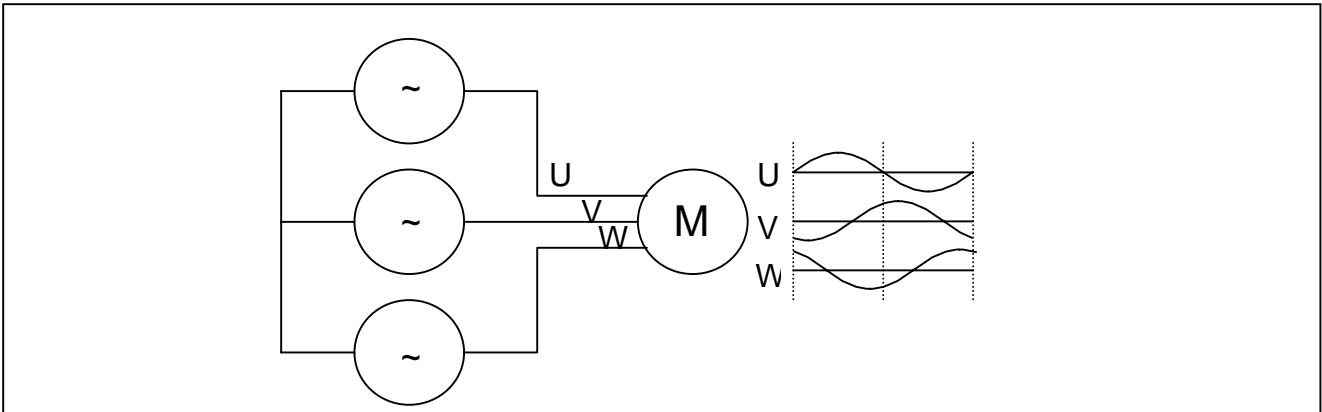


Figure 3.1 3-phase motor drive

Inverter control converts commercial AC power supply to DC power supply first, and then, generates motor drive frequency by switching transistors. This switching is controlled by MCU. Therefore, changing switching interval can generate voluntary motor drive frequency

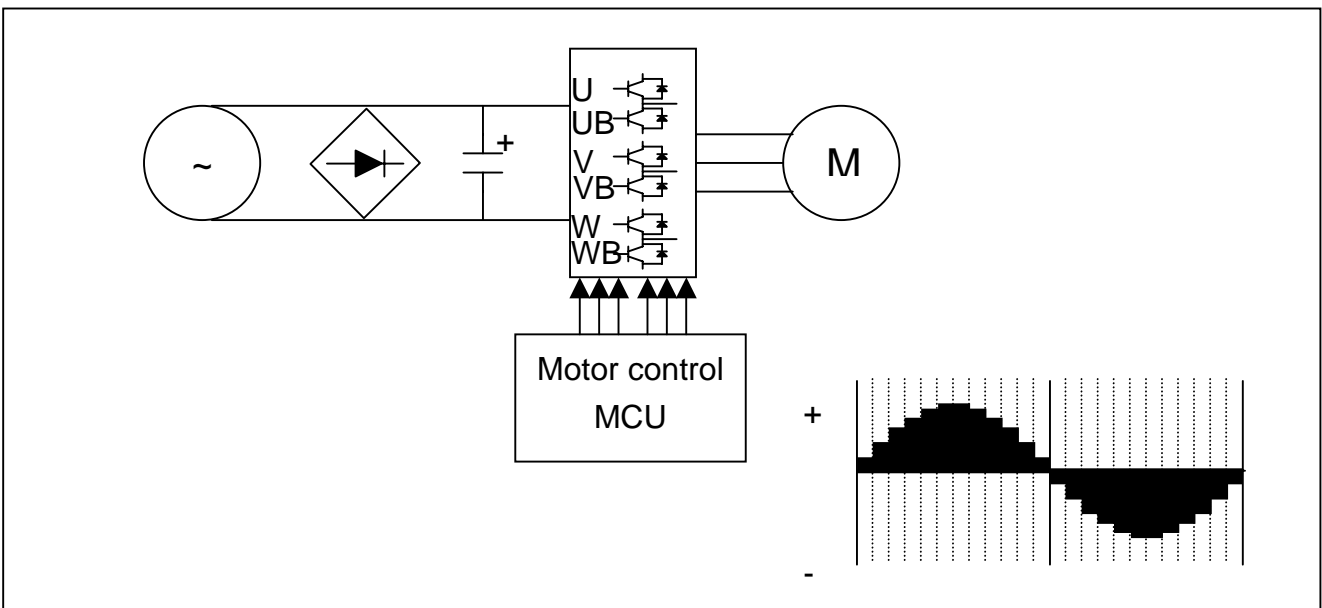


Figure 3.2 Example of inverter control using MCU

3.2 Output Waveforms

MCUs can not directly output a sinusoidal wave or high voltage for driving a motor. An interface circuit is therefore necessary between the MCU and the motor, as shown in Figure 3.3. The port output of the MCU is connected to U, V, W, B, VB, and WB terminals, as shown in the figure.

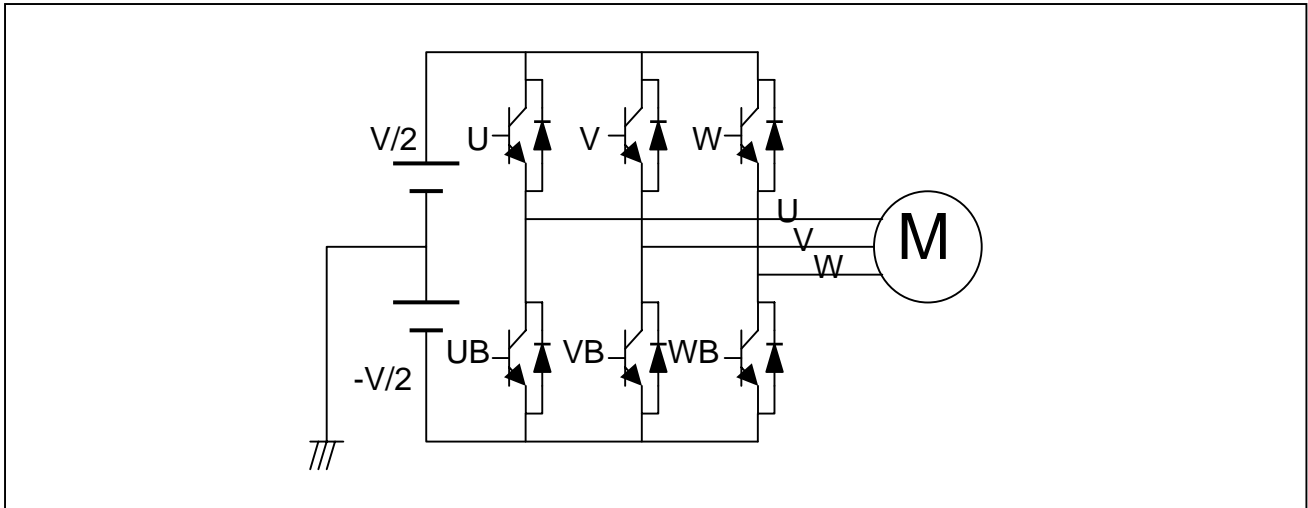


Figure 3.3 Power transistors such as IPM and IGBT

When alternating ON and OFF signals are input to U and UB terminals of the transistors in Fig. 3.4, the output voltages of the transistors are also altered, and the alternating current wave (square wave) is respectively generated in the U phase.

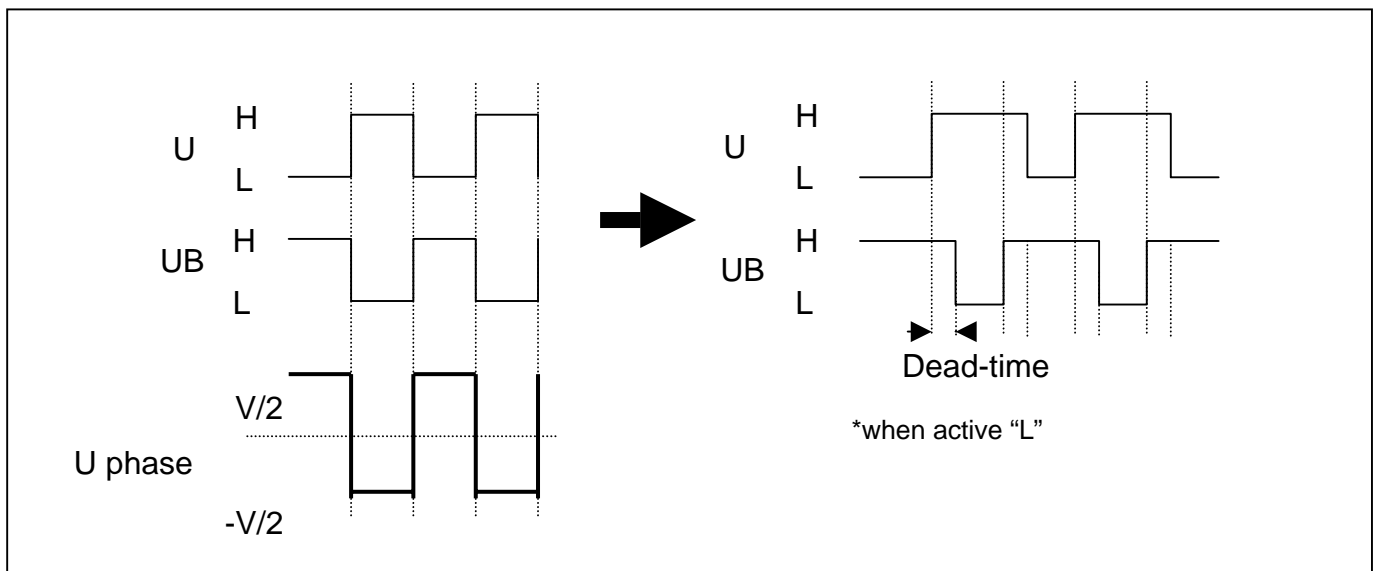


Figure 3.4 MCU output and generated waveforms

If both U and UB are turned on at the same time, the short circuit current goes through the transistors and causes a direct short-circuit, destroying the transistors. This is called an arm short-circuit.

To avoid an arm short-circuit, the 3-phase motor control timer makes a switching time delay, preventing both transistors from being switched ON at the same time. This time delay will be referred to as “dead-time” in this document. If you program the dead-time interval generation in the initial setup procedure of the motor drive control process, the MCU will automatically generate the corresponding dead-time.

Quantizing the sinusoidal wave by time domain and converting each unit of charging energy to the H (or L) length of the charging voltage will generate the equivalent sinusoidal waveform.

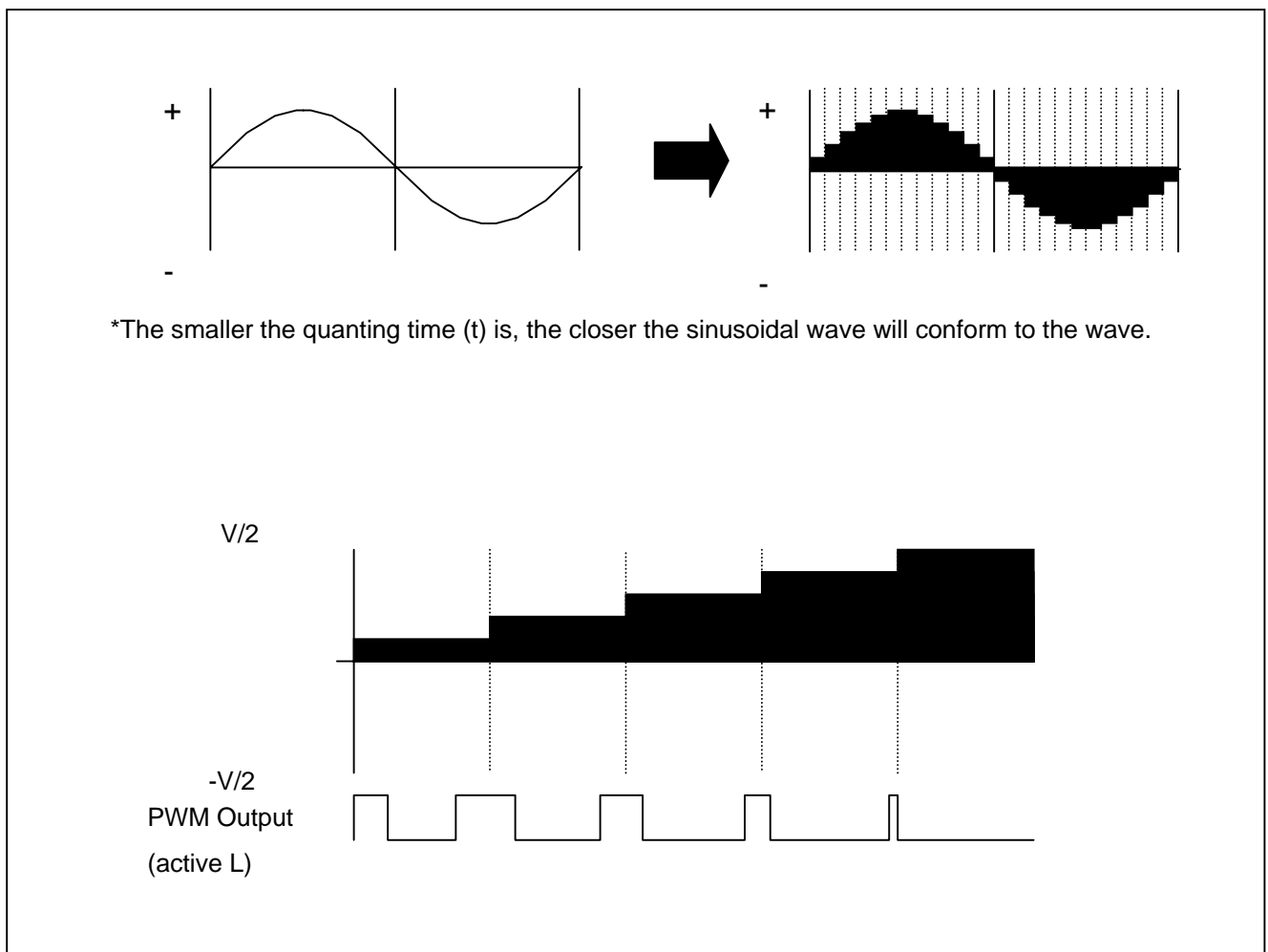


Figure 3.5 Quantized square waveform (by time unit)

4.0 How To Use Timer Function For Three-phase Motor Control

4.1 Three-Phase Waveform Output Methods

This chapter explains basic methods of 3-phase waveform output.

4.2 Carrier Frequency

The carrier frequency is the reference of PWM pulse width of the output voltage.

Assuming that the sinusoidal wave overlaps the carrier, the distance between the both waves becomes the output phase pulse width.

There are two methods of carrier frequencies; sawtooth wave modulation method and triangular (back-to-back sawtooth) wave modulation method.

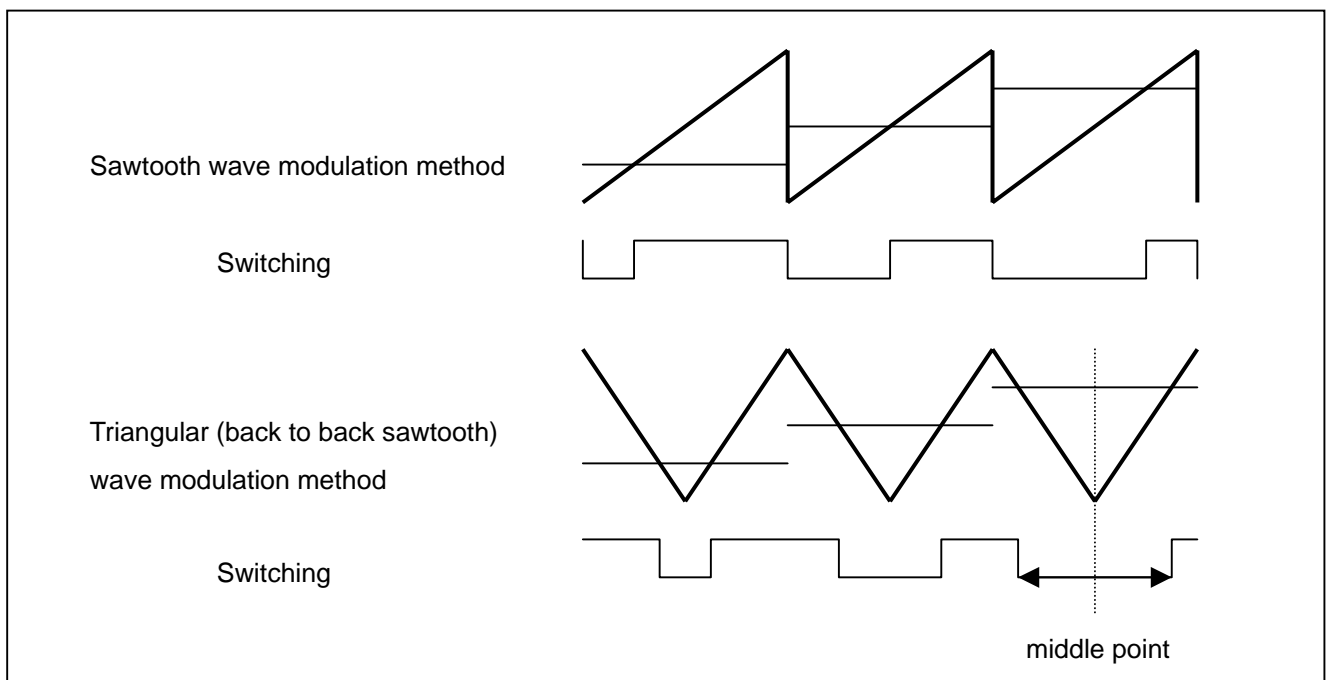


Figure 4.1 Methods of carrier frequency

Sawtooth wave modulation method changes duty with starting point of carrier period as a reference. On the other hand, triangular wave modulation changes duty with the middle point as a reference.

4.3 PWM Emulation Methods

3-phase motor control timer basically generates one full carrier if sawtooth wave modulation and a half carrier if triangular. The peak timing of the carrier period is generated by TB2. The TB2 underflow trigger starts TAI (i = 4,1,2) one shot. This TAI determines PWM duty.

In case of using sawtooth wave modulation, set the trigger of dead-time timer at rising and falling edge of TAI. In case of using triangular wave modulation, set the trigger at only rising edge of TAI.

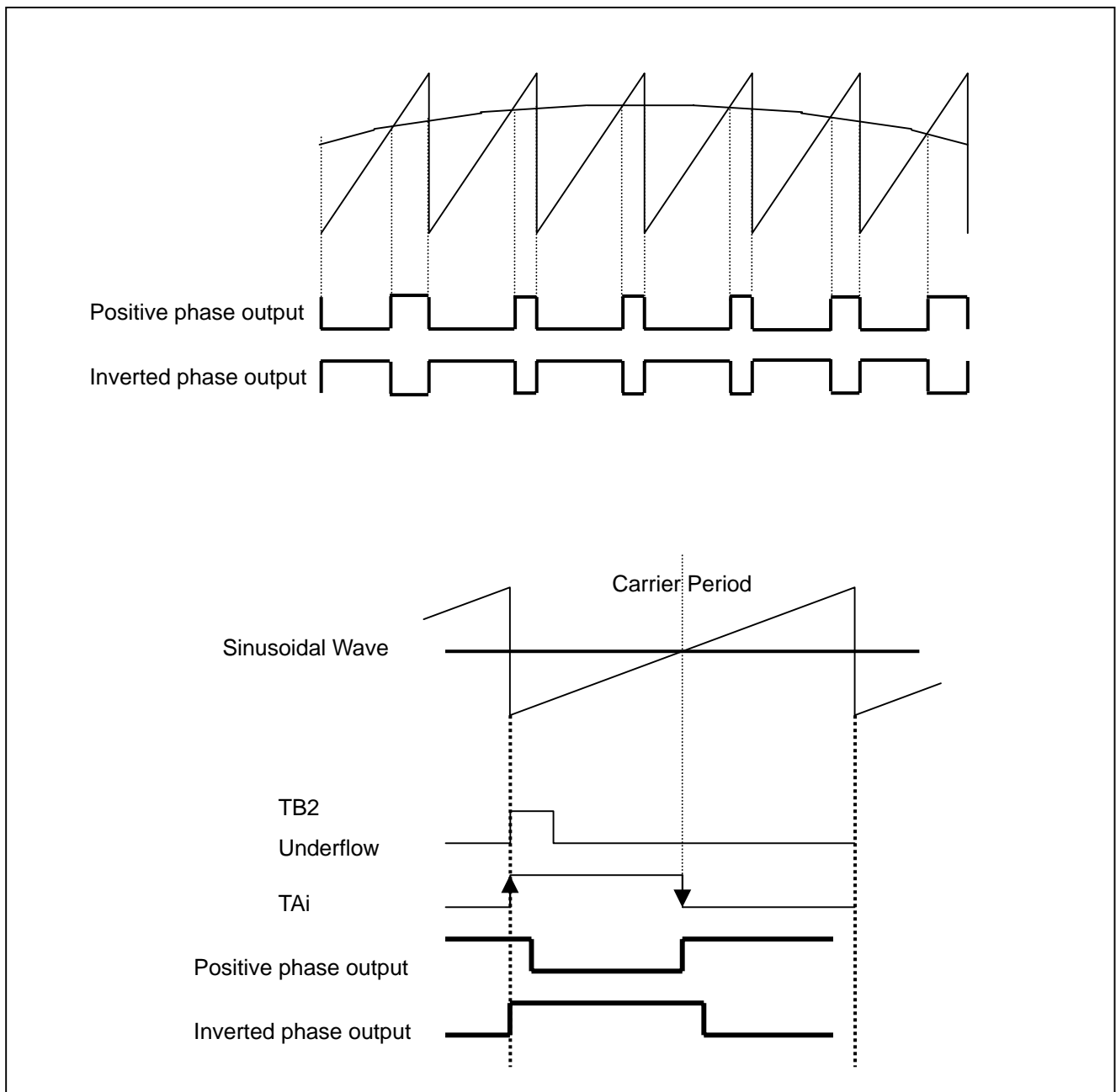


Figure 4.2 PWM in sawtooth wave modulation

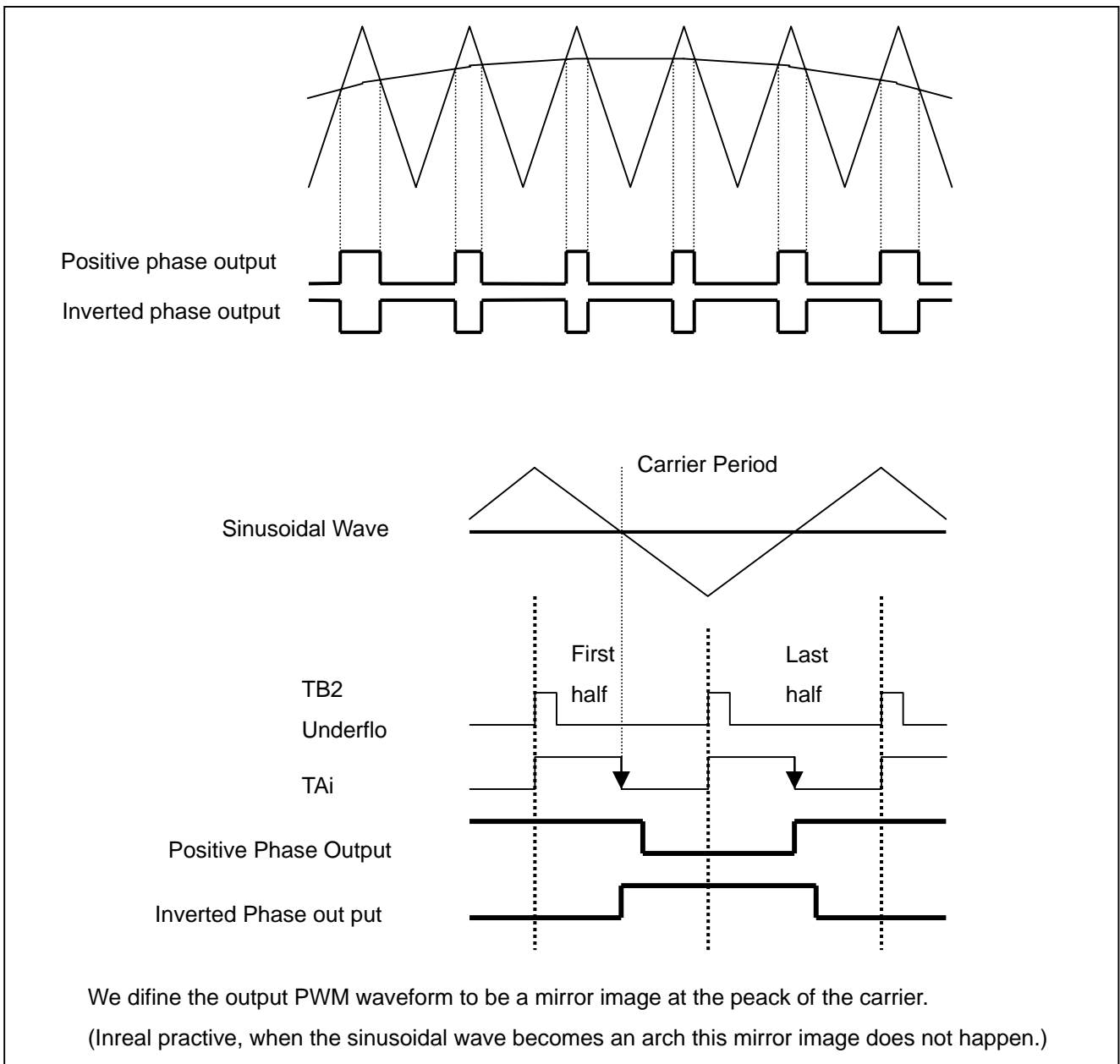


Figure 4.3 PWM in triangular sawtooth modulation

If you use privously described triangular modulation method to output waveform, you have to set timing by generating interrupt every half-period of the carrier and set TAI again.

To reduce this load, there is a function that sets TAI values for both first half and last half, in one full-carrier.

This function is called 3-phase mode 1.

3-phase mode 1 enables to set value to TAI alternatively from two registers.

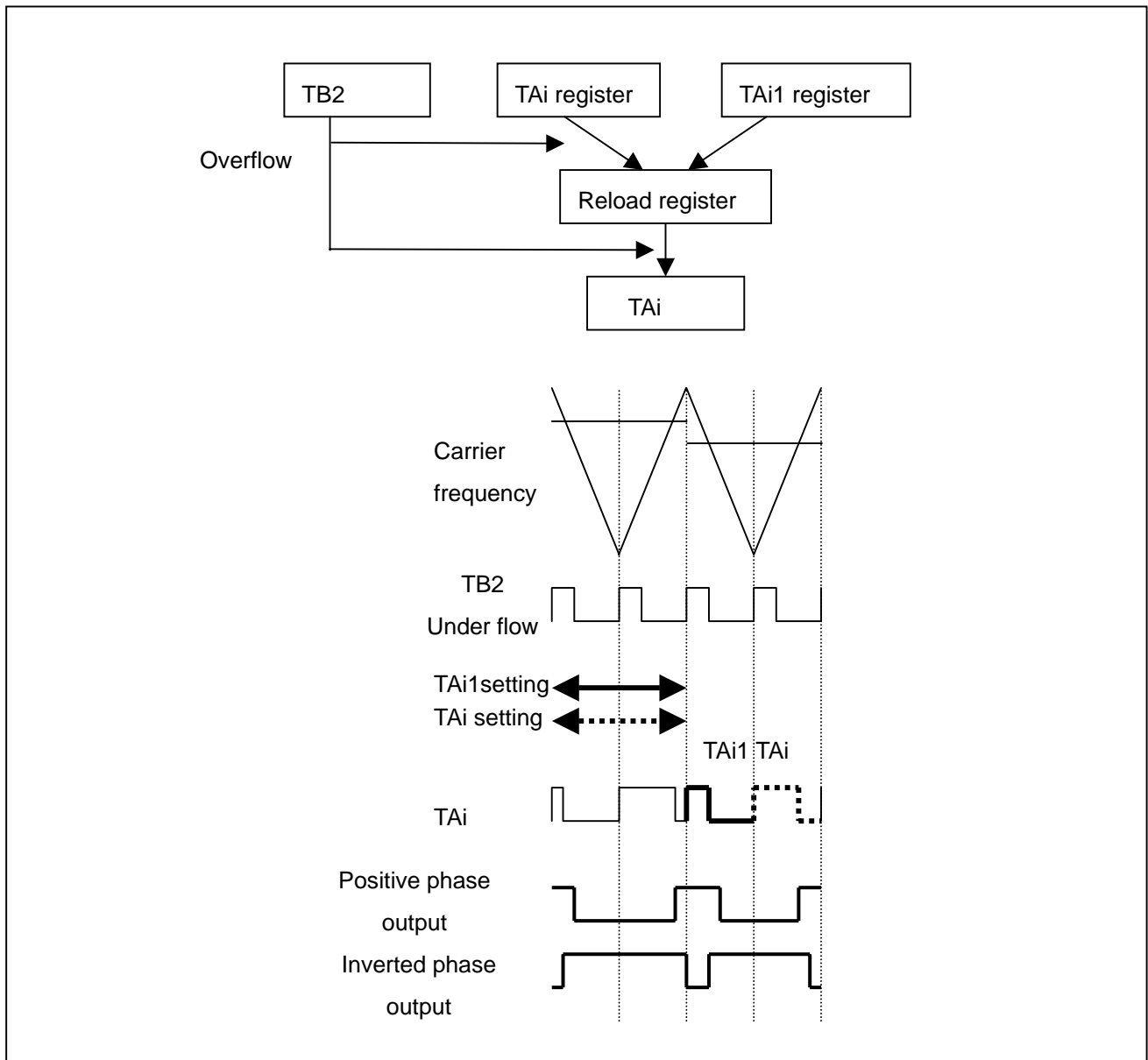


Figure 4.4 Relationship between TB2 and TAI/TAI1 registers in 3-phase mode 1

*TAI1 register cannot be updated near the TB2 underflow. Please refer to technical news for the detail.

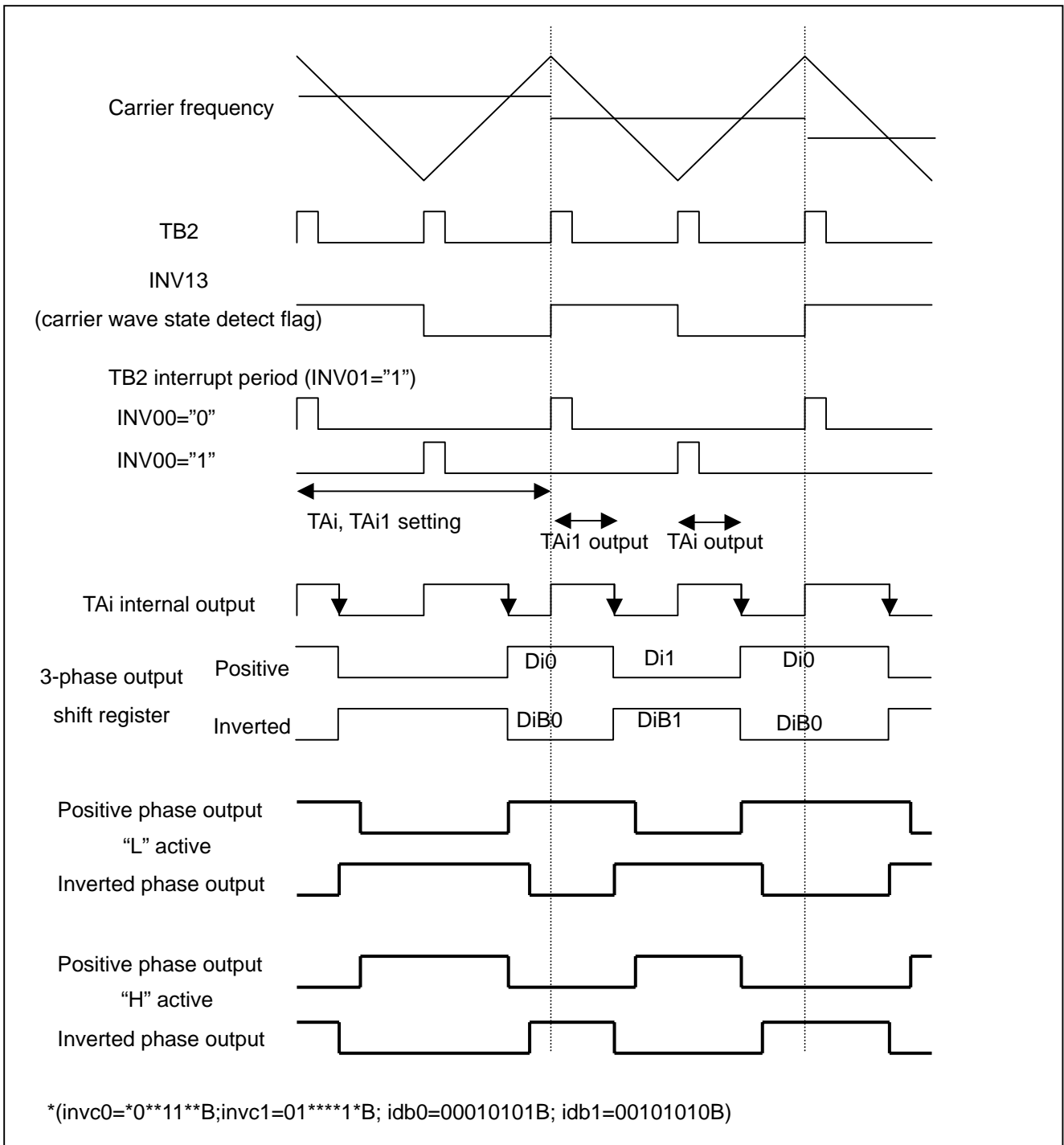


Figure 4.5 3-phase mode1 setting and output

If an initial value (INV01="0") is set to interrupt valid output specification and interrupt cycle setting counter(ICTB2) is set to "2", TB2 interrupt is generated at the same timing with INV00="1" as indicated in the figure 4.5, at the initial state. If the same timing with INV00="0" is required, please set odd number to interrupt cycle setting counter (ICTB2), for the first cycle only.

4.4 Carrier Frequency

To change carrier frequency, the TB2 interrupt period must be changed. However, when triangular wave modulation is used, two TB2 underflows become one carrier period. Therefore, special caution should be paid to carrier period modification.

When you use triangular wave modulation to control by every full-period of the carrier, (invc0=*0*****B and invc0=0*****1*B), set TB2 reload timing to Timer A output after odd numbered times (TB2sc=*****1B), and when you use rectangular wave modulation to control by every half period of carrier (invc0=*0*****B and invc0=1*****1*B or invc0=*****B and invc0=*****0*B), set TB2 reload timing to TB2 underflow (TB2sc=00000000B), so that variation of carrier period becomes in line with the TAI set timing.

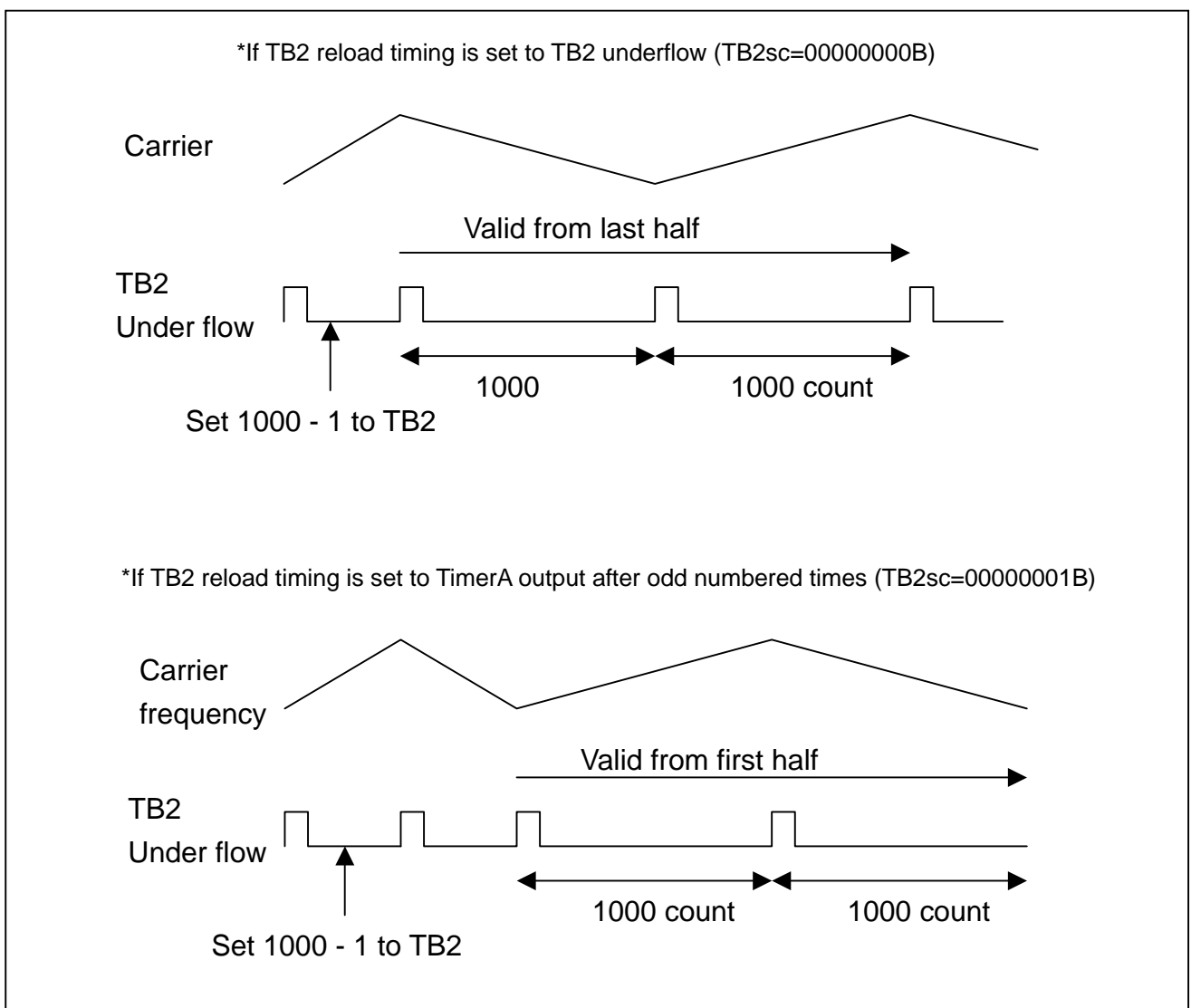


Figure 4.6 Carrier period setting and output

4.5 How To Use Three-phase Output Buffer Register

By updating 3-phase output buffer register value, output level can be fixed (continuous on, continuous off) for a certain period, or same leveled waveform can be output for both positive and inverted phase.

This function enables to realize 2-phase modulation output (See Chapter 5) and trapezoidal waveform output (See Chapter 6).

*The 3-phase output buffer register contents are transferred to the shift register at the first TB2 underflow after updating.

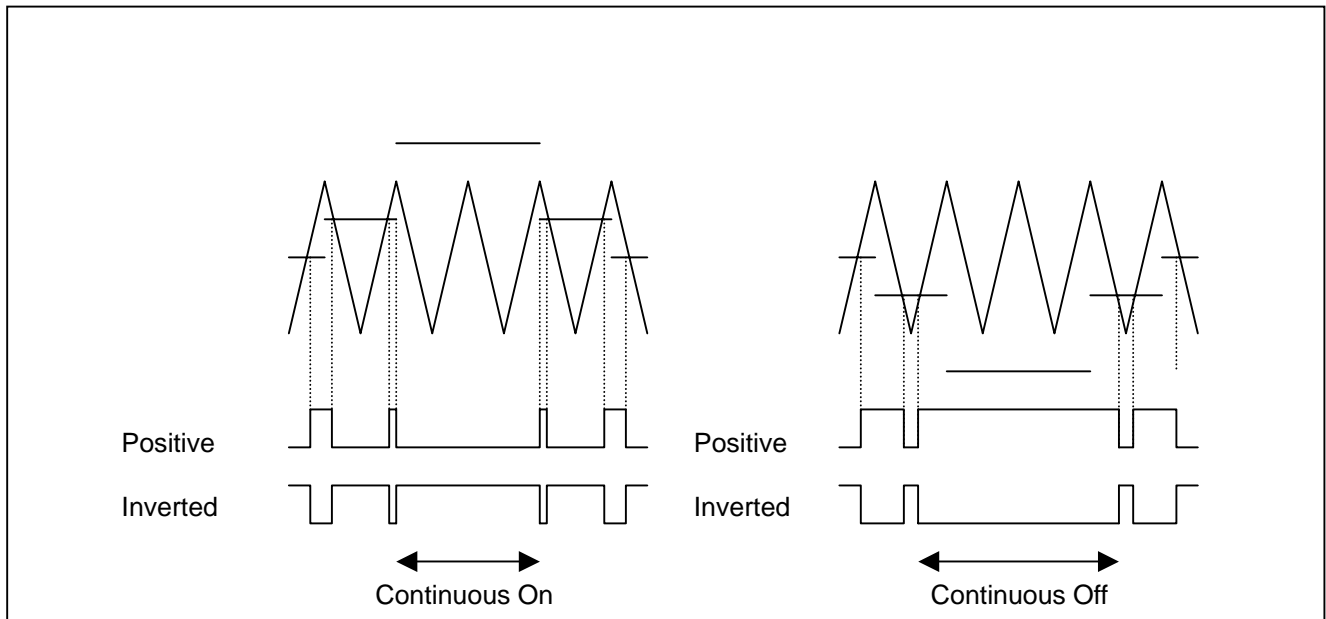


Figure 4.7 Waveform generated by 2-phase modulation output (See chapter 5)

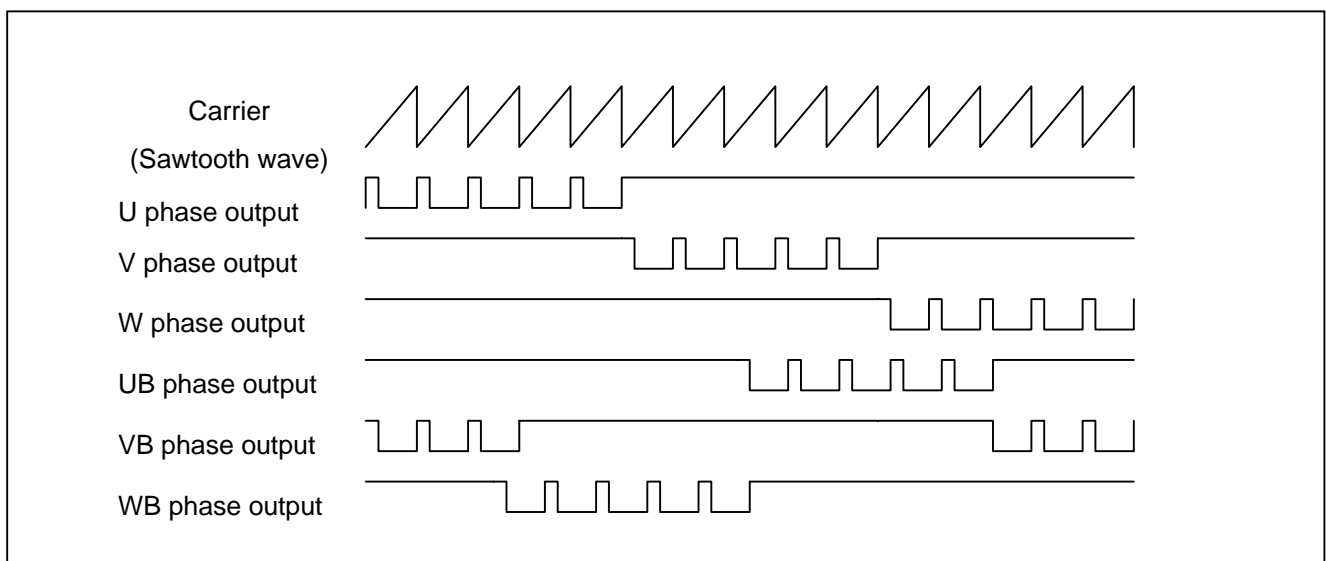


Figure 4.8 Waveform generated by trapezoidal waveform output (See Chapter 6)

4.6 Notes Concerning PWM Data Setup

The following are notes concerning the TAI setting and the 3-phase motor control timer.

1. Lower limit of TAI Value

If registers TAI is set to "0" (in case of dead-timer f2, "0", "1"), the internal TAI timer will not start counting and the falling edge will not be generated, and as a result, the output wave keeps its current state. Setting TAI0 or TAI 1 to "0" and "1" should be avoided except in special cases.

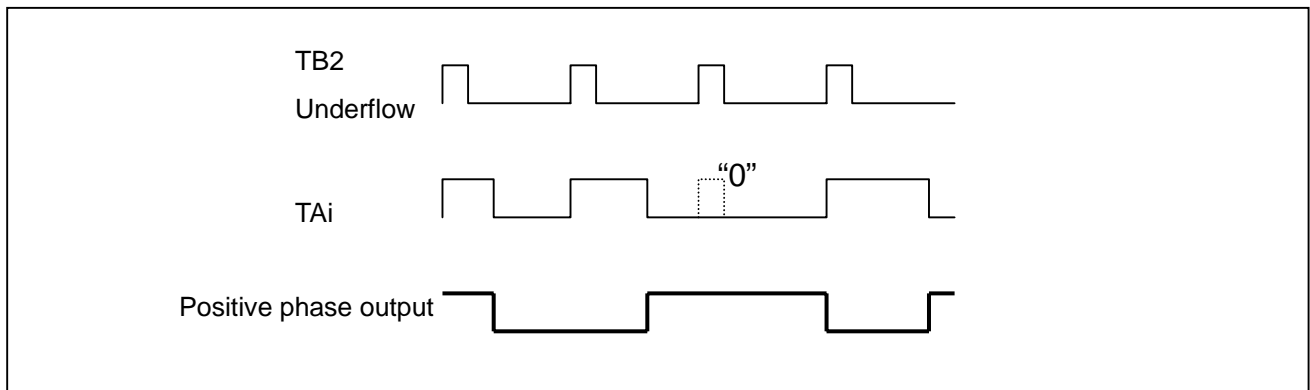


Figure 4.9 Positive phase output when TAI =0

2. Upper Limit of Tai Value

If the value larger than the TB2 value (in case of dead-timer f2, TB2 value-1) is set to registers TAI, the internal TAI timer continues counting even when it enters the TB2 periods. This inhibits the generation of the falling edge, and the output wave keeps its current state. The result, as shown in Fig.4.10, is a reversed wave. Please avoid setting larger values except in special cases.

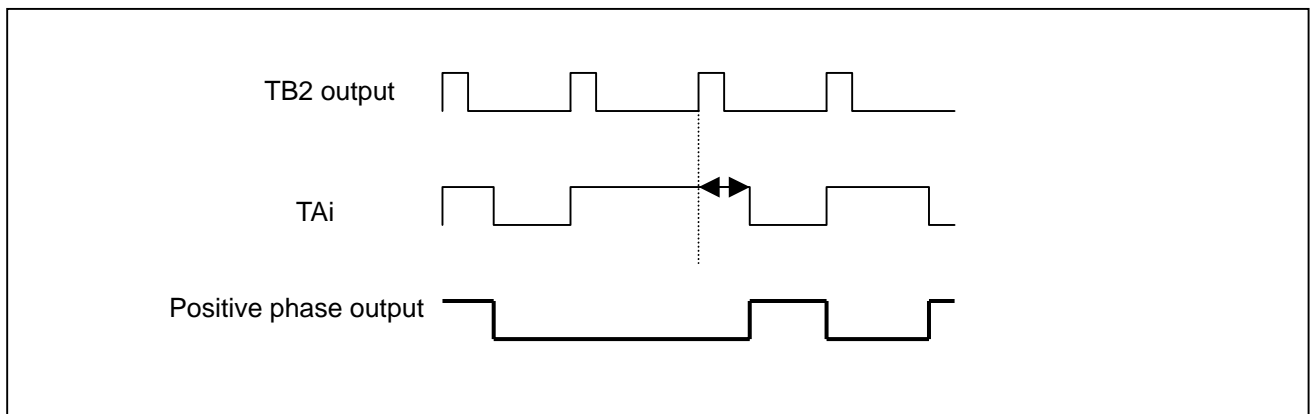


Figure 4.10 Output waveform when TAI>TB2

3. Re-starting Dead-time Timer

By setting the PWM data, the dead-time timer is re-started while it is counting, resulting in a dead-time overlap.

Triangular (back-to-back sawtooth) wave modulation method: dead-time f1

$$((TB2 \text{ value} + 1) - TAI1 \text{ value}) + TAI \text{ value} < \text{dead-time value}$$

$$TAI1 \text{ value} + ((TB2 \text{ value} + 1) - TAI \text{ value}) < \text{dead-time value}$$

Sawtooth wave modulation method: dead-time f1

$$((TB2 \text{ value} + 1) - TAI \text{ value}) - 1 < \text{dead-time value}$$

$$TAI \text{ value} - 1 < \text{dead-time value}$$

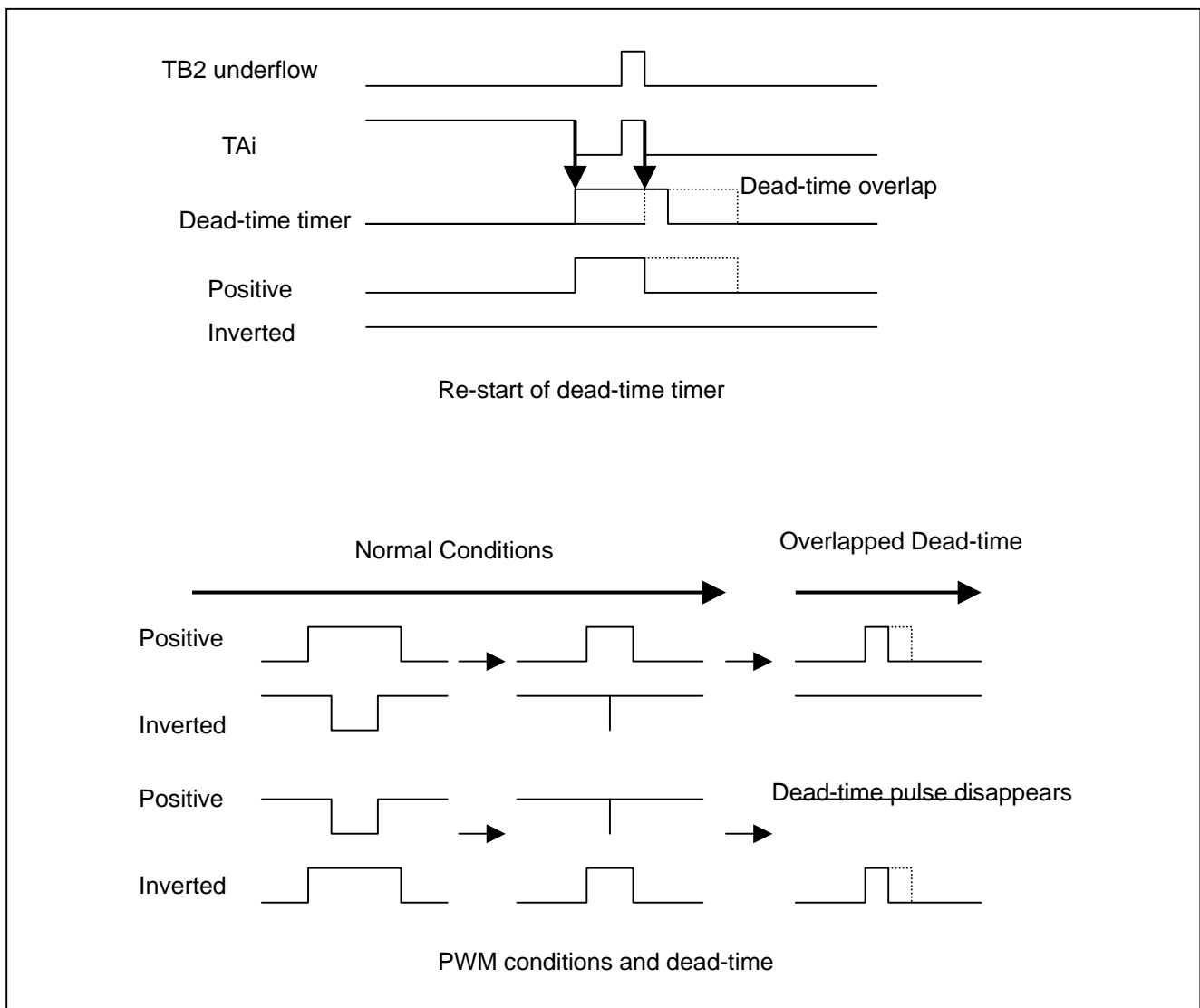


Figure 4.11

4. Output waveform after updating 3-phase output buffer register

When dead-time timer setting trigger select bit is "0" (invc1=000****B), and waveform output is changed from fixed to from output to PWM output by 3-phase output buffer register.

When positive phase is changed from "L fixed" to PWM output and $T_{Ai} > T_{B2}$ – dead-time, the output dead-time can be shortened.

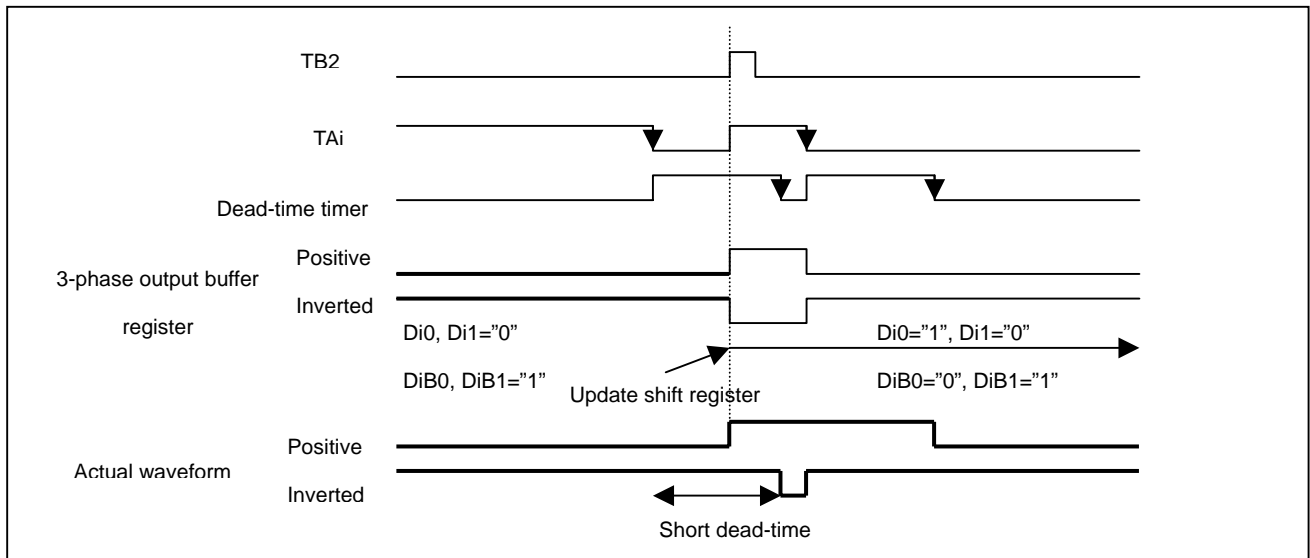


Figure 4.12 Waveform with short dead-time

When you change from fixed-wave output to PWM output by updating 3-phase output buffer register, max value should be set to TAI and TAI1 data (if dead-timer f1=TB2 value, if f2=TB2 value-1)

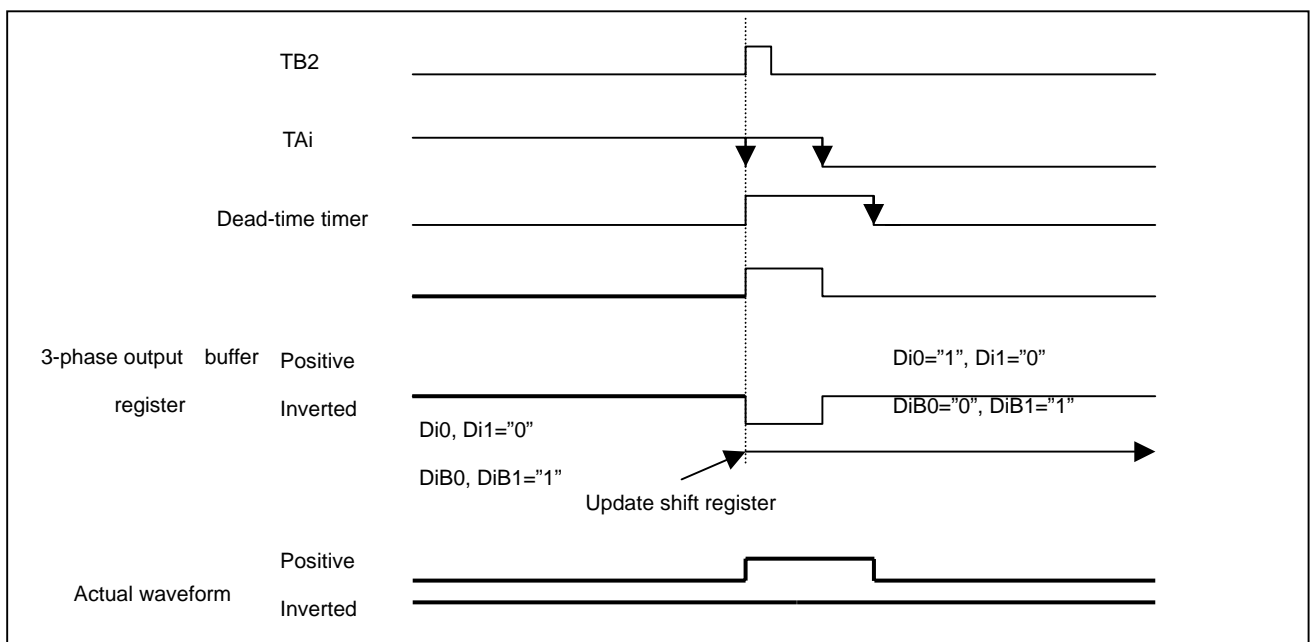


Figure 4.13 Waveform when setting max value to TAI and TAI1

5.0 Three-phase sinusoidal wave output

5.1 How to output Three-phase waveform

This chapter explains the methods of 3-phase sinusoidal waveform output by using 3-phase motor control timer.

5.2 Output methods by using Three-phase motor control timer

Select triangular modulation and 3-phase mode1 of 3-phase motor control timer function.

5.3 PWM Data Calculations

The following explains an example of how to calculate the TAI setting value.

The reference of output waveform should be Sine 0°=duty 50%, and TAI setting value will be a plus or minus of this reference.

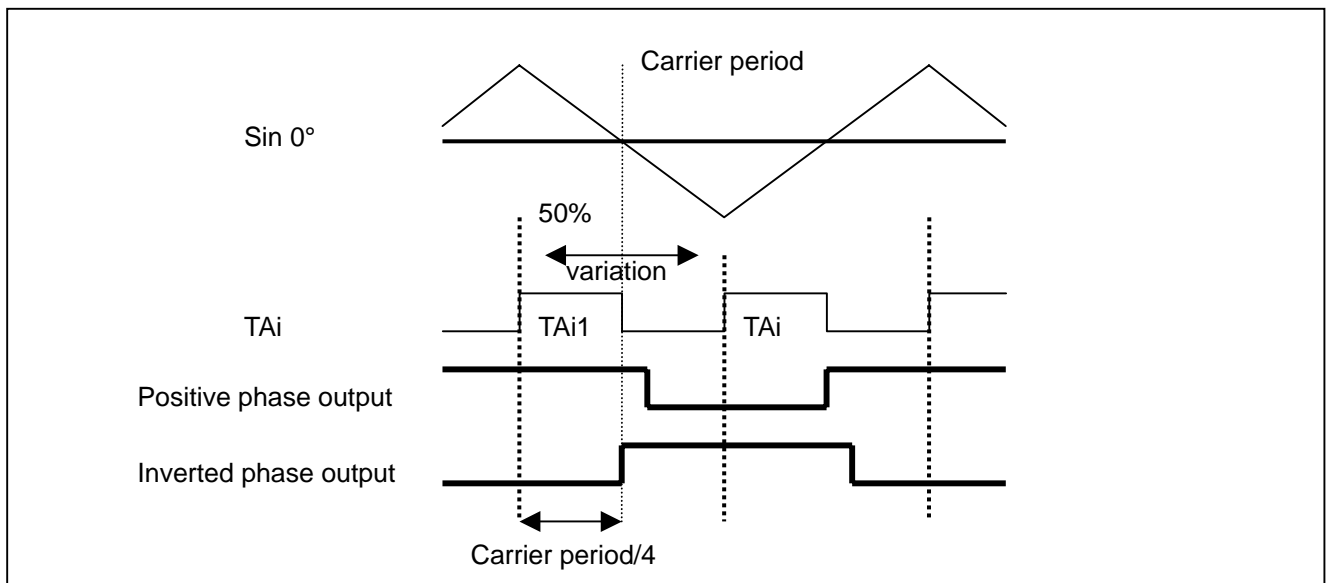


Figure 5.1 Relationship of TAI and PWM data

$$\text{Duty } 50\% = \text{carrier period}/4$$

$$\text{TAI1 data} = \text{carrier period}/4 - \text{variation value}$$

Since sinusoidal wave is $-1 \sim 0 \sim 1$, the waveform becomes as indicated in the below figure.

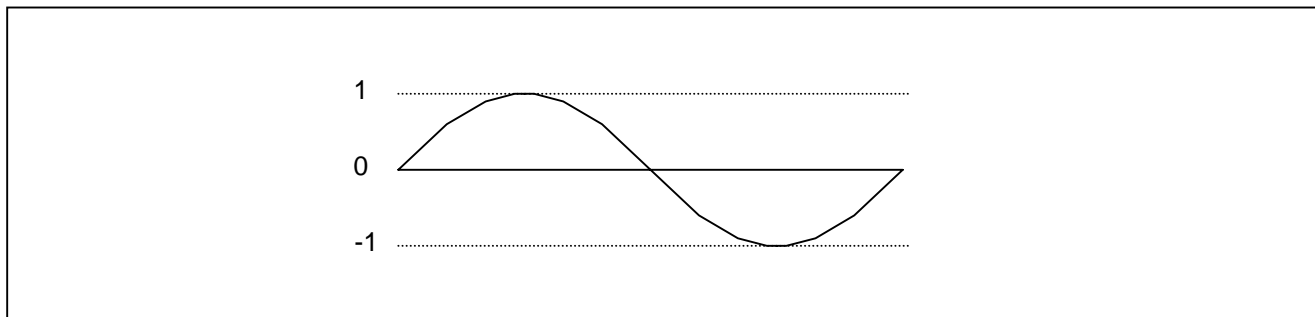


Figure 5.2 Sinusoidal wave

The value X that makes a duty of $-50 \sim 0 \sim 50\%$ is calculated as:

$$X = 50\% \times \sin N^\circ$$

$$X = \text{carrier period} / 4 \times \sin N^\circ$$

Therefore, the setting value of the width can be calculated as:

$$\text{TAi1data} = \text{carrier period} / 4 - \text{carrier period} / 4 \times \sin N^\circ$$

$$\text{TAi data} = \text{carrier period} / 2 - \text{TAi1 data}$$

When current control is performed by physics operation such as a vector operation, TAI data should be calculated as:

$$\text{TAi1data} = \text{carrier period} / 4 - \text{the calculated current control value} \times \text{the constant}$$

$$\text{TAi data} = \text{carrier period} / 2 - \text{TAi data}$$

5.4 Variation of Load Output Wave

The PWM load value is obtained by multiplying the pre-calculated PWM wave in each phase by the modulation rate.

$$\text{PWMi data} = \frac{\text{carrier frequency}}{4} \pm \frac{\text{carrier frequency} \times \sin N^\circ}{4} \times \text{modulation rate}$$

For example, if V/F control is used, the modulation value is varied in combination with the variation of the output frequency, and the relationship of the PWM wave length (V) and the output wave is controlled.

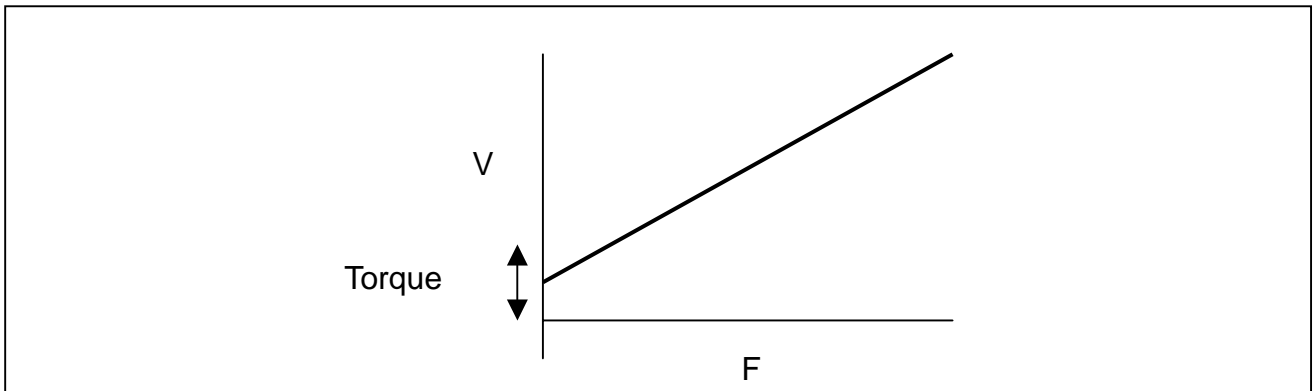


Figure 5.3 V/F control

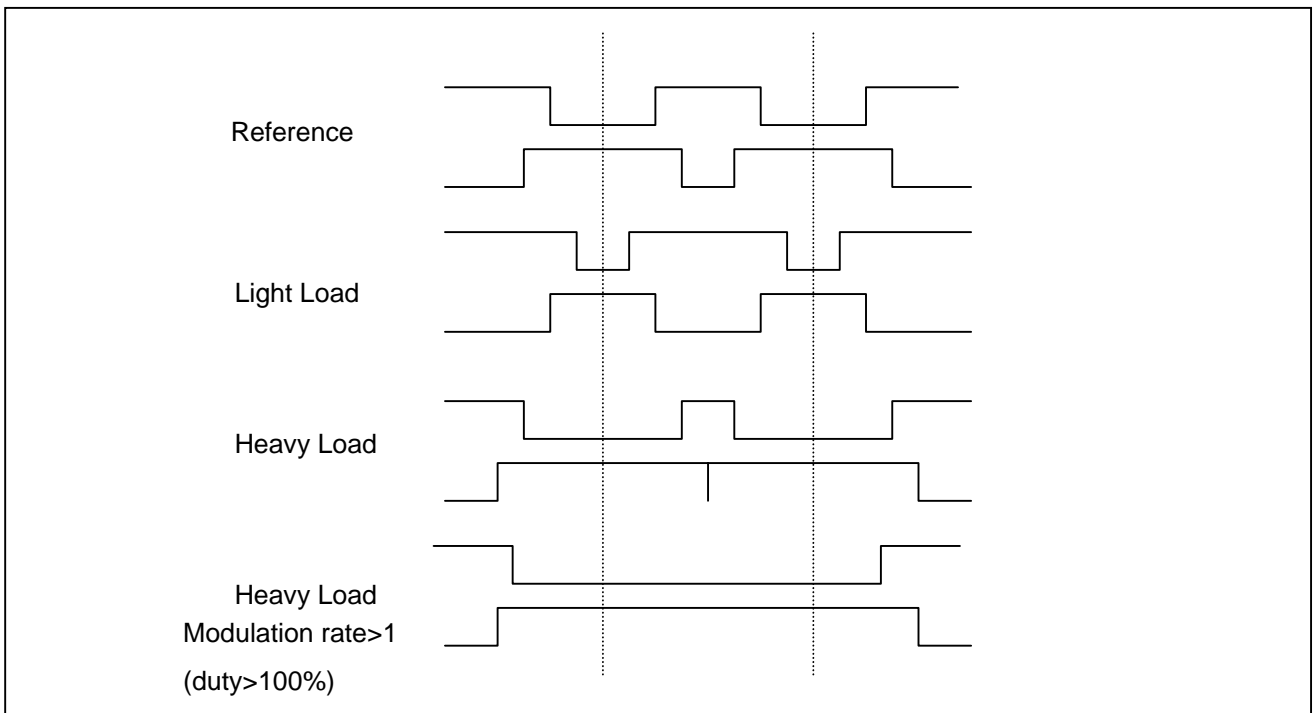


Figure 5.4 Relationship of modulation rate and output wave

5.5 Two-Phase Modulation Output (continuous on, continuous off)

When PWM duty is wider or narrower than the regular duty width, on/off control that is longer than the carrier period is carried out to reduce the dead-time loss.

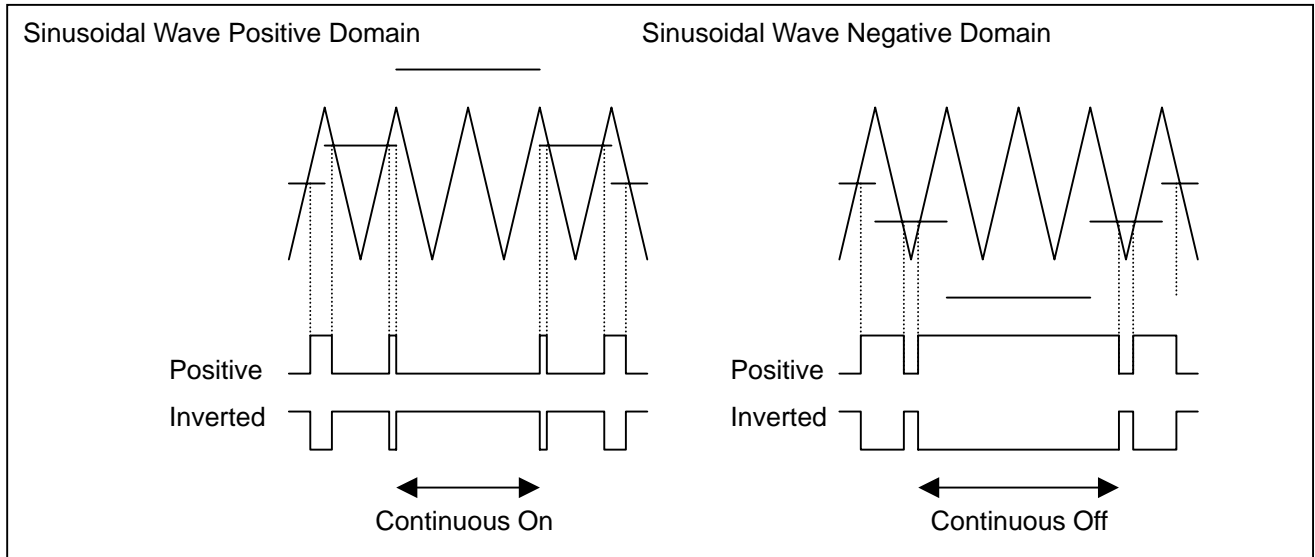


Figure 5.5 Waveform of 2-phase modulation output

5.5.1 Continuous on/off waveform output realized by updating Three-phase output buffer register

As shown in the figure below, continuous ON/OFF waveform output can be realized by updating the content of 3-phase output buffer register at right timing.

In this method, you are directly updating the contents of the 3-phase output buffer register, and updating the wrong bit can cause a short-circuit wave. Therefore, make sure you always set the simultaneous positive/inverted phase low output function (invc0 bit 4) to "1" in order to prevent a short circuit.

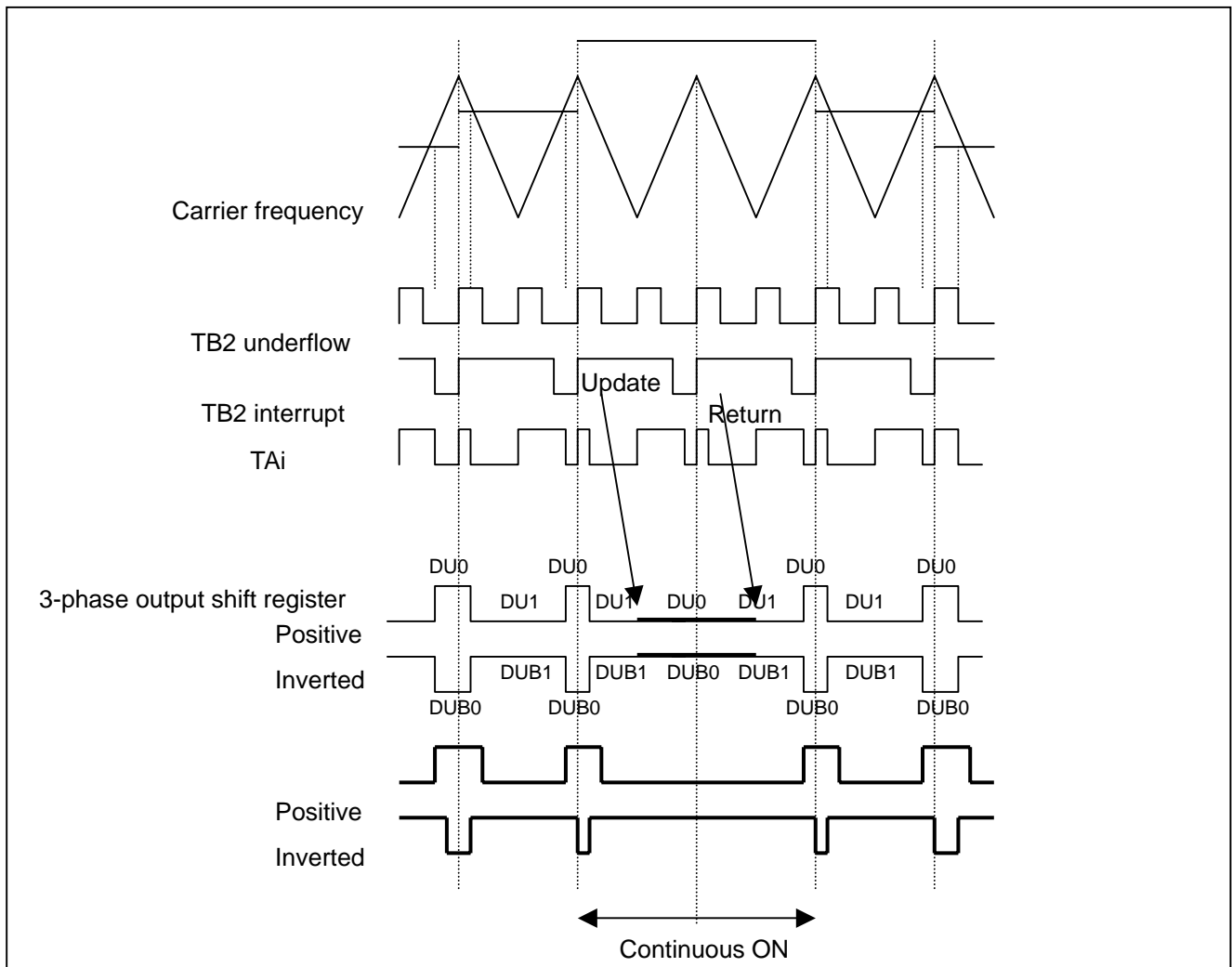


Figure 5.6.1 Timing of writing the 3-phase output buffer register

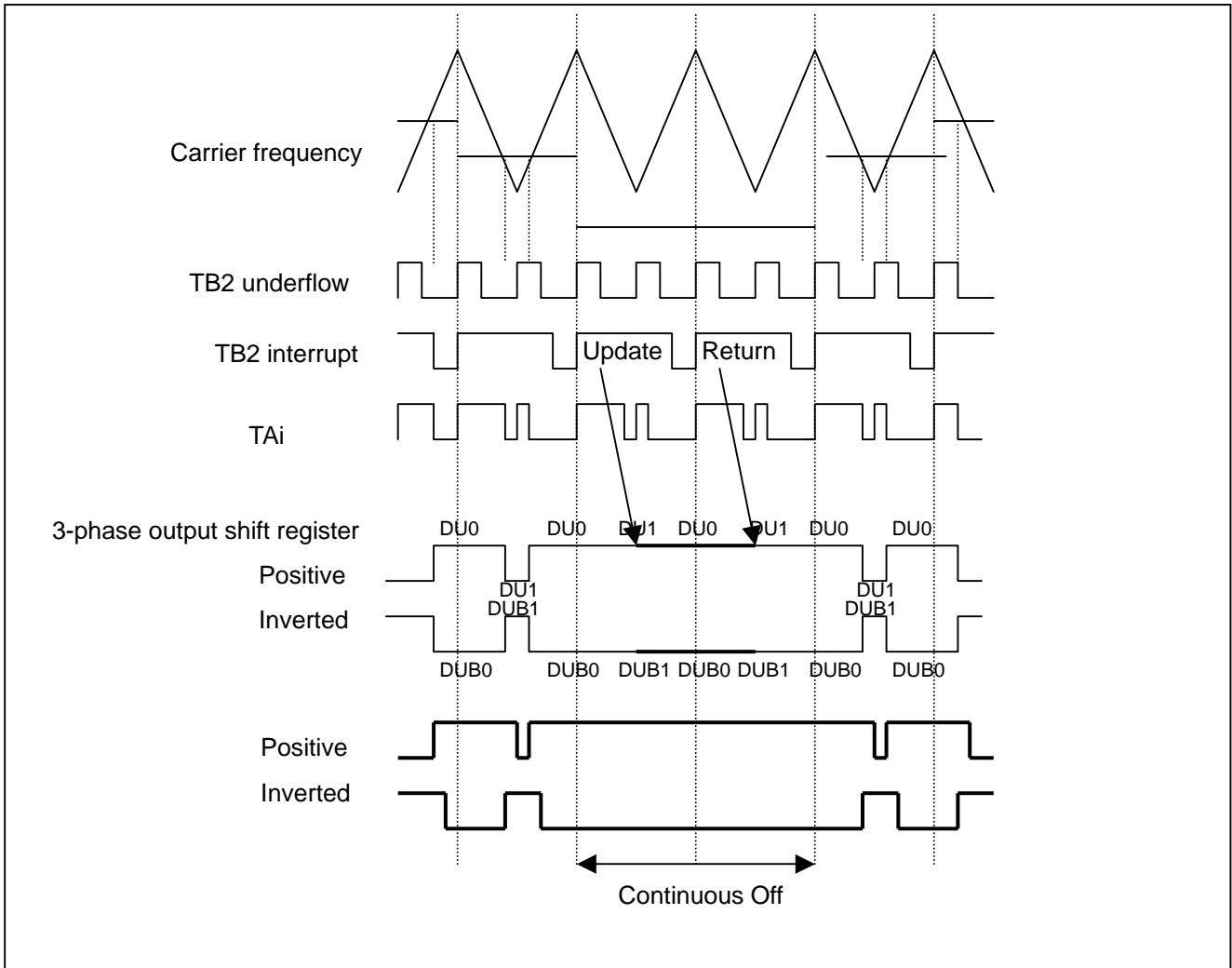


Figure 5.6.2 Timing of writing the 3-phase output buffer register

*3-phase output buffer register contents are transferred to the shift register at the first TB2 underflow after updating.

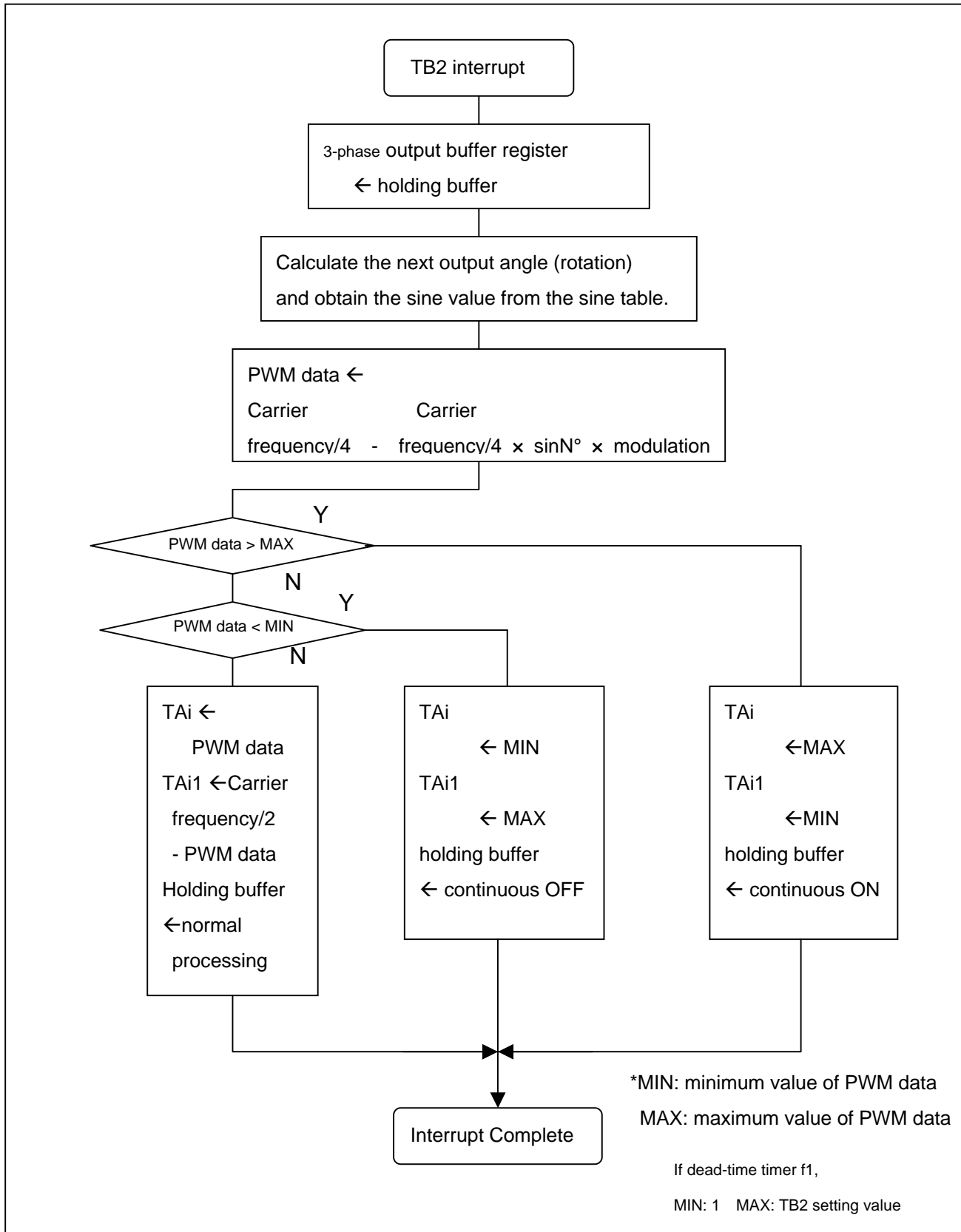


Figure 5.7 TB2 interrupt process flowchart (Method 1)

5.5.2 Continuous On/Off Waveform Output Realized by Timer Set Value

Continuous on/off waveform output can be generated by setting the TAI, TAI1 register to "0" and making the shift register contents go to the "no shift condition" as shown in Figure 5.8.

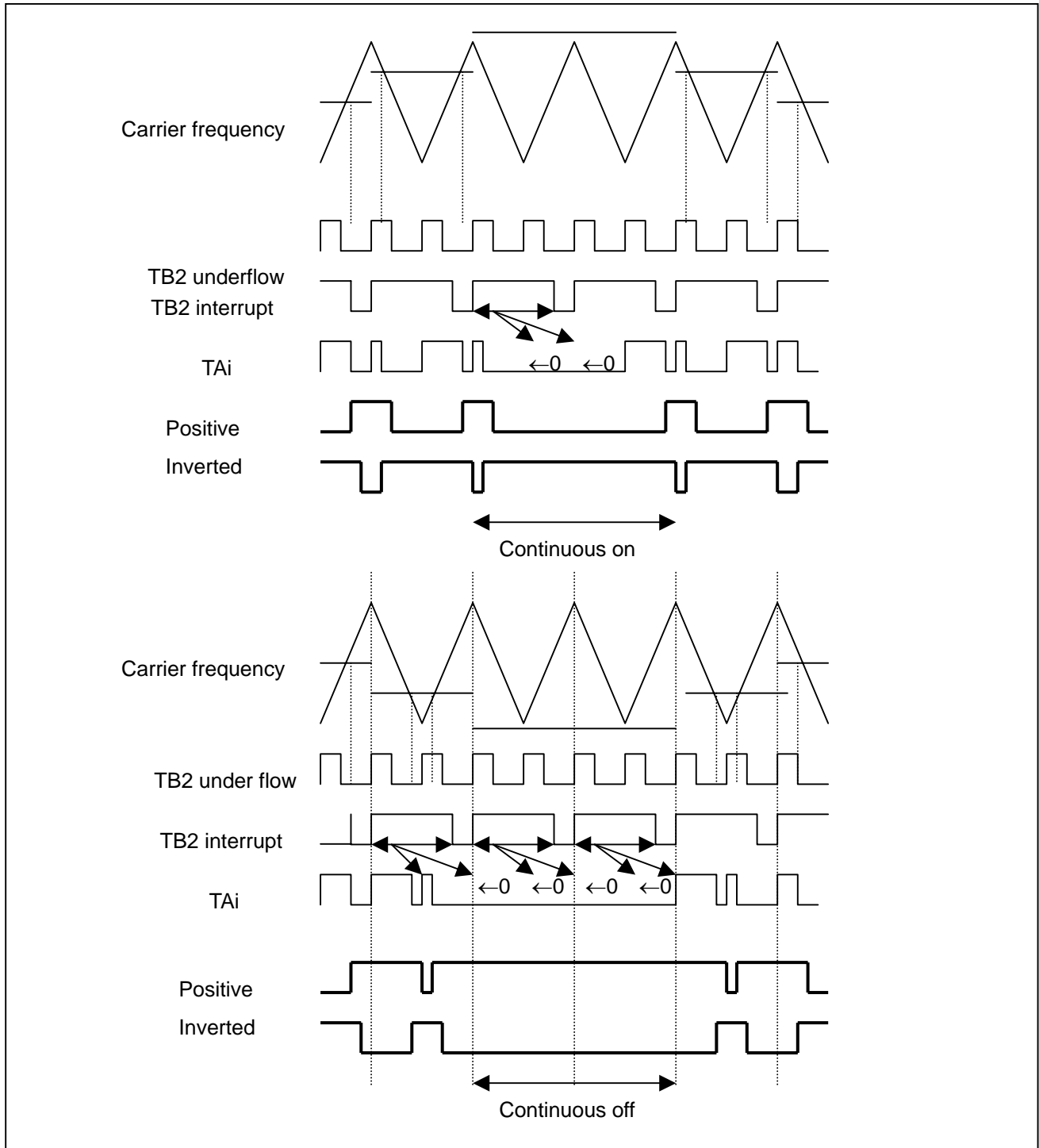


Figure 5.8 Timing and flow of setting register TAI

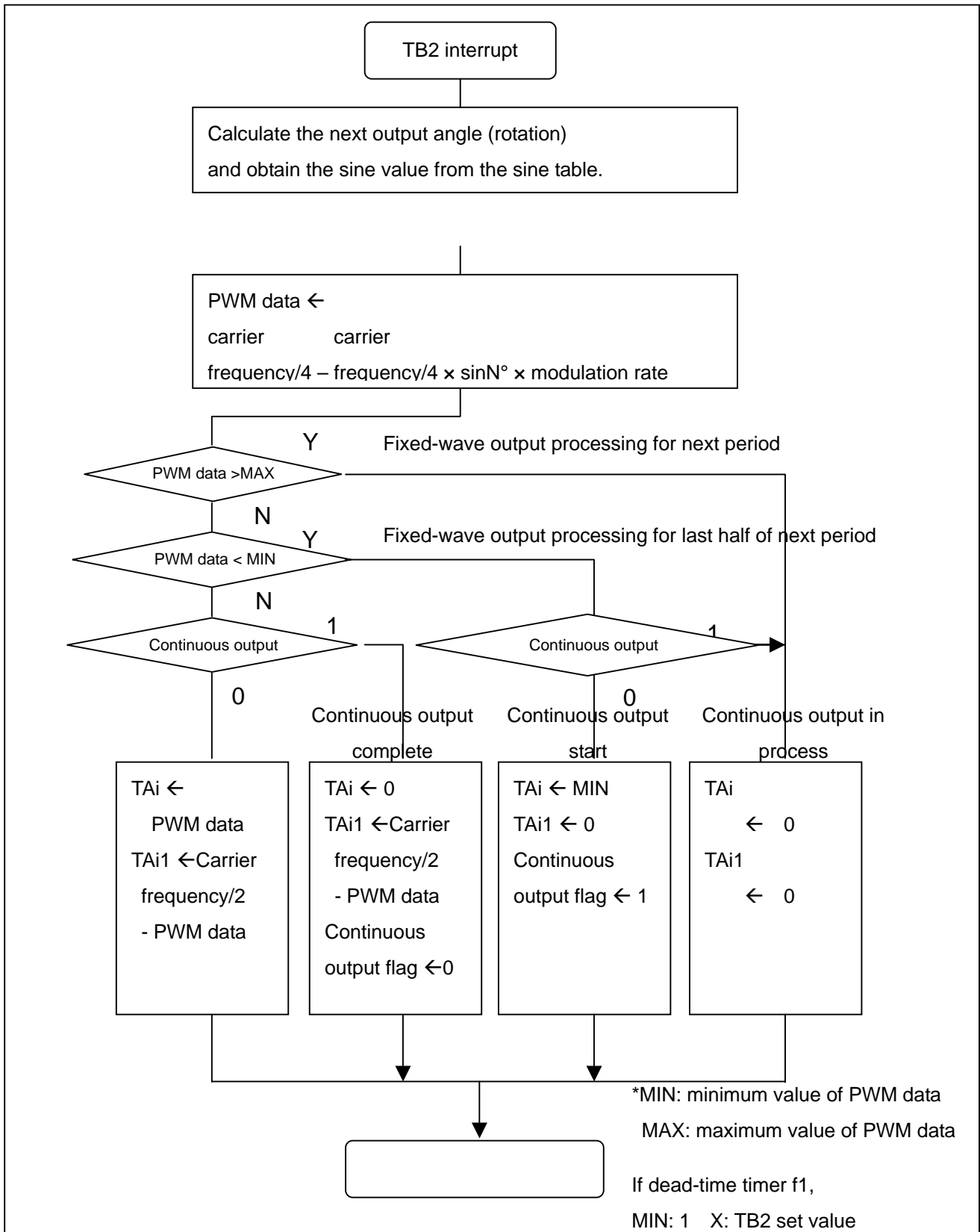


Figure 5.9 TB2 interrupt process flowchart (Method 2.)

5.6 Output Frequency Variation

The output frequency variation can be done by manipulating the sinusoidal wave output angle, which, for example, can be obtained by skip-reading the sine table. The following explains the manipulation method of the output angle for an induction motor.

For example, there are 360 units of data available. The sampling speed of the carrier frequency is 3.6kHz and each data is plotted as one-degree angle. If the output frequency is 10Hz, the data table is sampled every carrier period. But, if the output frequency is changed to 5Hz, the sinusoidal wave period is doubled, therefore, each data is sampled twice.

<u>Standard</u>		
angle	sinN° at 10Hz	sinN° at 5Hz
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

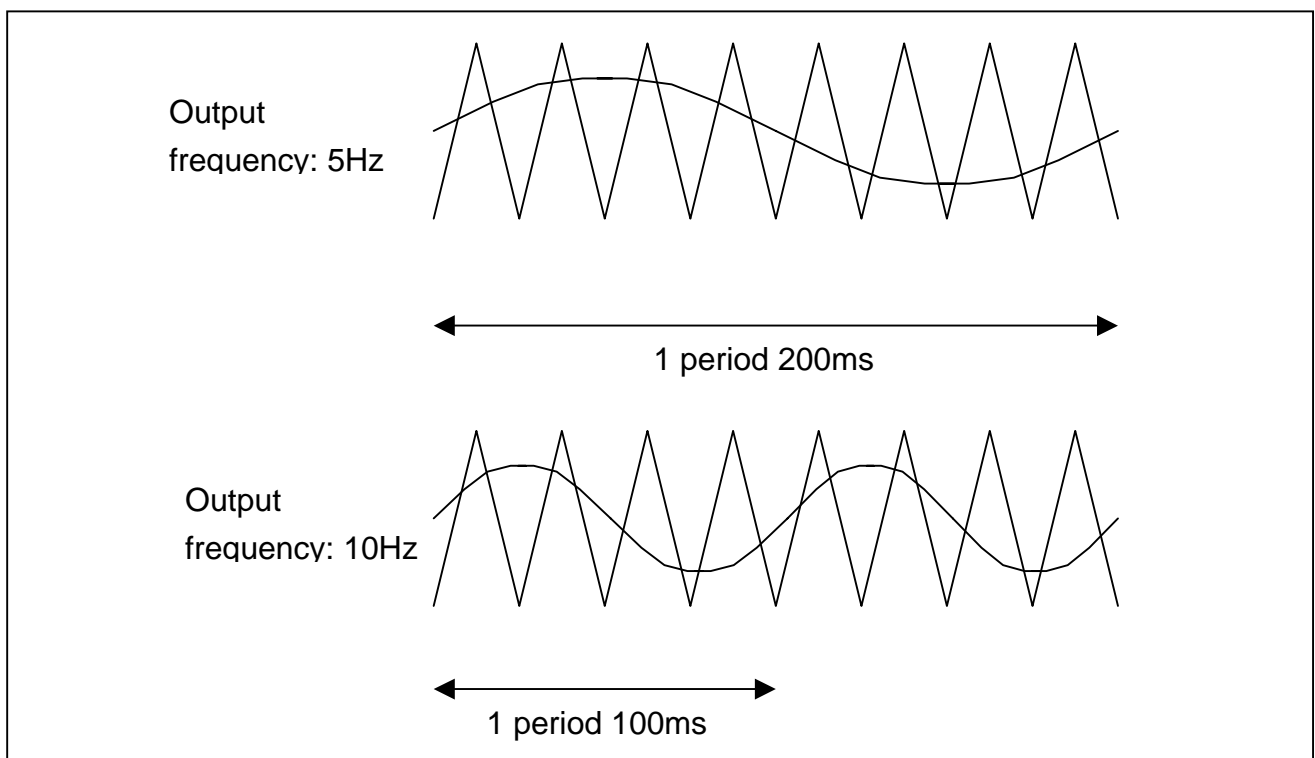


Figure 5.10 Output frequency variation

6.0 How to realize trapezoidal wave control

6.1 How to output trapezoidal waveform

This chapter explains an example of trapezoidal waveform output using 3-phase motor control timer function.

6.2 How to output trapezoidal waveform by using three-phase motor control timer function.

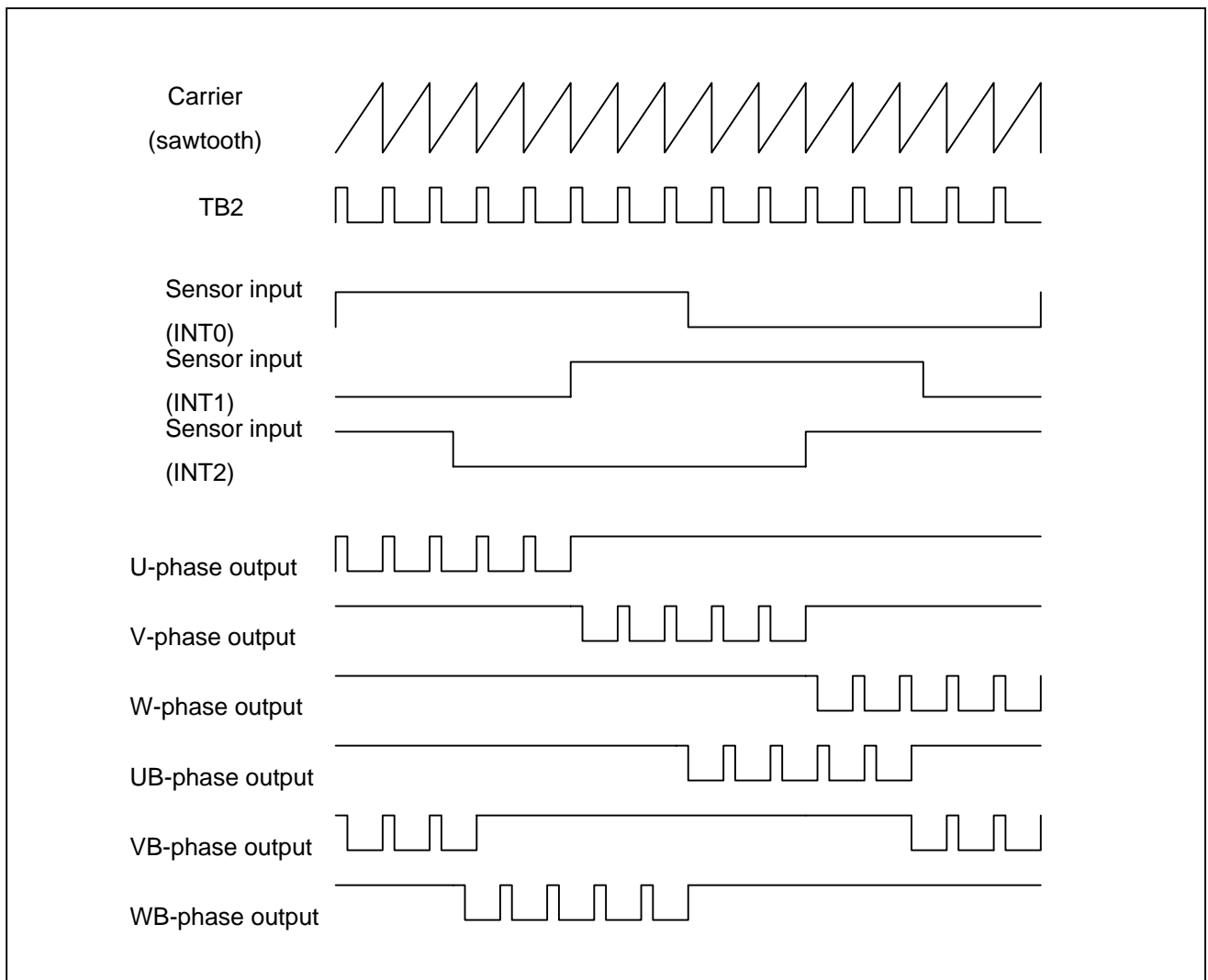


Figure 6.1 Relationship of sensor input of trapezoidal wave control and wave output

Select the sawtooth wave as the modulation mode, and use 3-phase mode 0. Switch each phase to active during the INT interrupt with the 3-phase output buffer register.

```
(invc0=*1**1100; invc1=0000010*; idb0=00*****; idb1=00*****; )
```

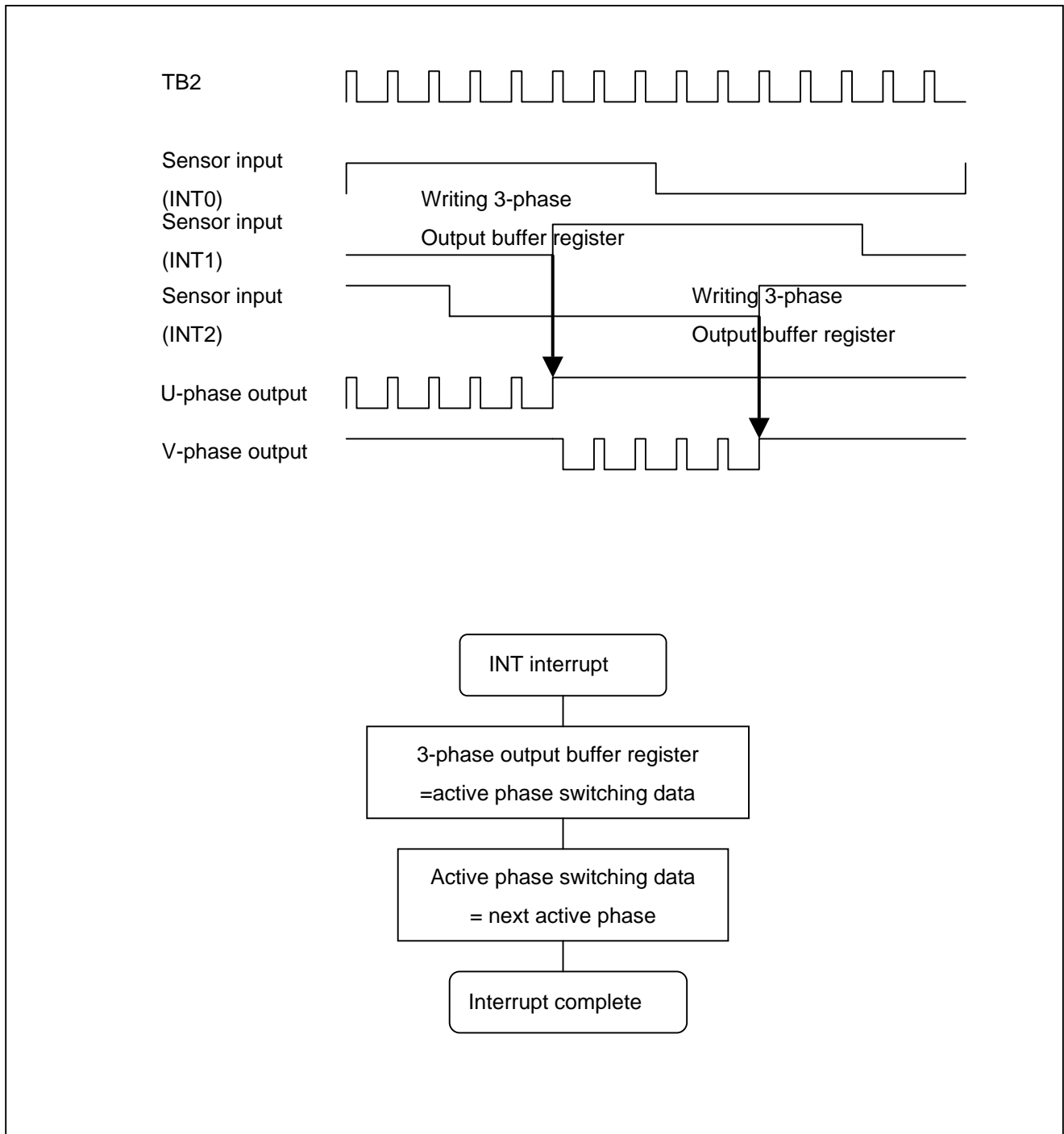


Figure 6.2 Sensor interrupt and output phase

6.3 High speed control

Trapezoidal wave control is based on the torque-to-speed ratio. Specifically, the user can control the speed by updating the values of TA4, TA1, and TA2 each time the torque command value changes.

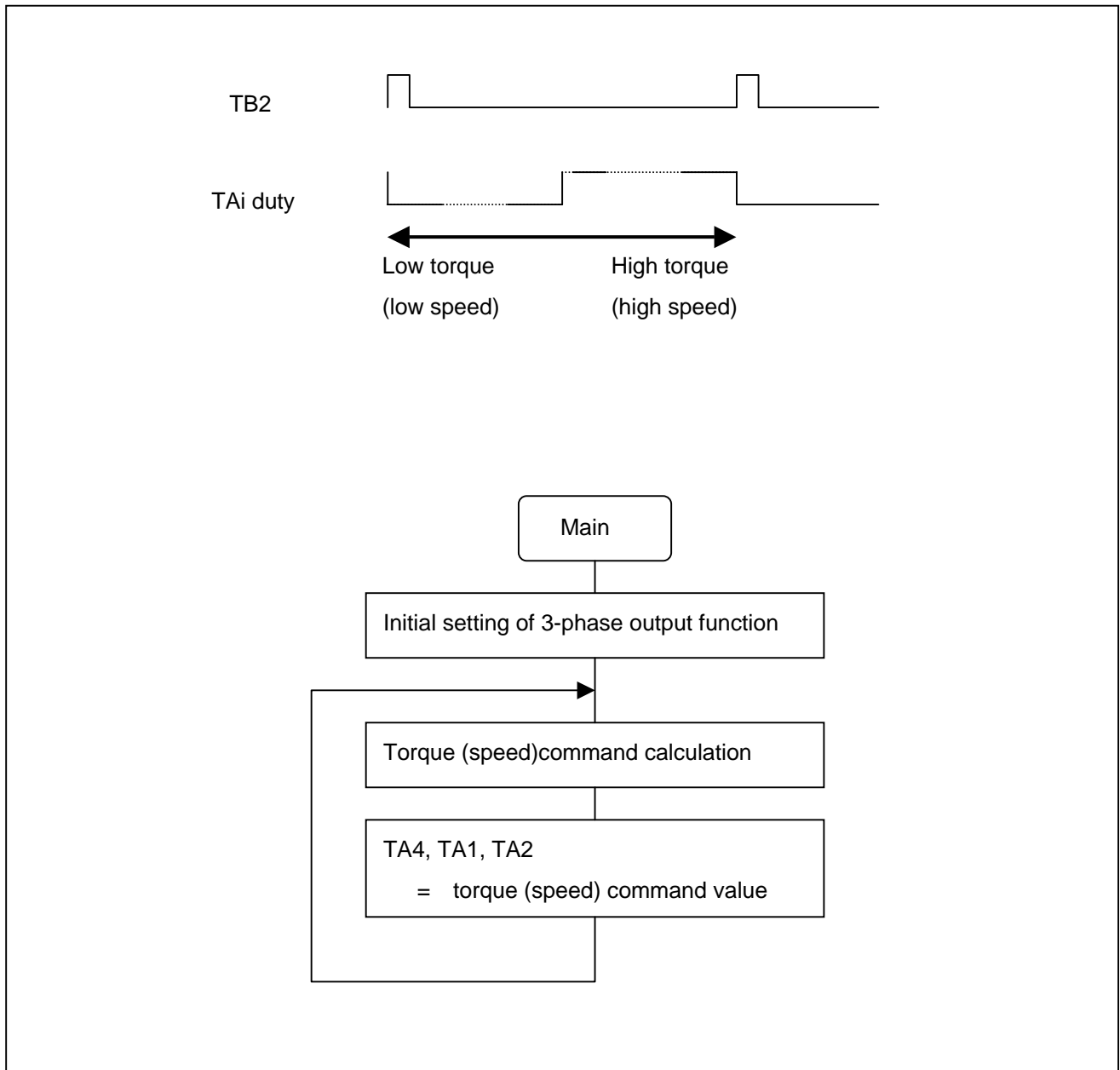


Figure 6.3 Relationship of torque command value and TAI

7.0 Sample Program

The following is a sample program to be used as a reference for realizing 3-phase motor drive waveform. This sample program may require modification and adjustment conforming to each user application.

7.1 Sample Program For Sinusoidal Waveform Output

The following is a sample program for realizing 3-phase sinusoidal waveform output.

```

/*****
*   Sample program for sinusoidal waveform output
*
*****/

/*****/
/*                                     */
/*   SFR Setting                       */
/*                                     */
/*****/

volatile char invc0;
#pragma ADDRESS invc00348h           /* 3-phase PWM control register 0 */
volatile char invc1;
#pragma ADDRESS invc10349h           /* 3-phase PWM control register 1 */
volatile char icTB2;
#pragma ADDRESS icTB2    034dh        /* Interrupt cycle set counter */
volatile char idb0;
#pragma ADDRESS idb0    034ah         /* 3-phase output buffer register 0 */
volatile char idb1;
#pragma ADDRESS idb1    034bh         /* 3-phase output buffer register1 */
volatile char ta1mr;
#pragma ADDRESS ta1mr    0397h        /* Timer A1 mode register */
volatile char ta2mr;
#pragma ADDRESS ta2mr    0398h        /* Timer A2 mode register */
volatile char ta4mr;
#pragma ADDRESS ta4mr    039ah        /* Timer A mode register */
volatile char TB2mr;
#pragma ADDRESS TB2mr    039dh        /* Timer B2 mode register */
volatile char TB2sc;
#pragma ADDRESS TB2sc    039eh        /* Timer B2 specific mode register*/
volatile char trgsr;
#pragma ADDRESS trgsr    0383h        /* Trigger select register */
volatile short TB2;
#pragma ADDRESS TB2    0394h          /* Timer B 2 */
volatile char dtt;
#pragma ADDRESS dtt    034ch          /* Dead-time timer */
volatile short ta4;
#pragma ADDRESS ta4    038eh          /* Timer A 4 */
volatile short ta1;

```

```

#pragma ADDRESS ta1 0388h          /* Timer A 1 */
volatile short ta2;
#pragma ADDRESS ta2 038ah          /* Timer A2 */
volatile short ta41;
#pragma ADDRESS ta41 0346h         /* Timer A4-1 */
volatile short ta11;
#pragma ADDRESS ta11 0342h         /* Timer A1-1 */
volatile short ta21;
#pragma ADDRESS ta21 0344h         /* Timer A2-1 */
volatile char TB2ic;
#pragma ADDRESS TB2ic 005ch        /* Timer B2 interrupt control register */
volatile char tabsr;
#pragma ADDRESS tabsr 0380h        /* Count start flag */
volatile char prcr;
#pragma ADDRESS prcr 000ah         /* Protect register */

/*****/
/* */
/* Initialize */
/* */
/*****/
void main_ini(void);

#define CLK 20000000 /*MCU Frequency Hz*/
#define CARR 20000 /*Carrier Frequency Hz*/
#define DTT_TM 40 /*Dead-time x 0.1 us */

#define carr_set_2 ((CLK/CARR)/2) /*Carrier period 1/2 */
#define carr_set_4 ((CLK/CARR)/4) /*Carrier period 1/4 */
#define dtt_set ((CLK*DTT_TM)/10000000) /* Dead-time timer value */

void main_ini()
{
    icTB2=1; /* One TB2 interrupt at every other TB2 underflow*/

    prcr=0x02; /* Protect cancel*/
    invc0=0x16; /* Select 3-phase motor control timer function output in halting state */
    invc1=0x42; /* 3-phase mode 1 */
    TB2sc=0x01; /* Synchronize reload timing every full-period of the carrier*/
    prcr=0x00; /* Protect*/

    idb0=0x15; /*Set idb0 */
    idb1=0x2a; /*Set idb1 */
}

```

```

ta1mr=0x12;      /*One-shot pulse mode */
ta2mr=0x12;      /*One-shot pulse mode */
ta4mr=0x12;      /*One-shot pulse mode */
TB2mr=0x00;      /*Timer mode */
trgsr=0x45;      /*Triger select register TB2 trigger*/

TB2=carr_set_2-1;      /*One-half carrier period */
dtt=dtc_set;      /*Dead-time timer value */
ta4=ta1=ta2=carr_set_4;      /*Duty 50%*/
ta41=ta11=ta21=carr_set_4;      /*Duty 50%*/

TB2ic=0x07;      /* Interrupt level 7 */

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

tabsr=0x96;      /*Count start flag, timer start*/

prcr=0x02;
invc0=0x1e;      /* Enable output*/
prcr=0x00;      /*Protect*/

}

```



```

/*****/
/*                                          */
/* Sine table                               */
/*                                          */
/* To improve the precision of your calculations, prepare a sine table by a factor of FFFF/2 */
/*                                          */
/*****/

```

```

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,
2062120174, 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

```

};

```

/*****/
/*                                          */
/*  Calculations performed in main routine (ex. Induction motor)  */
/*    1.Skip-read value of sine table                               */
/*    2.Torque command value for output frrequency                */
/*                                          */
/*****/
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 command value x carrier/4*/
signed short sin_cut;       /*Skip-read value for sine pinter */

void main_pro()
{

/*Skip-read value=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 comand 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;  /*idb0 operation-use*/
unsigned char idb1_b=0x2a;  /*idb1 operation-use*/
signed short tq_dat;      /*Torque command value x carrier/4*/
signed short sinpt_sum;   /*Counter for sine pointer sum*/
signed short sin_pt;      /*Sine table pointer*/
signed short PWM_u_w;     /*U-phase PWM command value*/
signed short PWM_v_w;     /*V-phase PWM command value*/
signed short PWM_w_w;     /*W-phase PWM command value*/
#define PWM_max (carr_set_2-1) /*Max. PWM setting*/
#define PWM_min 1           /*Min. PWM setting*/

/*   TB2 interrupt (no 2-phase modulation)   */

#pragma INTERRUPT/B TB2_int
void TB2_int(void)
{
    angle();          /*Calculate sine table pointer */
    i_con();          /*Calculate PWM command value */
    PWM_uvwa_set();  /*PWM command value upper limit revision, set timer*/
}

```

```

/*      TB2 interrupt (with 2-phase modulation by updating timer value)      */

#pragma INTERRUPT/B TB2_int
void TB2_int(void)
{
    angle();          /*Calculate sine table pointer */
    i_con();          /*Calculate PWM command value */

    PWM_uvwa_set();  /*PWM command value upper limit revision, set timer*/
}

/*      TB2 interrupt (with 2-phase modulation by updating 3-phase output buffer)      */

#pragma INTERRUPT/B TB2_int
void TB2_int(void)
{
    PWM_buf();        /*Set 3-phase output buffer*/
    angle();          /*Calculate sine table pointer */
    i_con();          /*Calculate PWM command value */

    PWM_uvwb_set();  /*PWM command value upper limit revision, set timer*/
}

```

```

/*****/
/*
/*  Module for TB2 interrupt (in case of induction motor)
/*  Calculation of sine angle
/*  Calculation of PWM Duty
/*
/*****/

void angle()
{
    sinpt_sum=sin_cut+sinpt_sum;    /*sine pointer sum←skip-read value + sine pointer sum*/
    if(sinpt_sum>23040)             /*Sine pointer sum max. value? 23040=360° x 64*/
    {
        sinpt_sum=sinpt_sum-23040; /*Sine pointer sum max. value revision*/
    }
    else
    {
    }
    sin_pt=sinpt_sum>>6;           /*Sine pointer←Sine pointer sum / 64*/
}

void i_con()
{
/*  U-phase PWM command value = carrier/4-(sinN° x (toruqe command 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 command value = carrier/4-(sin(N+120) ° x (toruqe command 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 command value = carrier/4-(sin(N+240) ° x (toruqe command 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               */
/*                                     */
/*****/

/*          no 2-phase modulation      */

void PWM_uvw_set()
{

/*U-phase PWM revision*/
  if(PWM_max<PWM_u_w)                 /*Duty at MAX?*/
  {
    ta41=PWM_max;                     /*First half←MAX value */
    ta41=PWM_max;                     /*First half←MAX value */
    ta4=PWM_min;                      /*Last half←MIN value  */
  }
  else
  {
    if(PWM_min>PWM_u_w)               /*Duty at MIN?*/
    {
      ta41=PWM_min;                   /*First half←MIN value */
      ta41=PWM_min;                   /*First half←MIN value */
      ta4=PWM_max;                    /*Last half←MAX value  */
    }
    else                               /*MIN<Duty<MAX*/
    {
      ta41=PWM_u_w;                   /*First half←PWM command value */
      ta41=PWM_u_w;                   /*First half←PWM command value */
      ta4=carr_set_2-PWM_u_w;         /*Last half←carrier period/2 - U-phase PWM
                                       command value*/
    }
  }
}

/*V-phase PWM revision*/
  if(PWM_max<PWM_v_w)                 /*Duty at MAX?*/
  {
    ta11=PWM_max;                     /*First half←MAX value */
    ta11=PWM_max;                     /*First half←MAX value */
    ta1=PWM_min;                      /*Last half←MIN value  */
  }
  else
  {
    if(PWM_min>PWM_v_w)               /*Duty at MIN?*/
    {

```

```

        ta11=PWM_min;          /*First half←MIN value */
        ta11=PWM_min;          /*First half←MIN value */
        ta1=PWM_max;           /*Last half←MAX value*/
    }
else
    /*MIN<Duty<MAX*/

    {
        ta11=PWM_v_w;          /*First half←PWM command value */
        ta11=PWM_v_w;          /*First half←PWM command value */
        ta1=carr_set_2-PWM_v_w; /*Last half←carrier period/2- V-phase PWM
                                command value*/
    }
}

/*W-phase PWM revision*/
if(PWM_max<PWM_w_w)          /*Duty at MAX?*/
{
    ta21=PWM_max;             /*First half←MAX value */
    ta21=PWM_max;             /*First half←MAX value*/
    ta2=PWM_min;              /*Last half←MIN value */
}
else
{
    if(PWM_min>PWM_w_w)       /*Duty at MIN?*/
    {
        ta21=PWM_min;         /*First half←MIN value */
        ta21=PWM_min;         /*First half←MIN value */
        ta2=PWM_max;          /*Last half←MAX value */
    }
    else
    /*MIN<Duty<MAX */
    {
        ta21=PWM_w_w;          /*First half←PWM command value */
        ta21=PWM_w_w;          /*First half←PWM command value */
        ta2=carr_set_2-PWM_w_w; /*Last half←carrier period/2- W-phasePWM
                                command value*/
    }
}
}
}

```



```

/* with 2-phase modulation by setting timer value to "0" */

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

void PWM_uvwa_set()
{
/*U-phase PWM revision*/
    if(PWM_max<PWM_u_w)                /*Duty at MAX? */
    {
        ta41=0;                        /*Continuous output start/ in process*/
        ta41=0;                        /*Continuous output start/ in process*/
        ta4=0;                          /* ←0 */
    }
    else
    {
        if(PWM_min>PWM_u_w)            /*Duty at MIN?*/
        {
            if(arm_u_flg==1)          /*Continuous output in process */
            {
                ta41=0;                /*First half←0 */
                ta41=0;                /*First half←0 */
                ta4=0;                  /*Last half←0 */
            }
            else                       /*Continuous output start */
            {
                ta41=PWM_min;          /*First half←MIN value*/
                ta41=PWM_min;          /*First half←MIN value*/
                ta4=0;                  /*Last half←0 */
                arm_u_flg=1;           /*Continuous output flag set */
            }
        }
        else                           /*MIN<Duty<MAX*/
        {
            if(arm_u_flg==1)          /*Continuous output complete*/
            {
                ta41=0;                /*First half←0 */
                ta41=0;                /*First half←0 */
                ta4=carr_set_2-PWM_u_w; /*Last half←carrier period/2- U-phase PWM
                                         command value*/
            }
        }
    }
}

```

```

        arm_u_flg=0;                /*Continuous output flag clear */
    }
    else                            /*Normal processing*/
    {
        ta41=PWM_u_w;              /*First half←PWM command value */
        ta41=PWM_u_w;              /*First half←PWM command value */
        ta4=carr_set_2-PWM_u_w;    /*Last half←carrier period/2 – U-phase PWM
                                   command value*/
    }
}
}

/*V-phase PWM revision*/
if(PWM_max<PWM_v_w)                /*Duty at MAX?*/
{
    ta11=0;                        /*Continuous output start/ in process*/
    ta11=0;                        /*Continuous output start/ in process*/
    ta1=0;                          /* ←0 */
}
else
{
    if(PWM_min>PWM_v_w)            /*Duty at MIN? */
    {
        if(arm_v_flg==1)          /*Continuous output in process */
        {
            ta11=0;                /*First half←0 */
            ta11=0;                /*First half←0 */
            ta1=0;                 /*Last half←0 */
        }
        else                      /*Continuou-arm start*/
        {
            ta11=PWM_min;          /*First half←MIN value */
            ta11=PWM_min;          /*First half←MIN value */
            ta1=0;                 /*Last half←0 */
            arm_v_flg=1;           /*Continuous output flag set */
        }
    }
}
}

```

```

else                                     /*MIN<Duty<MAX*/
{
    if(arm_v_flg==1)                     /*Continuous output complete*/
    {
        ta11=0;                          /*First half←0      */
        ta11=0;                          /*First half←0      */
        ta1=carr_set_2-PWM_v_w;          /*Last half←carrier period/2 - V-phase PWM
                                         command value*/
        arm_v_flg=0;                      /*Continuous output flag clear */
    }
    else                                   /*Normal processing*/
    {
        ta11=PWM_v_w;                    /*First half←PWM command value */
        ta11=PWM_v_w;                    /*First half←PWM command value */
        ta1=carr_set_2-PWM_v_w;          /*Last half←carrier period/2 - V-phase PWM
                                         command value*/
    }
}
}

/*W-phase PWM revision*/
if(PWM_max<PWM_w_w)                      /*Duty at MAX?*/
{
    ta21=0;                              /*Continuous output start / in process*/
    ta21=0;                              /*Continuous output start / in process*/
    ta2=0;                                /* ←0      */
}
else
{
    if(PWM_min>PWM_w_w)                  /*Duty at MIN? */
    {
        if(arm_w_flg==1)                /*Continuous output in process */
        {
            ta21=0;                      /*First half←0      */
            ta21=0;                      /*First half←0      */
            ta2=0;                        /*Last half←0      */
        }
        else                              /*Continuous output start */
        {
            ta21=PWM_min;                 /*First half←MIN value */
            ta21=PWM_min;                 /*First half←MIN value*/
            ta2=0;                         /*Last half←0      */
            arm_w_flg=1;                  /*Continuous arm flag set */
        }
    }
}
else                                     /*MIN<Duty<MAX*/

```

```

    {
        if(arm_w_flg==1)          /*Continuous-arm complete*/
        {
            ta21=0;              /*First half←0 */
            ta21=0;              /*First half←0 */
            ta2=carr_set_2-PWM_w_w; /*Last half←carrier period/2 – W-phase PWM
                                   command value*/
            arm_w_flg=0;         /*Continuous-arm flag clear */
        }
        else                      /*Normal processing */
        {
            ta21=PWM_w_w;        /*First half←PWM command value */
            ta21=PWM_w_w;        /*First half←PWM command value */
            ta2=carr_set_2-PWM_w_w; /*Last half←carrier period/2 – W-phase PWM
                                   command value*/
        }
    }
}

```

/*2-phase modulation realized by setting 3-phase output buffer register */

```

void PWM_buf()
{
    idb0=idb0_b;      /*Set 3-phase output buffer register to "0" */
    idb1=idb1_b;      /*Set 3-phase output buffer register to "1" */
}

```

```

void PWM_uvwb_set()
{
    /*U-phase PWM revision*/
    if(PWM_max<PWM_u_w)          /*Duty at MAX ? */
    {
        idb0_b=idb0_b&0xfc;
        idb0_b=idb0_b|0x02;

        ta41=PWM_max;           /*First half←MAX value */
        ta41=PWM_max;           /*First half←MAX value */
        ta4=PWM_min;            /*Last half←MIN value */
    }
    else

```

```

{
    if(PWM_min>PWM_u_w)          /*Duty at MIN? */
    {
        idb1_b=idb1_b&0xfc;
        idb1_b=idb1_b|0x01;
        ta41=PWM_min;           /*First half←MIN value */
        ta41=PWM_min;           /*First half←MINE value */
        ta4=PWM_max;            /*Last half←MAX value */
    }
    else                          /*MIN<Duty<MAX */
    {
        idb0_b=idb0_b&0xfc;
        idb0_b=idb0_b|0x01;
        idb1_b=idb1_b&0xfc;
        idb1_b=idb1_b|0x02;

        ta41=PWM_u_w;           /*First half←PWM command value */
        ta41=PWM_u_w;           /*First half←PWM command value */
        ta4=carr_set_2-PWM_u_w; /*Last half←carrier period/1 – U-phase PWM
                                command value*/
    }
}

/*V-phase PWM revision*/
if(PWM_max<PWM_v_w)            /*Duty at MAX ?*/
{
    idb0_b=idb0_b&0xf3;
    idb0_b=idb0_b|0x08;

    ta11=PWM_max;              /*First half←MAX value */
    ta11=PWM_max;              /*First half←MAX value */
    ta1=PWM_min;               /*Last half←MIN value */
}
else
{
    if(PWM_min>PWM_v_w)        /*Duty at MIN? */
    {
        idb1_b=idb1_b&0xf3;
        idb1_b=idb1_b|0x04;

        ta11=PWM_min;          /*First half←MIN value */
        ta11=PWM_min;          /*First half←MIN value */
        ta1=PWM_max;           /*Last half←MAX value */
    }
    else                          /*MIN<Duty<MAX*/
    {

```

```

        idb0_b=idb0_b&0xf3;
        idb0_b=idb0_b|0x04;
        idb1_b=idb1_b&0xf3;
        idb1_b=idb1_b|0x08;

        ta11=PWM_v_w;          /*First half←PWM command value */
        ta11=PWM_v_w;          /*First half←PWM command value */
        ta1=carr_set_2-PWM_v_w; /*Last half←carrier period/2 - V-phase PWM
                                command value*/
    }
}

/*W-phase PWM revision */
if(PWM_max<PWM_w_w)          /*Duty at MAX?*/
{
    idb0_b=idb0_b&0xcf;
    idb0_b=idb0_b|0x20;

    ta21=PWM_max;            /*First half←MAX value */
    ta21=PWM_max;            /*First half←MAX value*/
    ta2=PWM_min;             /*Last half←MIN value*/
}
else
{
    if(PWM_min>PWM_w_w)      /*Duty at MIN? */
    {
        idb1_b=idb1_b&0xcf;
        idb1_b=idb1_b|0x10;
        ta21=PWM_min;        /*First half←MIN value */
        ta21=PWM_min;        /*First half←MIN value*/
        ta2=PWM_max;        /*Last half←MAX value*/
    }
    else                      /*MIN<Duty<MAX */
    {
        idb0_b=idb0_b&0xcf;
        idb0_b=idb0_b|0x10;
        idb1_b=idb1_b&0xcf;
        idb1_b=idb1_b|0x20;

        ta21=PWM_w_w;        /*First half←PWM command value */
        ta21=PWM_w_w;        /*First half←PWM command value */
        ta2=carr_set_2-PWM_w_w; /*Last half←carrier period/2 - W-phase PWM
                                command value*/
    }
}
}
}

```

7.2 Trapezoidal waveform sample program

The following is a sample program to be used as a reference for trapezoidal waveform output.

```

/*****
*   Trapezoidal waveform sample programming
*
*****/
/*****/
/*                               */
/*   SFR setting                               */
/*                               */
/*****/
volatile char invc0;
#pragma ADDRESS    invc0        0348h        /* 3-phase PWM control register0 */
volatile char invc1;
#pragma ADDRESS    invc1        0349h        /* 3-phase PWM control register 1 */
volatile char icTB2;
#pragma ADDRESS    icTB2        034dh        /* Interrupt cycles setting counter */
volatile char idb0;
#pragma ADDRESS    idb0         034ah        /* 3-phase output buffer register 0 */
volatile char idb1;
#pragma ADDRESS    idb1         034bh        /* 3-phase output buffer register 1 */
volatile char ta1mr;
#pragma ADDRESS    ta1mr        0397h        /* Timer A1 mode register */
volatile char ta2mr;
#pragma ADDRESS    ta2mr        0398h        /* Timer A2 mode register */
volatile char ta4mr;
#pragma ADDRESS    ta4mr        039ah        /* Timer A4 mode register */
volatile char TB2mr;
#pragma ADDRESS    TB2mr        039dh        /* Timer B2 mode register */
volatile char TB2sc;
#pragma ADDRESS    TB2sc        039eh        /* Timer B2 special mode register*/
volatile char trgsr;
#pragma ADDRESS    trgsr        0383h        /* Trigger select register */
volatile short TB2;
#pragma ADDRESS    TB2          0394h        /* Timer B2 */
volatile char dt;
#pragma ADDRESS    dt           034ch        /* Dead-time timer */
volatile short ta4;
#pragma ADDRESS    ta4          038eh        /* TimerA4 */
volatile short ta1;
#pragma ADDRESS    ta1          0388h        /* TimerA1 */
volatile short ta2;
#pragma ADDRESS    ta2          038ah        /* TimerA2 */
volatile short ta41;
#pragma ADDRESS    ta41         0346h        /* TimerA4-1 */
volatile short ta11;
#pragma ADDRESS    ta11         0342h        /* TimerA1-1 */

```

```
volatile short ta21;
#pragma ADDRESS ta21 0344h /* TimerA2-1 */
volatile char TB2ic;
#pragma ADDRESS TB2ic 005ch /* Timer B2 interrupt control register */
volatile char tabsr;
#pragma ADDRESS tabsr 0380h /* Count start flag */
volatile char prcr;
#pragma ADDRESS prcr 000ah /* Protect register */
volatile char ifsr;
#pragma ADDRESS ifsr 035fh /* External interrupt trigger select register */
volatile char int0ic;
#pragma ADDRESS int0ic 005dh /* INT0 interrupt control register */
volatile char int1ic;
#pragma ADDRESS int1ic 005eh /* INT1 interrupt control register */
volatile char int2ic;
#pragma ADDRESS int2ic 005fh /* INT2 interrupt control register */
volatile char p7;
#pragma ADDRESS p7 03edh /* Port 7 register */
volatile char p8;
#pragma ADDRESS p8 03f0h /* Port 8 register */
volatile char pd7;
#pragma ADDRESS pd7 03efh /* Port 7 direction register */
volatile char pd8;
#pragma ADDRESS pd8 03f2h /* Port 8 direction register */
volatile char tprc;
#pragma ADDRESS tprc 025ah /* 3-phase protet control register */
volatile char pfc;
#pragma ADDRESS pfc 0358h /* Port function control register */
```

```

/*****/
/* */
/* Initialize */
/* */
/*****/

```

```
void main_ini(void);
```

```
#define CLK 20000000 /* MCU frequency Hz */
```

```
#define CARR 2000 /* Carrier frequency Hz */
```

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

```
void main_ini()
```

```
{
```

```
    icTB2=1; /* One TB2 interrupt at every other TB2 underflow*/
```

```
    prcr=0x02; /* Protect*/
```

```
    invc0=0x44; /*Select 3-phase motor control timer functionm, output in halting state*/
```



```

invc1=0x60;      /*3-phase mode 0 dead-time timer invalid*/
prcr=0x00;      /*Protect*/

TB2sc=0x00;      /* Synchronize reload timing with TB2 */

idb0=0x0ff;     /*Set 3-phase output buffer register to "0" */
idb1=0x0ff;     /*Set 3-phase output buffer register to "1" */

ta1mr=0x12;     /*One-shot pulse mode */
ta2mr=0x12;     /*One-shot pulse mode */
ta4mr=0x12;     /*One-shot pulse mode */
TB2mr=0x00;     /*Timer mode */
trgsr=0x45;     /*Trigger select register TB2 trigger */

TB2=carr_set;   /*Carrier period */
dtt=1;         /*Dead-timer timer MIN. value setting*/

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

/*Use INT for position input*/
ifsr=0x07;     /*INT 0,1,2 both edges*/
int0ic=7;      /*INT 0,1,2 interrupt level setting*/
int1ic=7;
int2ic=7;

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

tabsr=0x96;    /*Count start flag, Timer start*/

prcr=0x02;     /* protect cancel*/
invc0=0x4c;    /* Enable output*/
prcr=0x00;     /* Protect */
}

```

```

/*****/
/*                                     */
/*  Calculation performed in main routine                                     */
/*  1.  Setting PWM Duty                                                 */
/*                                     */
/*****/
void dc_PWM_set(void);

unsigned short  tk_dat_w;                /*Torque command value*/

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

/*****/
/*                                     */
/*  Angle detect ( when using INT)interrupt                               */
/*  Switching by 3-phase output buffer register of active phase in 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 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;      /*Change output phase */
    if(edge0_flg==0)
    {
        edge0_flg=1;
        rad_datw=60;     /*Set next angle*/
    }
    else
    {
        edge0_flg=0;
        rad_datw=240;    /*Set next angle*/
    }

    rad_to_PWM();       /*Set next output phase*/
}

```

```
void int1_int(void)
{
    idb0=idb0_bufb;      /*Change output phase*/
    if(edge1_flg==0)
    {
        edge1_flg=1;
        rad_datw=180;    /*Set next angle*/
    }
    else
    {
        edge1_flg=0;
        rad_datw=0;      /*Set next angle */
    }

    rad_to_PWM();       /*Set next output phase*/
}

```

```
void int2_int(void)
{
    idb0=idb0_bufb;      /*Change output phase*/
    if(edge2_flg==0)
    {
        edge2_flg=1;
        rad_datw=300;    /*Set next angle*/
        rad_to_PWM();
    }
    else
    {
        edge2_flg=0;

        rad_datw=120;    /*Set next anlge*/
    }
}

```

```

    }
    rad_to_PWM();          /*Set next output phase*/
}

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

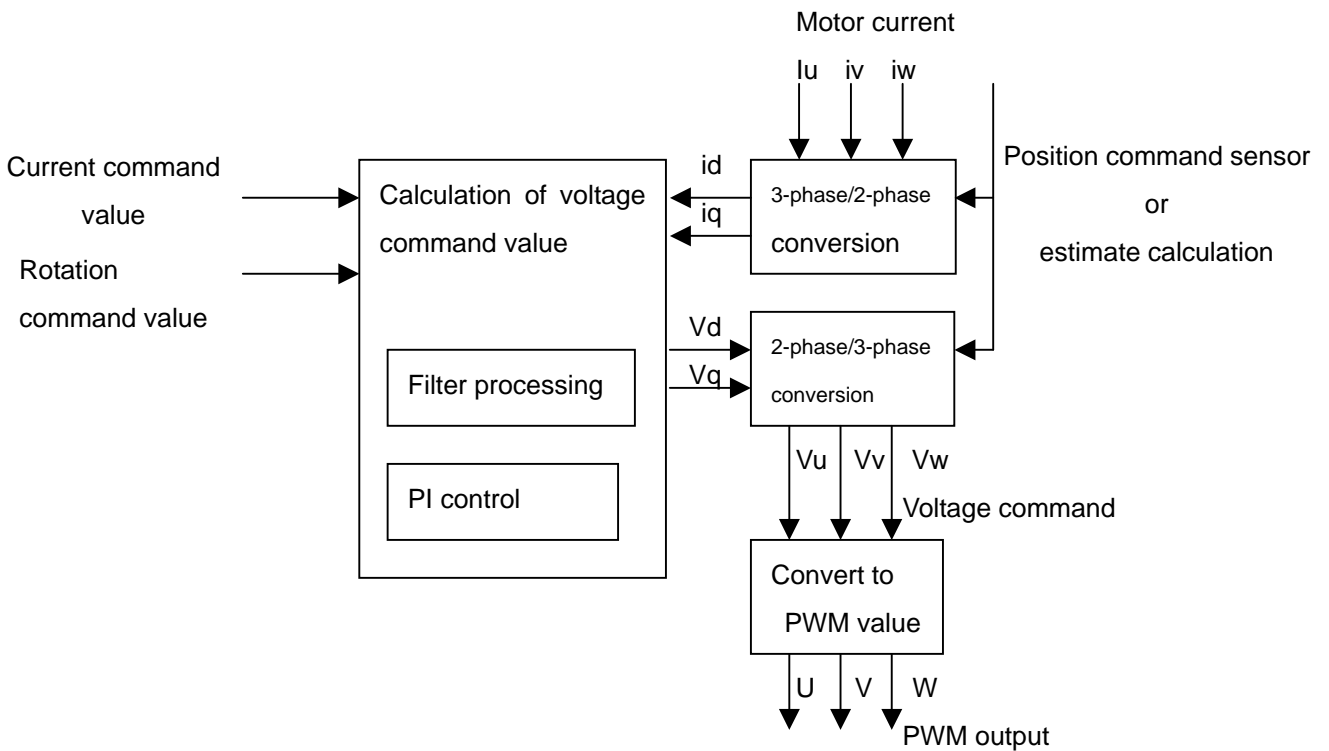
```

7.3 Feedback control sample program

The followings are the control modules used for program development of vector control.

- 3-phase/2-phase conversion
- 2-phase/3-phase conversion
- PI control
- Filter processing

*Please refer to previous sample program for PWM output



7.3.1 Three-phase / two-phase conversion module

Following is the an example of 3-phase/2-phase conversion.

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

To improve precision of calculation, the rest of the sample program is cacculated as follows.

iu: U-phase current value	$\times 2^6$
iv: V-phase current value	$\times 2^6$
iw: W-phase current value	$\times 2^6$
id: d shaft current value	$\times 2^6$
iq: q shaft current value	$\times 2^6$
Sine table	$\times 2^{15}$

```

/*****
*      3-phase/2-phase conversion sample program
*
*****/
void dq_samp(void);

signed short iu_6w;      /* U-phase current[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;     /* sine 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 × U-phase current value[2^6]      */
    /*      +cos(θ+240) × V-phase current value[2^6] */
    /*      +cos(θ+120) × W current value[2^6] ) */
    /*      *sine table is [2^15]                */

    id_6w=(short)((((long)work_6w*26755)>>15);      /* d shaft current [2^6]=work[2^6] × √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 × U-phase current value[2^6]      */
    /*      -sin(θ+240) × V-phase current value[2^6] */
    /*      -sin(θ+120) × W current value[2^6] ) */
    /*      *sine table is [2^15]                */

    iq_6w=(short)((((long)work_6w*26755)>>15);      /* q shaft current [2^6]=work[2^6] × √2/3 */
}

```

7.3.2 Two phase/three-phase conversion module

The following is an example of 2-phase/3-phase conversion.

$$\begin{pmatrix} V_u \\ V_v \\ V_w \end{pmatrix} = \sqrt{\frac{2}{3}} \begin{pmatrix} \cos\theta & -\sin(\theta) \\ \cos(\theta+240) & -\sin(\theta+240) \\ \cos(\theta+120) & -\sin(\theta+120) \end{pmatrix} \begin{pmatrix} V_d \\ V_q \end{pmatrix}$$

To improve precision of calculation, the rest of the sample program is calculated as follows.

Vd: d shaft voltage value	$\times 2^6$
Vq: q shaft voltage value	$\times 2^6$
Vu: U-phase voltage command value	$\times 2^6$
Vv: V-phase voltage command value	$\times 2^6$
Vw: W-phase voltage command value	$\times 2^6$
Sine table	$\times 2^{15}$


```

/*****
*      2-phase/3-phase conversion sample program
*
*****/

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 command [2^6] */
signed short vv_6w;      /* V-phase voltage command [2^6] */
signed short vw_6w;      /* W-phase voltage command [2^6] */
signed short work_6w;     /* work[2^6]                  */
signed short sin_pt;     /* sine 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× d shaft voltage[2^6]      */
            /*      -sinθ× q shaft voltage[2^6]      */
            /*      *Sine table is [2^15]             */

    vu_6w=(short)(((long)work_6w*26752)>>15); /* U-phase voltage command [2^6]=work[2^6] ×√ 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]      */
            /*      *Sine table is [2^15]             */

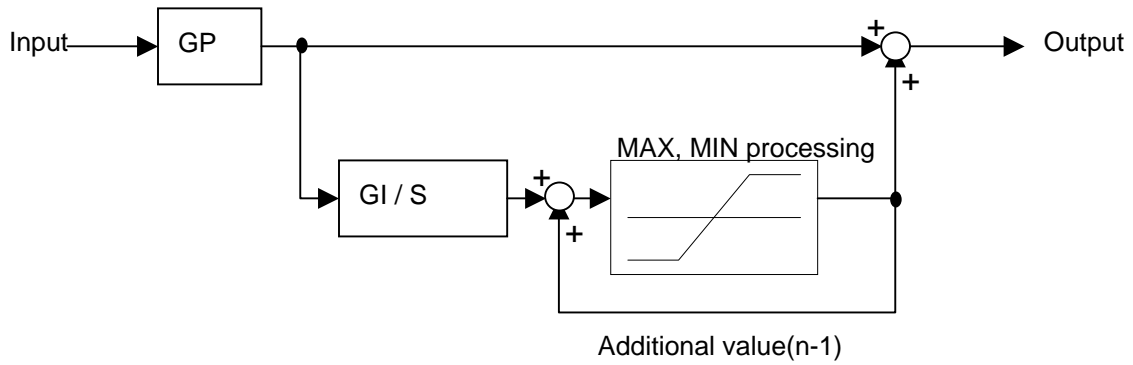
    vv_6w=(short)(((long)work_6w*26752)>>15); /* V-phase voltage command[2^6]=work[2^6] ×√ 2/3 */

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

```

7.3.3 PI Control Module

The following is an example of PI control



To improve precision of calculation, the rest of sample program is calculated as follows.

Input data	$\times 2^6$
Output data	$\times 2^6$
Gain P	$\times 2^{10}$
Gain I / control period	$\times 2^{18}$

```

/*****
*      PI control sample program
*
*****/
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 period t)[2^18] */
signed long sum_16l;            /* Additional 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 is 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] */

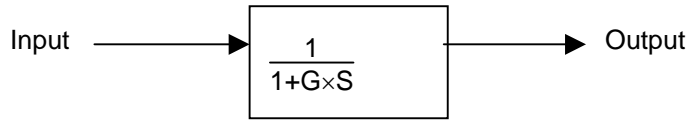
    if(work2_16l > work2_limit)
    {
        work2_16l = work2_limit;
    }else
    {
        if(work2_16l < -work2_limit)
        {
            work2_16l = -work2_limit;
        }
    }

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

```

7.3.4 Filter control module

The following is an example of filter control.



To improve precision of calculation, the rest of sample program is calculated as follows.

Input data	$\times 2^6$
Output data	$\times 2^6$
Gain P	$\times 2^{10}$
Gain I / control period	$\times 2^{18}$

```

/*****
*   Filter control sample program
*
*****/
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 period) [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 period)[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 */
}

```

8.0 Reference Document

Data sheet

Please refer to M16C/28 Group data sheet

9.0 Website URL and Customer Support Center

Renesas Technology Corp. Website
<http://www.renesas.com/>

Inquiries for Renesas products
Customer Support Center: csc@renesas.com

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	2004.09.01	-	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.