To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

RENESAS

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.

Application Note

Single-Phase Induction Motor Control by μPD78F0714

Two-Phase Sine Wave Inverter Drive via V/f Control

μPD78F0714

Document No. U17481EJ1V0AN00 (1st edition)
Date Published September 2005 N CP(K)

[MEMO]

NOTES FOR CMOS DEVICES

① **VOLTAGE APPLICATION WAVEFORM AT INPUT PIN**

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (MAX) and V_{IH} (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (MAX) and V_{IH} (MIN).

② **HANDLING OF UNUSED INPUT PINS**

Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

③ **PRECAUTION AGAINST ESD**

A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

④ **STATUS BEFORE INITIALIZATION**

Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.

⑤ **POWER ON/OFF SEQUENCE**

In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current.

The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

⑥ **INPUT OF SIGNAL DURING POWER OFF STATE**

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements.

Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC Electronics product in your application, please contact the NEC Electronics office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

[GLOBAL SUPPORT]
http://www.necel.com/en/support/support.html

NEC Electronics America, Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65030

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
- **Filiale Italiana**
  Milano, Italy
  Tel: 02-66 75 41
- **Branch The Netherlands**
  Eindhoven, The Netherlands
  Tel: 040-265 40 10
- **Tyskland Filial**
  Taeby, Sweden
  Tel: 08-63 87 200
- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-558-3737

NEC Electronics Shanghai Ltd.
Shanghai, P.R. China
Tel: 021-5888-5400

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377

NEC Electronics Singapore Pte. Ltd.
Novena Square, Singapore
Tel: 6253-8311

J05.6

INTRODUCTION

Target Readers This application note is intended for users who understand the functions of the μPD78F0714 and who design application systems that use this microcontroller. The applicable products are shown below.

- μPD78F0714

Purpose This application note explains inverter control of two-phase sine wave drive via V/f control using PWM output and A/D converter input as a system example of the timer/counter function of the μPD78F0714.

Organization This application note is divided into the following sections.

- Control method
- Hardware configuration
- Software configuration
- Program list

How to Use This Manual It is assumed that the reader of this application note has general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

For details of hardware functions (especially register functions, setting methods, etc.) and electrical characteristics
→ See the **μPD78F0714 User's Manual (U16928E)**.

For details of instruction functions
→ See the **78K/0 Series Instructions User's Manual (12326E)**.

Conventions	Data significance:	Higher digits on the left and lower digits on the right
	Active low representation:	$\overline{\text{xxx}}$ (overscore over pin or signal name)
	Memory map address:	Higher addresses on the top and lower addresses on the bottom
	Note:	Footnote for item marked with **Note** in the text
	Caution:	Information requiring particular attention
	Remark:	Supplementary information
	Numeric representation:	Binary ... xxxx or xxxxB Decimal ... xxxx Hexadecimal ... xxxxH
	Prefix indicating the power of 2 (address space, memory capacity):	K (kilo): 2^{10} = 1,024 M (mega): 2^{20} = $1{,}024^2$ G (giga): 2^{30} = $1{,}024^3$
	Data type:	Word: 32 bits Halfword: 16 bits Byte: 8 bits

Related Documents The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

Documents Related to Devices

Document Name	Document No.
μPD78F0714 User's Manual	U16928E
78K/0 Series Instructions User's Manual	U12326E
Inverter Control by μPD78F0714 120° Excitation Method Control by Zero-Cross Detection Application Note	U17297E
Single-Phase Induction Motor Control by μPD78F0714 Two-Phase Sine Wave Inverter Drive via V/f Control Application Note	This manual

Caution The related documents listed above are subject to change without notice. Be sure to use the latest version of each document when designing.

Documents Related to Development Tools (Software) (User's Manuals)

Document Name		Document No.
RA78K0 Ver. 3.80 Assembler Package	Operation	U17199E
	Language	U17198E
	Structured Assembly Language	U17197E
CC78K0 Ver. 3.70 C Compiler	Operation	U17201E
	Language	U17200E
SM+ System Simulator	Operation	U17246E
	User Open Interface	U17247E
ID78K0-QB Ver. 2.81 Integrated Debugger	Operation	U16996E
PM plus Ver. 5.20		U16934E

Documents Related to Development Tools (Hardware) (User's Manuals)

Document Name	Document No.
QB-78K0KX1H In-Circuit Emulator	U17081E

Documents Related to Flash Memory Programming

Document Name	Document No.
PG-FP3 Flash Memory Programmer User's Manual	U13502E
PG-FP4 Flash Memory Programmer User's Manual	U15260E

Other Documents

Document Name	Document No.
SEMICONDUCTOR SELECTION GUIDE – Products and Packages –	X13769X
Semiconductor Device Mount Manual	**Note**
Quality Grades on NEC Semiconductor Devices	C11531E
NEC Semiconductor Device Reliability/Quality Control System	C10983E
Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD)	C11892E

Note See the "Semiconductor Device Mount Manual" website (http://www.necel.com/pkg/en/mount/index.html).

Caution The related documents listed above are subject to change without notice. Be sure to use the latest version of each document when designing.

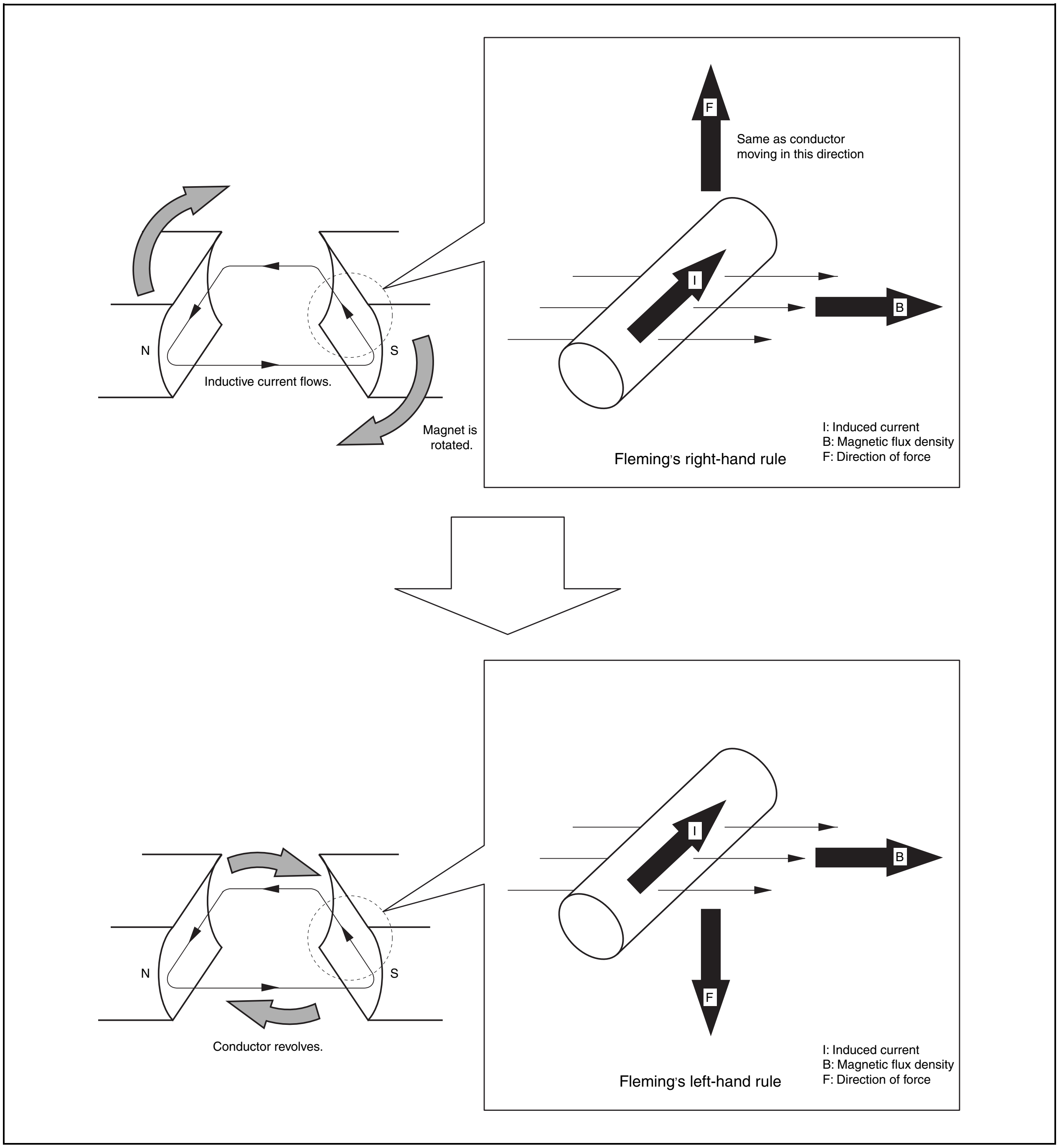
CONTENTS

CHAPTER 1 CONTROL METHOD

1.1 Outline of Induction Motor Control

The basic principle of revolving an induction motor is that, when magnets surrounding a closed conductor are rotated, an inductive current flows through the conductor, which generates a force and the conductor revolves (refer to **Figure 1-1**).

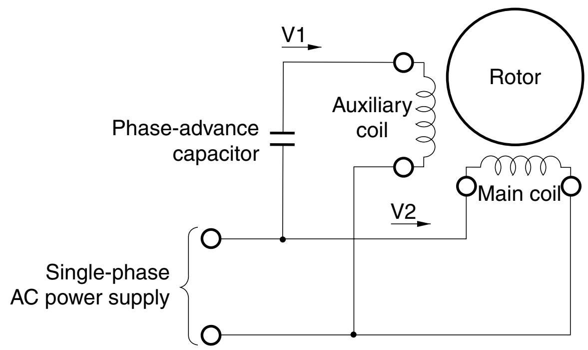
Figure 1-1. Principle of Revolving Induction Motor



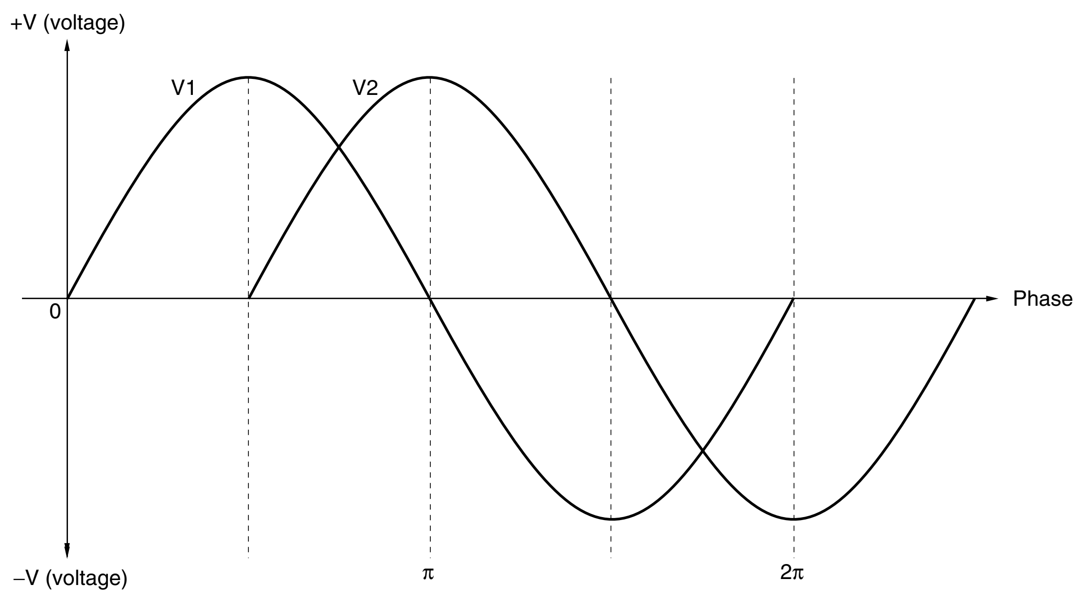
An induction motor that uses a single-phase AC power supply is called a single-phase induction motor. Because rotating magnetic field cannot be created with a single-phase AC power supply, however, the phase is shifted 90° by using a phase-advance capacitor. Therefore, two-phase AC power is actually used (refer to **Figure 1-2**).

Figure 1-2. Single-Phase Induction Motor

(a) Conversion from single phase to two phases with phase-advance capacitor



(b) Two-phase AC waveform



1.2 Outline of Single-Phase Induction Motor Control

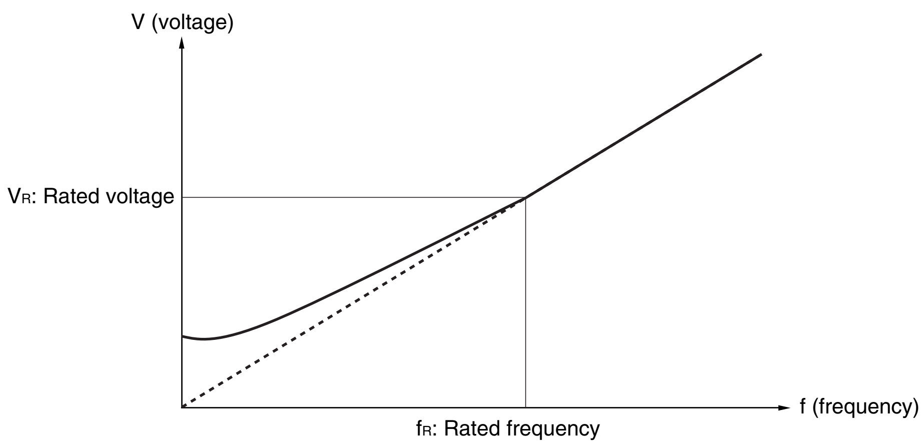
This section explains how to control a single-phase induction motor which uses the μPD78F0714.

As explained in **1.1 Outline of Induction Motor Control**, a single-phase induction motor actually revolves on a two-phase AC power with one phase shifted 90° from the other, the number of revolutions is controlled through inverter control of a two-phase sine wave.

To control the motor, a three-phase PWM inverter is used and a two-phase sine wave is driven through V/f control.

The principle of V/f control is that a rotating magnetic field is created by holding the ratio of the voltage (V) and frequency (f) constant, so that the torque can be roughly held constant. Even if the V/f ratio were to be held constant, however, the rotating magnetic field would not be held constant in a low-frequency region (refer to the straight line (dotted line) in Figure 1-3. Therefore, the rotating magnetic field is held constant by correcting the voltage drop as shown in the curve (solid line) in Figure 1-3. In this application, a method by which the V/f ratio is treated as a straight line by dividing the frequency into several regions is employed.

Figure 1-3. Concept of V/f Ratio



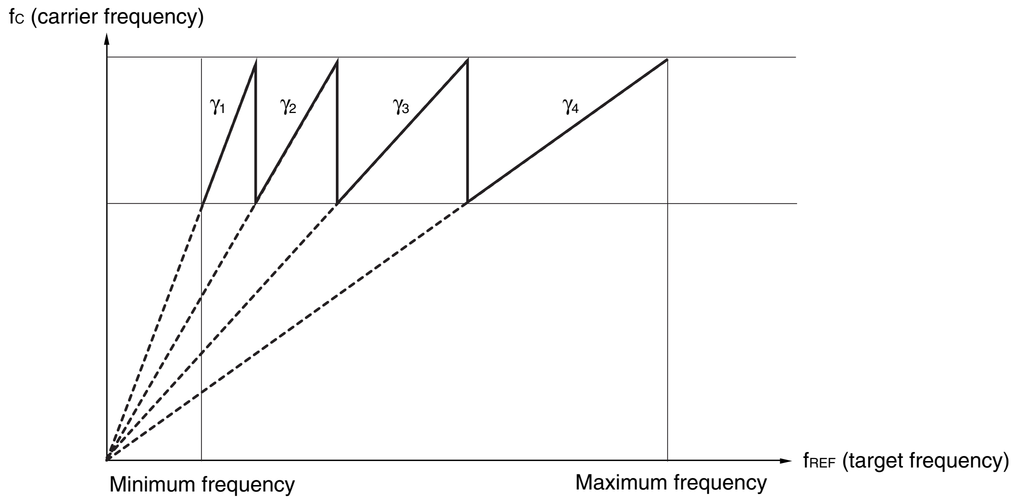
Let's see how to select a carrier frequency (f_C) next.

Carrier frequency f_C is calculated by the expression $f_C = \gamma f_{REF}$, where f_{REF} is the target frequency and γ is a proportional constant that is determined by the system.

An integer is usually selected (a multiple of 12 in this application system) as this proportional constant so that the relative positions of the carrier wave and sine wave are the same.

The proportional constant γ must be a sufficiently large value because more noise (shift from sine wave) is included in the reference sine wave as the value of the constant decreases. On the other hand, however, the switching loss of the inverter increases as the value of the constant increases. Therefore, proportional constant γ is determined for each target frequency (f_{REF}) region with an upper limit and a lower limit of the carrier frequency (f_C) determined (refer to **Figure 1-4**).

Figure 1-4. Relationship Between Revolving Frequency and Carrier Frequency



If proportional constant γ is determined, the carrier frequency (f_C) vis-à-vis the target frequency (f_{REF}) is determined, and the pulse width of the carrier wave is determined.

The phase can be calculated by integrating $1/\gamma f_{REF}$, and the set value of PWM can be expressed as follows.

$$V = \sqrt{2}\,(af_{REF} + b)\sin\left(2\pi f_{REF}\left(\int \frac{1}{\gamma f_{REF}}\right) dt\right)$$

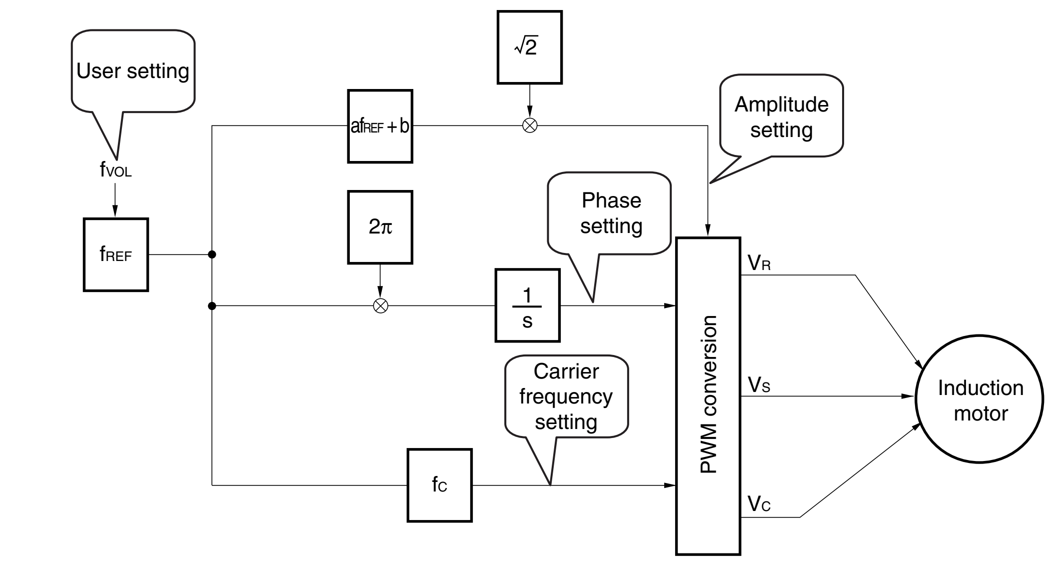
Remark a: V/f ratio

b: Offset value

f_{REF}: Target frequency

The control block diagram is shown below.

Figure 1-5. Control Block Diagram



Remark

- a: V/f ratio
- b: Offset value
- $\frac{1}{s}$: Laplace inverse-transformation operator
- f_{VOL}: Set frequency
- f_{REF}: Target frequency
- f_C: Carrier frequency
- V_R: R-phase voltage
- V_S: S-phase voltage
- V_C: C-phase voltage

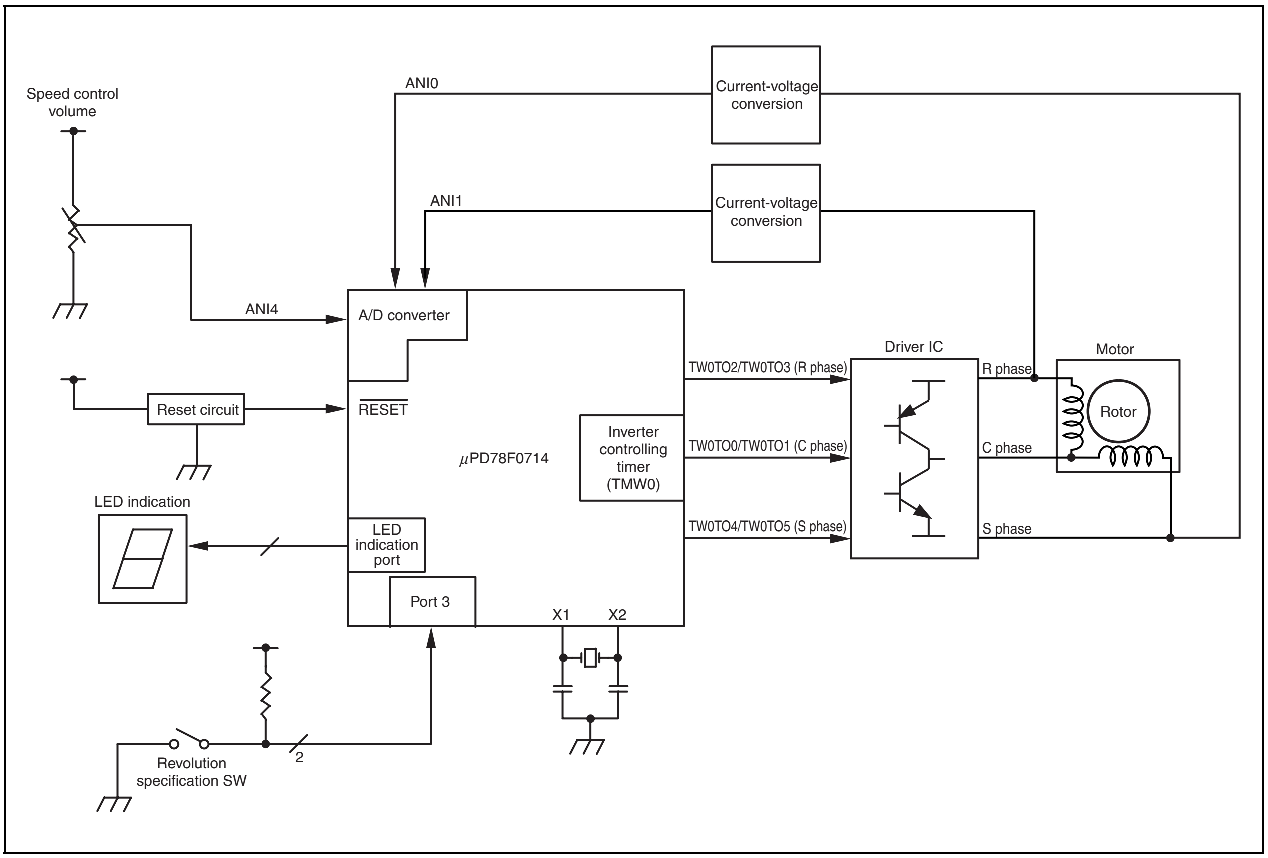
CHAPTER 2 HARDWARE CONFIGURATION

This chapter explains the hardware configuration.

2.1 Configuration

The major functions of this application system are illustrated below. In this application system, a single-phase induction motor starts revolving when a revolution specification SW is pressed after power application.

Figure 2-1. Overall System Configuration



(1) Speed controlling volume
Volume for increasing or decreasing the number of revolutions of the motor

(2) Revolution specification SW
CW and STOP switch

(3) LED indication
LED indicating the number of revolutions and errors

(4) Drive IC
Motor drive IC

(5) Current-voltage converter circuit
Converts the driving current of the motor into a voltage and is used to detect an overcurrent.

CHAPTER 3 SOFTWARE CONFIGURATION

This chapter explains the software configuration.

3.1 Peripheral I/O

This application system uses the following peripheral I/Os.

Table 3-1. Peripheral I/Os Used

Function	Peripheral I/O Function Name (μPD78F0714)
Inverter timer	Timer W0 (TMW0)
Overcurrent detection	ANI0, ANI1
Setting speed (volume)	ANI4
CW key input	P30
STOP key input	P33
LED output	P47 to P40, P57 to P50

(1) Description of peripheral I/O functions

(a) Inverter timer

PWM waveform is output by using an inverter timer.

The software of this application is set as follows.

- Inverter timer output: Low active
- Dead time: 3 μs
- Symmetrical triangular wave mode
- The carrier frequency is set as follows, depending on the set speed.

Set Speed	Carrier Frequency
Up to 1200 rpm	6.5 kHz
1201 to 2400 rpm	13 kHz
2401 to 3600 rpm	19.5 kHz
3601 to 4800 rpm	26 kHz

(b) Overcurrent detection

Overcurrent is detected by using ANI0 and ANI1.

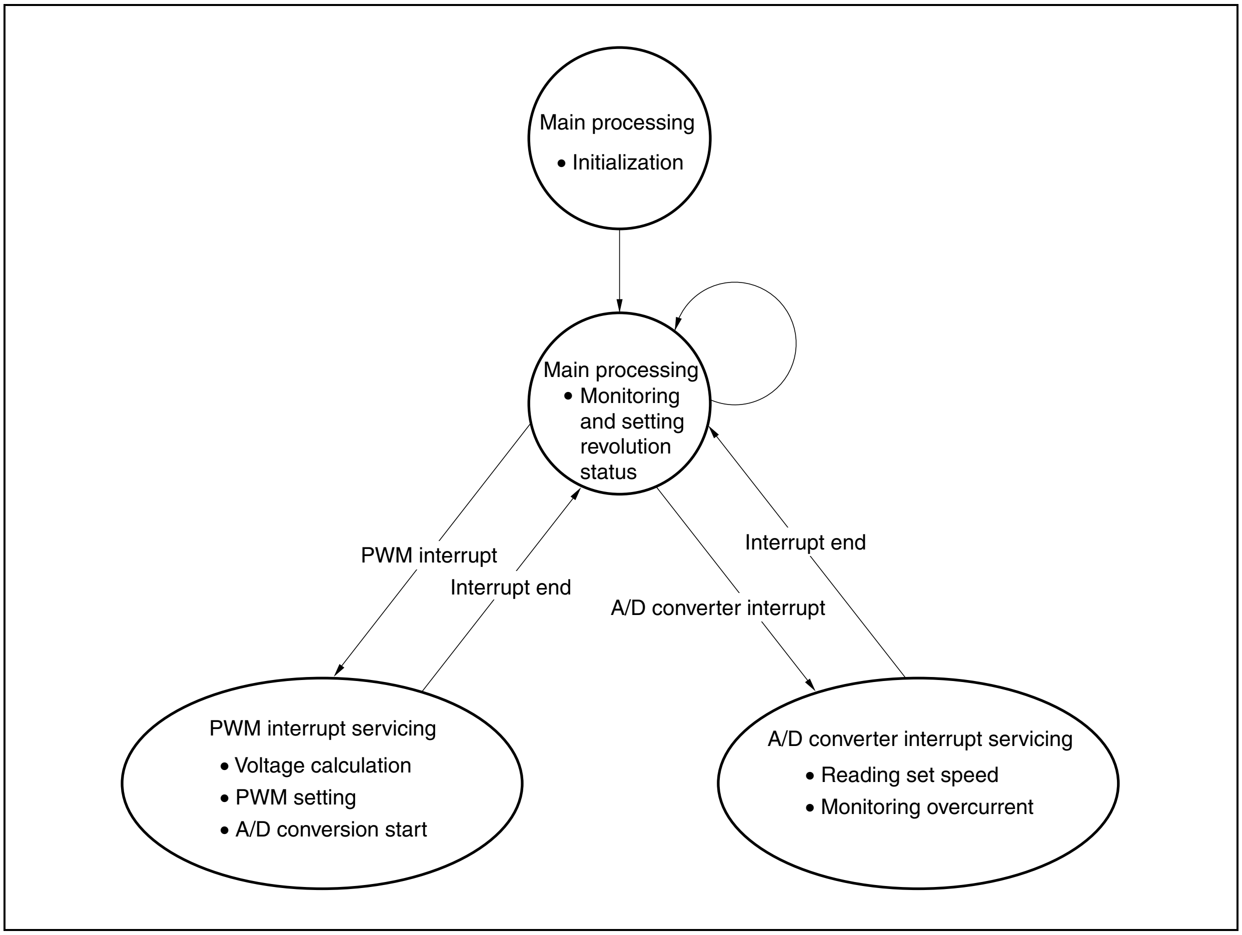
(c) Setting speed (volume) input

ANI4 is used to set a speed.

3.2 Software Processing Structure

The software processing structure is illustrated below.

Figure 3-1. Software Processing Structure



The software of this application consists of the following three types of processing.

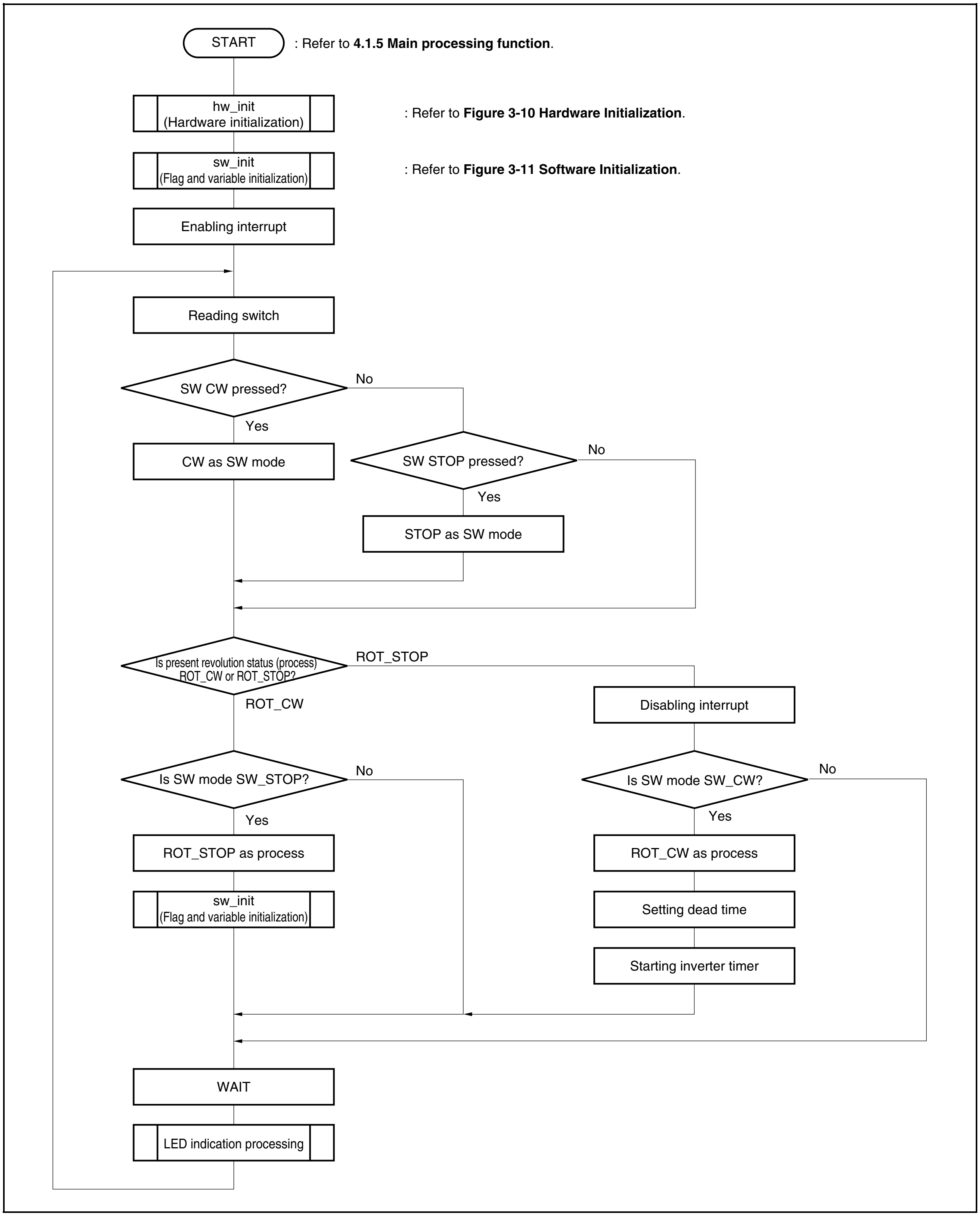
- Main processing
  Initializes the system and sets a revolution status (CW/STOP).
- PWM interrupt servicing
  Sets the voltage value of each phase (R and S phases) to an inverter timer.
- A/D converter interrupt servicing
  Reads the set speed (ANI4) and monitors the overcurrent (ANI0 and ANI1).

3.3 Flowchart

3.3.1 Main processing

Figure 3-2 shows the flowchart of the main processing.

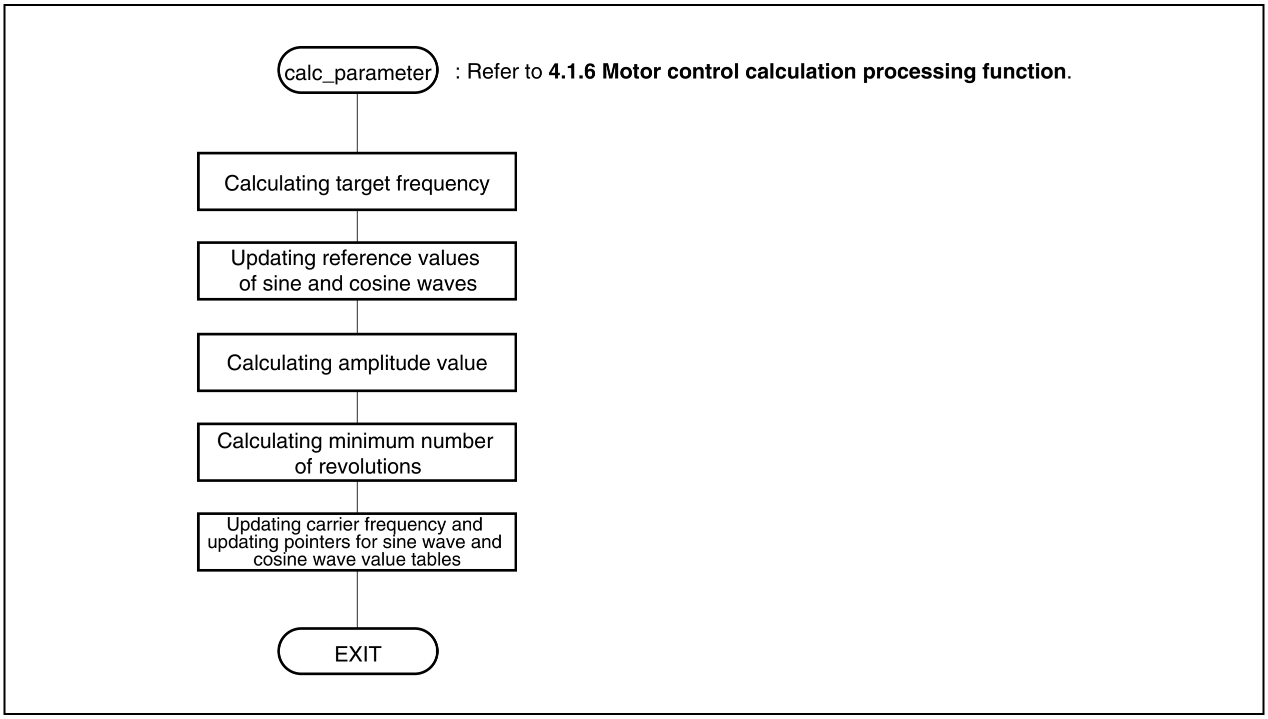
Figure 3-2. Main Processing



3.3.2 Motor control calculation processing

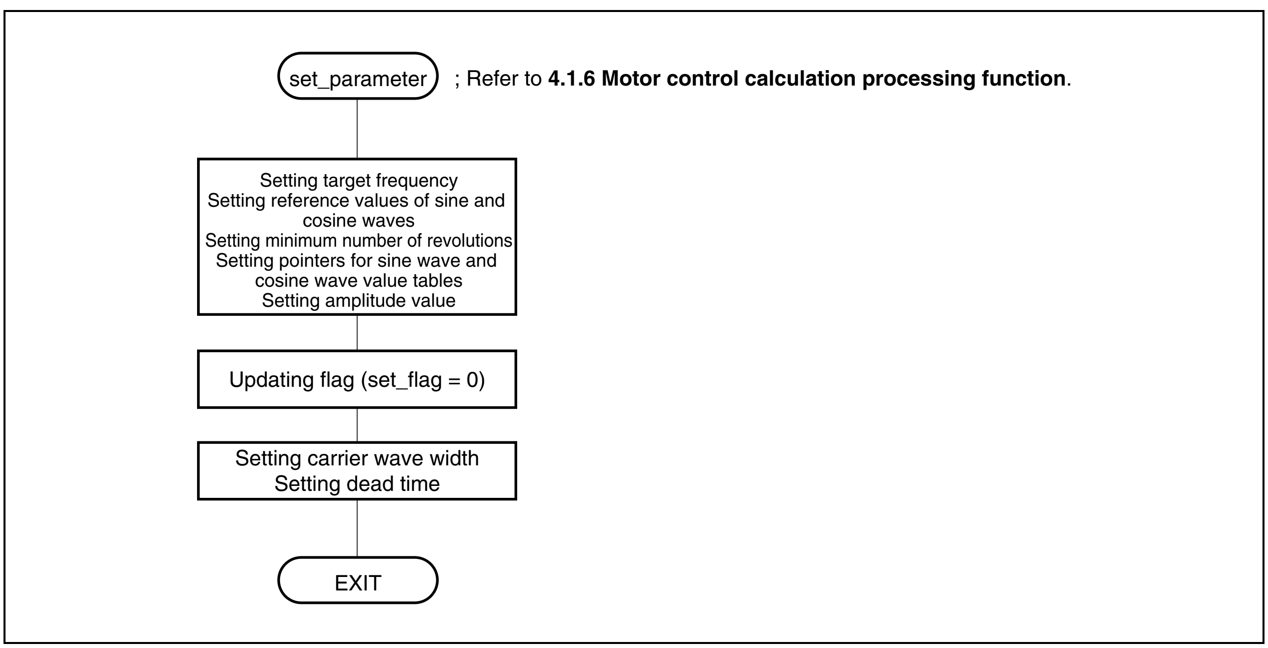
(1) calc_parameter function

Figure 3-3. calc_parameter Function



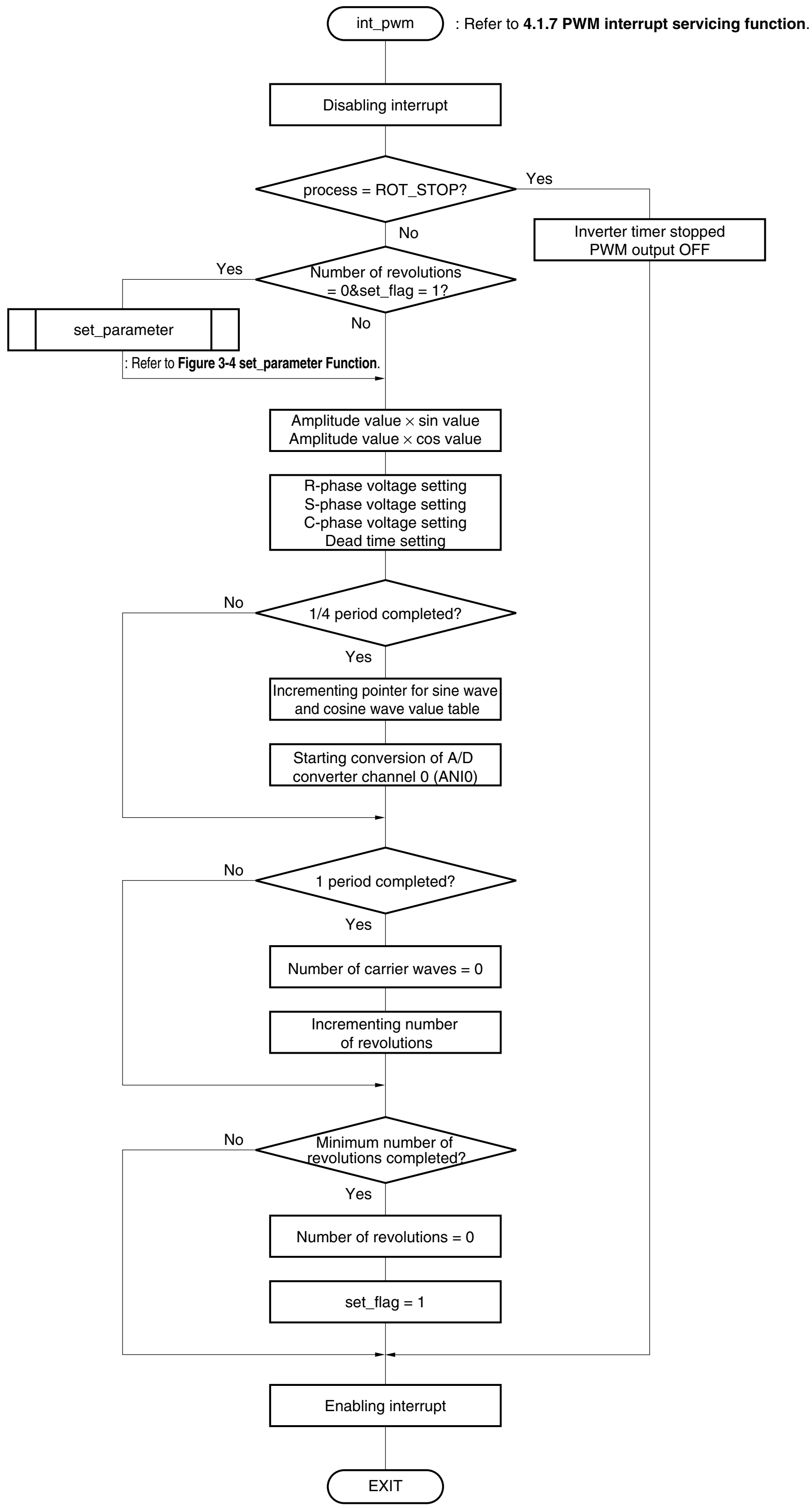
(2) set_parameter function

Figure 3-4. set_parameter Function



3.3.3 PWM interrupt servicing

Figure 3-5. PWM Interrupt Servicing



3.3.4 A/D converter interrupt servicing

Figure 3-6. A/D Converter Interrupt Servicing

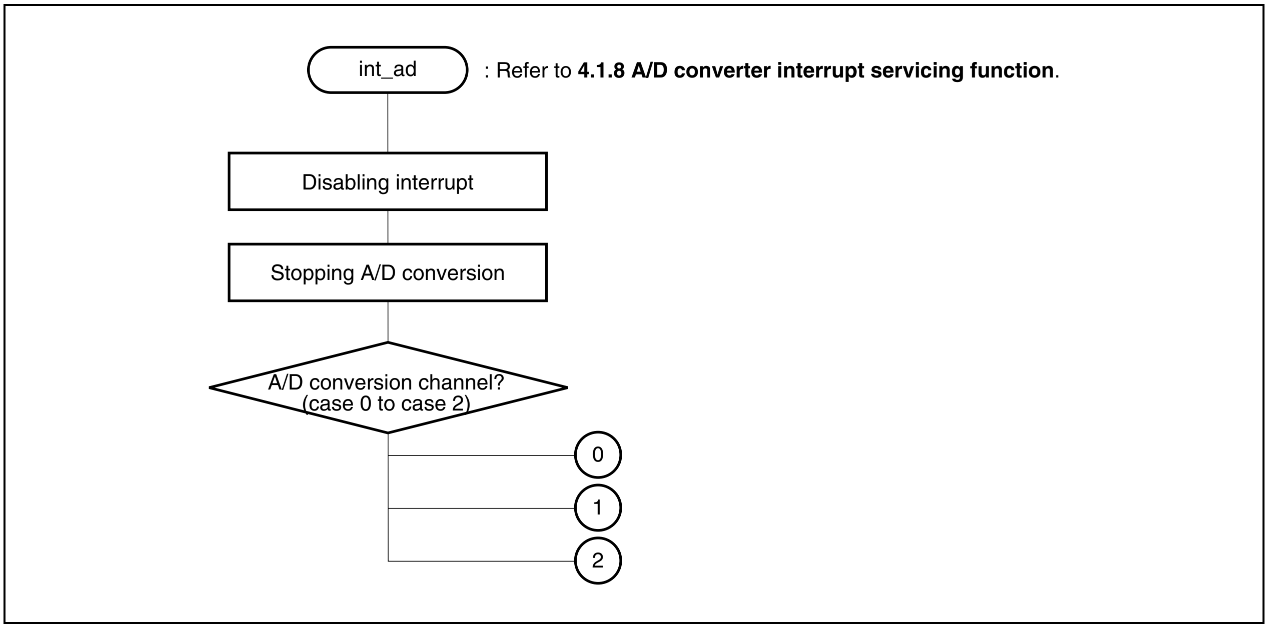


Figure 3-7. case 0 (A/D Converter Channel 1 Interrupt Servicing)

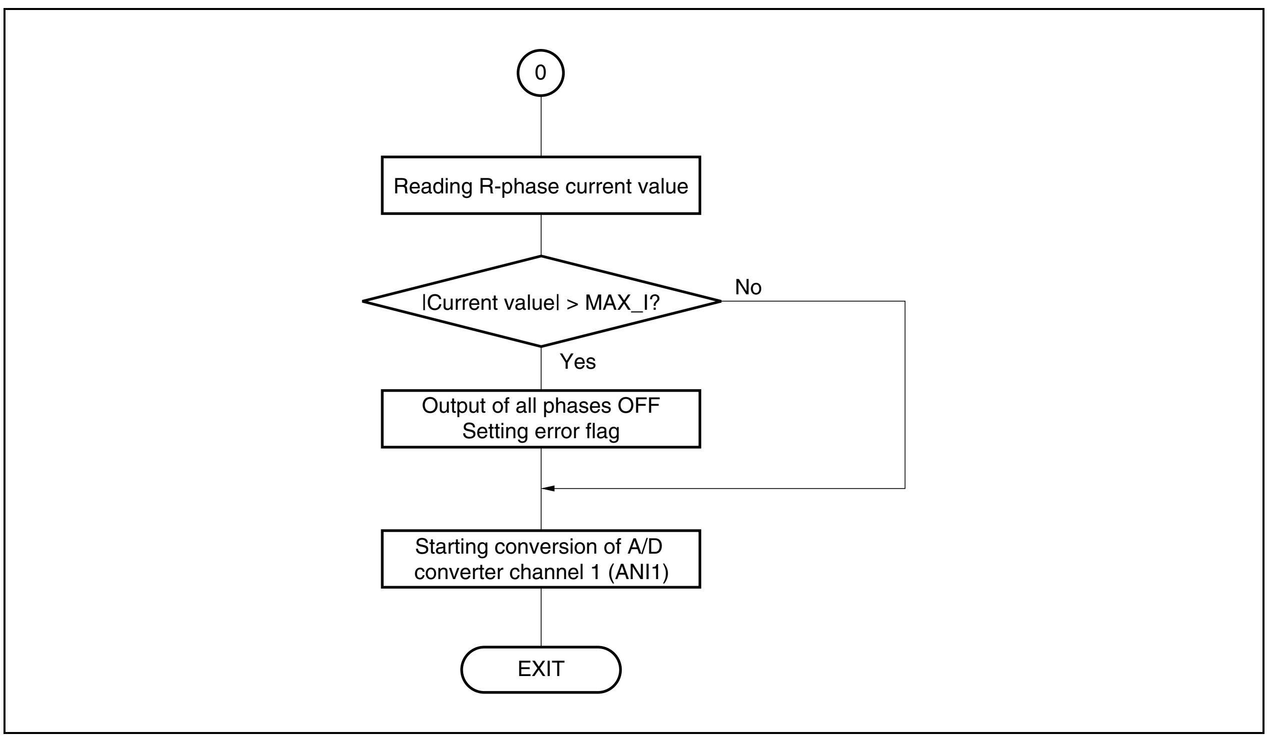


Figure 3-8. case 1 (A/D Converter Channel 4 Interrupt Servicing)

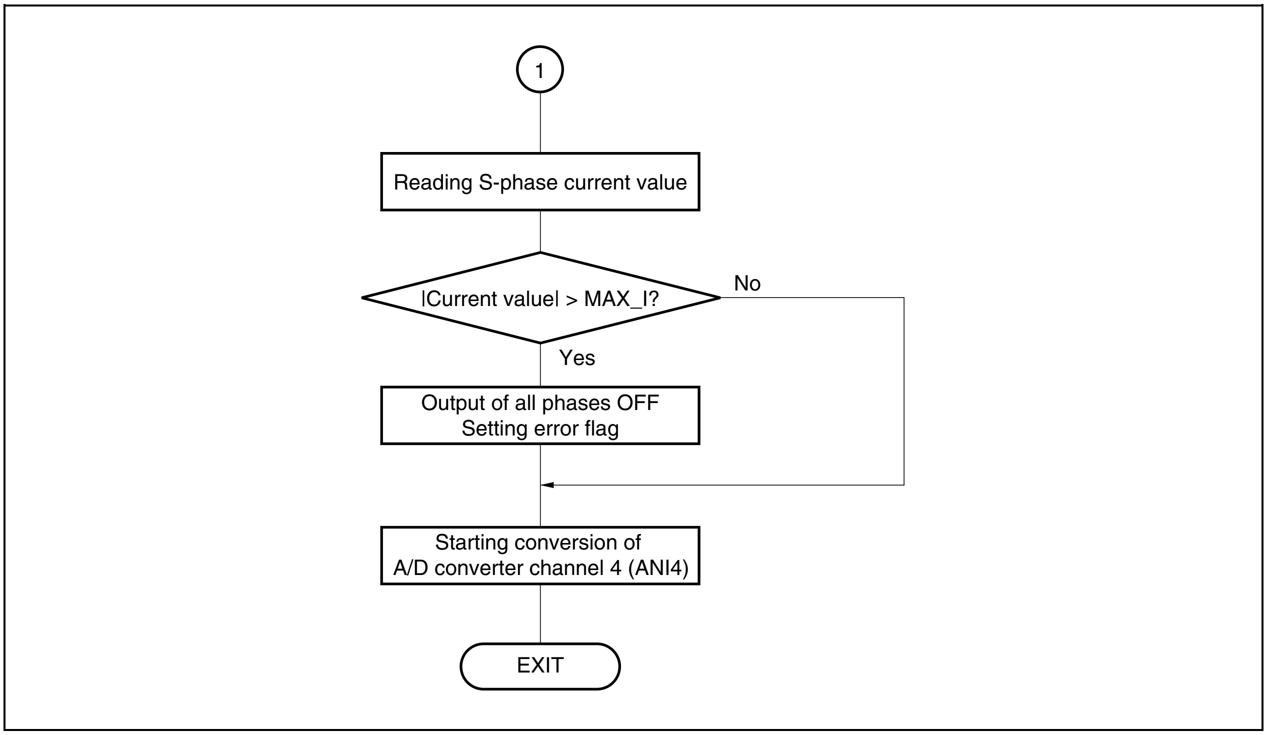
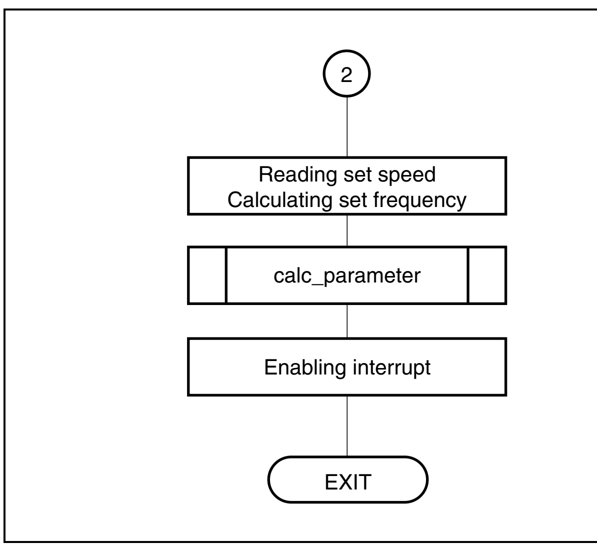


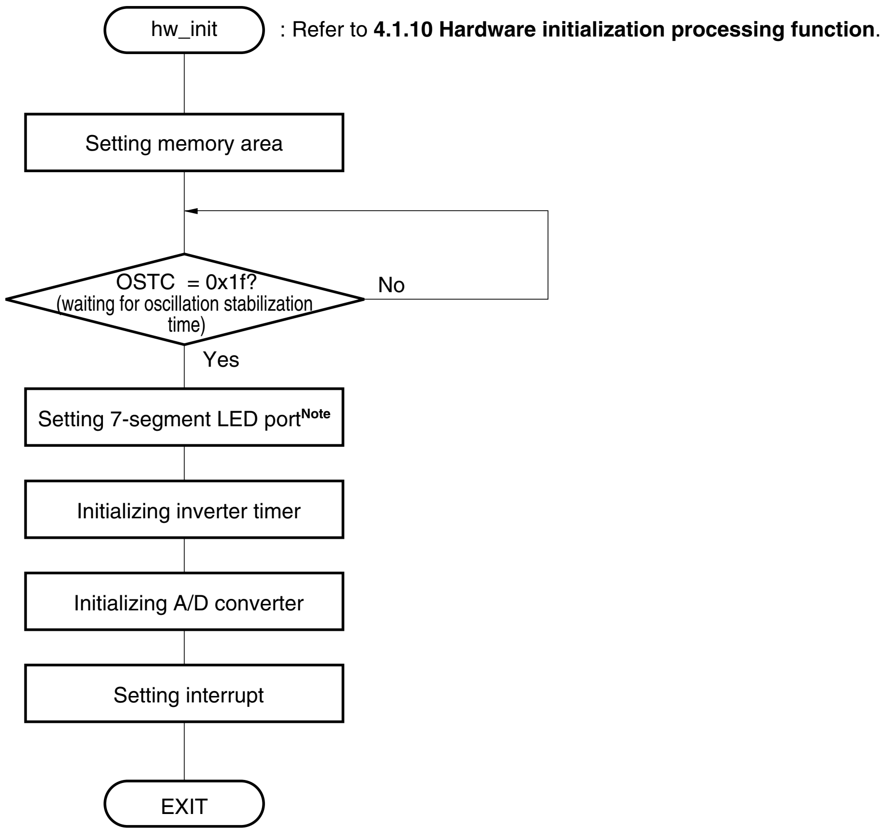
Figure 3-9. case 2 (Motor Control Calculation Processing)



: Refer to **Figure 3-3 calc_parameter Function**.

3.3.5 Hardware initialization

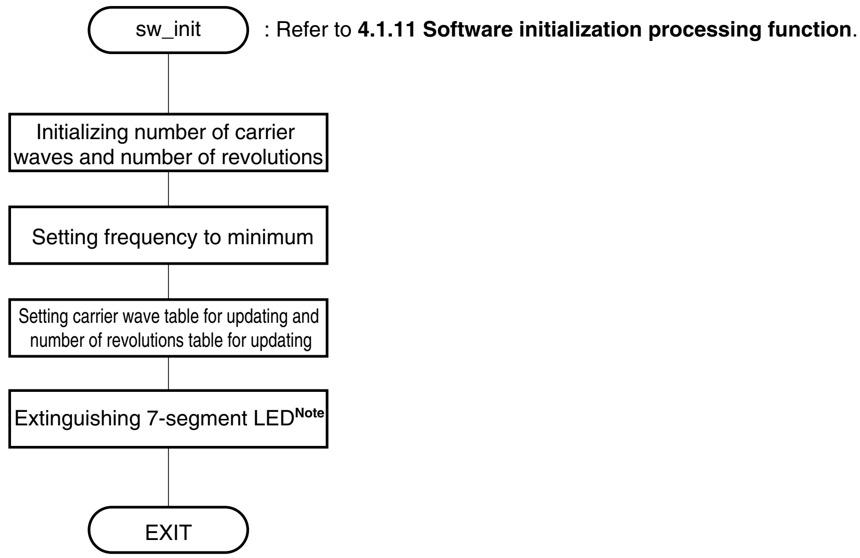
Figure 3-10. Hardware Initialization



Note This setting is under NEC Electronics' environment. The setting differs depending on the environment.

3.3.6 Software initialization

Figure 3-11. Software Initialization



Note This setting is under NEC Electronics' environment. The setting differs depending on the environment.

3.4 Tables

(1) Number of carrier waves table data

This is a table that sets the number of carrier waves.

```
unsigned short set_carrier[ 64 ] = {191,180,171,162,310,295,283,271,260,250,
                                    241,232,224,217,210,203,197,191,186,180,
                                    175,171,166,162,238,232,227,221,217,212,
                                    207,203,199,195,191,187,184,180,177,174,
                                    171,168,165,162,213,210,206,203,200,197,
                                    194,191,188,186,183,180,178,175,173,171,
                                    169,166,164,162}
```

(2) Sine wave value table data

This is a table that sets sinθ value, where θ is 0° to 90° (sin value multiplied by 512).

```
unsigned short sinwt[ 96 ] = {0, 8, 16, 25, 33, 41,50, 58, 66, 75, 83, 91, 99,108,116,
                              124,132,140,148,156,164,172,180,188,195,203,211,
                              218,226,233,241,248,256,263,270,277,284,291,298,
                              304,311,318,324,331,337,343,349,356,362,367,373,
                              379,384,390,395,401,406,411,416,421,425,430,434,
                              439,443,447,451,455,459,462,466,469,473,476,479,
                              482,484,487,489,492,494,496,498,500,502,503,505,
                              506,507,508,509,510,510,511,511,511}
```

(3) Cosine wave value table data

This is a table that sets cosθ value, where θ is 0° to 90° (cos value multiplied by 512).

```
unsigned short coswt[ 96 ] = {512,511,511,511,510,510,509,508,507,506,505,503,
                              502,500,498,496,494,492,489,487,484,482,479,476,
                              473,469,466,462,459,455,451,447,443,439,434,430,
                              425,421,416,411,406,401,395,390,384,379,373,367,
                              362,356,349,343,337,331,324,318,311,304,298,291,
                              284,277,270,263,255,248,241,233,226,218,211,203,
                              195,188,180,172,164,156,148,140,132,124,116,108,
                              99,91,83,75,66,58,50,41,33,25,16,8}
```

3.5 Constant Definition

The major constants used in this application system are listed below.

Symbol	Usage	Value
F_INV1	Constant defining range of carrier wave	21
F_INV2	Constant defining range of carrier wave	41
F_INV3	Constant defining range of carrier wave	61
FCBYF1	Number of carrier frequencies per period: Up to 20 Hz	384
FCBYF2	Number of carrier frequencies per period: 21 Hz to 40 Hz	192
FCBYF3	Number of carrier frequencies per period: 41 Hz to 60 Hz	128
FCBYF4	Number of carrier frequencies per period: 61 Hz to 80 Hz	96
TERM	Number of carrier frequencies/4	FCBYF1/4
VBYF_RATE	V/f ratio × 2 × 1.4142	3
VBYF_OFFSET	V/f ratio × 2 × 1.4142	55
NUM_POLE	Polarity logarithm	2
MIN_FREQ	Minimum value of set frequency	17
MAX_FREQ	Maximum value of set frequency	80
WAIT	Wait	10
MAX_I	Maximum current value	800
ADM_DEF	Setting to stop A/D conversion, setting of A/D conversion speed, setting of A/D conversion standby mode	09H
ADM_START	Starting A/D conversion	89H
ADM_STOP	Stopping A/D conversion	09H

CHAPTER 4 PROGRAM LIST

4.1 Program List (for μPD78F0714)

4.1.1 Program processing definitions

```
#pragma sfr
#pragma EI
#pragma DI
#pragma INTERRUPT INTTW0UD     int_pwm  rb1
#pragma INTERRUPT INTAD        int_ad   rb2
```

4.1.2 Header file

```
/*****************************************************************************/
/*      Header file                                                          */
/*****************************************************************************/
#include <stdlib.h>
#include "table.h"        /* sin value, cos value, carrier wave width table */
```

4.1.3 Constant definitions

```
/*****************************************************************************/
/*      External I/O definitions                                             */
/*****************************************************************************/
#define BASE_IO     0xc200000

#define WRESET      0x0d                /* Watchdog timer */
#define SW          0x0e                /* SW */
#define DIPSW       0x0f
#define DA1         0x0a
#define DA2         0x0b
#define DA3         0x0c
#define MODE        0x10

#define LED11       0x03                /* LED */
#define LED12       0x02                /* LED */
#define LED13       0x01                /* LED */
#define LED14       0x00                /* LED */
#define LED21       0x05                /* LED */
#define LED22       0x04                /* LED */
#define LED31       0x07                /* LED */
#define LED32       0x06                /* LED */
#define LED41       0x09                /* LED */
#define LED42       0x08                /* LED */
```

```
/******************************************************************************/
/*      Constant definitions                                                  */
/******************************************************************************/
#define F_INV1      21
#define F_INV2      41
#define F_INV3      61

#define FCBYF1      384         /* Number of carrier frequencies per period up to 20 Hz */
#define FCBYF2      192         /* Number of carrier frequencies per period 21 Hz to 40 Hz */
#define FCBYF3      128         /* Number of carrier frequencies per period 41 Hz to 60 Hz */
#define FCBYF4      96          /* Number of carrier frequencies per period 61 Hz to 80 Hz */

#define TERM        FCBYF1 >> 2         /* Carrier frequency/4 */

#define VBYF_RATE   3                   /* V/f ratio × 2 × 1.4142 */
#define VBYF_OFFSET 55                  /* V/f ratio × 2 × 1.4142 */

#define DEAD_TIME   0x0a                /* Dead time */

#define NUM_POLE    2                   /* Polarity logarithm */
#define MIN_FREQ    17                  /* Minimum value of set frequency */
#define MAX_FREQ    80                  /* Maximum value of set frequency */

#define WAIT        10                  /* Wait */
#define MAX_I       800                 /* Maximum current value */

#define LED_1       1                   /* LED number */
#define LED_2       2                   /* LED number */
#define LED_3       3                   /* LED number */
#define LED_4       4                   /* LED number */

#define LOW         0
#define HIGH        1

#define ADM_DEF     9                   /* Stopping A/D conversion and setting A/D conversion speed */
                                        /* Setting A/D conversion standby mode */
#define ADM_START   (ADM_DEF|0x80)      /* Starting A/D conversion */
#define ADM_STOP    ADM_DEF             /* Stopping A/D conversion */

#define CE0_ACTIVE  1

/******************************************************************************/
/*      Flag definitions                                                      */
/******************************************************************************/
unsigned short  error_flag;             /* Error flag */
unsigned short  set_flag;               /* Parameter set flag */

#define ERROR1      1                   /* Overcurrent error */
```

```
#define ERROR2       2                  /* Speed difference error */

#define SW_STOP      0                  /* Stopping revolution */
#define SW_CW        1                  /* Forward revolution */
#define SW_CCW       2                  /* Reverse revolution */

#define ROT_STOP     0                  /* Stopping revolution */
#define ROT_CW       1                  /* Forward revolution */
#define ROT_CCW      2                  /* Reverse revolution */
```

4.1.4 Symbol definitions

```
/******************************************************************************/
/*      Variable definitions                                                  */
/******************************************************************************/
unsigned char    sw_mode;         /* Specified number of revolutions */
unsigned char    process;         /* Revolution status */

unsigned short   f_ref_temp;      /* Target frequency */
unsigned short   offset;          /* sin and cos reference values, set carrier wave frequency value/2 */
unsigned char    ptr_rate;
unsigned short   num_pulse;       /* Number of carrier waves */
unsigned short   v_amp;           /* Set voltage amplitude value */
unsigned short   default_rot;     /* Minimum number of revolutions */
unsigned short   n_f_ref_temp;    /* Target frequency: for updating */
unsigned short   n_offset;        /* sin and cos reference values, set carrier wave frequency value/2: for updating */
unsigned char    n_ptr_rate;
unsigned short   n_num_pulse;     /* Number of carrier waves: for updating */
unsigned short   n_v_amp;         /* Set voltage amplitude value: for updating */
unsigned short   n_default_rot;   /* Minimum number of revolutions: for updating */

unsigned short   num_carrier;     /* Number of carrier waves */
unsigned short   num_nc;          /* Number of carrier wave counts */
unsigned short   term;            /* 1/4 revolution */
unsigned short   rot_num;         /* Number of revolutions */

signed short     sgn_ph;          /* Variable for phase ratio */
signed short     sgn_sin_amp;     /* sin amplitude value */
signed short     sgn_cos_amp;     /* cos amplitude value */

unsigned short   ex_set_value_c;
unsigned short   ex_set_value_r;
unsigned short   ex_set_value_s;

unsigned short   f_ref;           /* Set frequency */

unsigned short   v_r;
unsigned short   v_s;
```

```
unsigned long    sv;
unsigned long    cv;

const unsigned short led_pat[10] = { 0xfc, 0x60, 0xda, 0xf2, 0x66, 0xb6, 0xbe, 0xe0, 0xfe,
                                     0xe6 };

extern unsigned short      set_carrier[ 64 ];
extern unsigned short      sinwt[ 96 ];
extern unsigned short      coswt[ 96 ];
```

4.1.5 Main processing function

```
/********************************************************************************/
/*      Function definitions                                                    */
/********************************************************************************/
void main( void );                              /* Main */
void calc_parameter( void );                    /* Calculation of control parameter */
void set_parameter( void );                     /* Updating control parameter */

void hw_init( void );                           /* Hardware initialization */
void sw_init( void );                           /* Software initialization */
void output_data( unsigned short reg, unsigned short data );        /* I/O output */
unsigned short input_data( unsigned short reg );                    /* I/O input */
void led( unsigned short led_number, unsigned short led_data );     /* LED */

/********************************************************************************/
/*      Induction motor V/f control program main processing                     */
/********************************************************************************/
void main( void )
{
unsigned short    ic;
unsigned short    sw_set;

/* Initialization */
    hw_init();                                  /* Hardware initialization */
    sw_init();                                  /* Software initialization */

    EI();                                       /* Enabling interrupt */

    while( 1 ) {

    sw_set =  ~input_data( SW ) & 0x07;         /* Reading operation button */
        if  ( sw_set & 0x01 ) {
            sw_mode = SW_CW;
        } else if ( sw_set & 0x02 ) {
            sw_mode = SW_CCW;
        } else if ( sw_set & 0x04 ) {
            sw_mode = SW_STOP;
```

```
        }

        switch ( process ) {
            case ROT_STOP:
                DI();
                if ( sw_mode == SW_CW ) {
                    process = ROT_CW;
                    TW0DTIME = DEAD_TIME;       /* Dead time 3 us */
                    CE0 = 1;
                } else if ( sw_mode == SW_CCW ) {
                    process = ROT_CW;           /* Only single direction */
                    TW0DTIME = DEAD_TIME;       /* Dead time 3 us */
                    CE0 = 1;
                }
                EI();
                break;

            case ROT_CW:
                if ( (sw_mode == SW_CCW) || (sw_mode == SW_STOP) ) {
                    process = ROT_STOP;
                    sw_init();
                }
                break;

            case ROT_CCW:
                if ( (sw_mode == SW_CW) || (sw_mode == SW_STOP) ) {
                    process = ROT_STOP;
                    sw_init();
                }
                break;
        }

        ic = WAIT;
        while(ic--);

/* LED indication */
        led( LED_2, f_ref_temp );               /* LED indication */

        if ( error_flag ) {
            CE0 = 0;                            /* Disabling output of all phases of PWM */
            process = ROT_STOP;
            led( LED_1, 1000 | error_flag );    /* LED indication */
        }
    }
}
```

4.1.6 Motor control calculation processing function

```
/******************************************************************************/
/*     Updating inverter frequency, carrier wave, and amplitude value        */
/******************************************************************************/
void set_parameter( void )
{

/* Updating inverter frequency 17 Hz to 80 Hz */
       f_ref_temp = n_f_ref_temp;

/* Setting sin and cos offset values */
       offset = n_offset;               /* Equivalent to 1/4 of carrier wave */
       num_pulse = n_num_pulse;         /* Setting number of carrier waves */
       ptr_rate = n_ptr_rate;

       TW0CM3 = offset << 1;
       TW0BFCM3 = offset << 1;

/* Amplitude value setting */
       v_amp = n_v_amp;                 /* V/f ratio × 2 */

/* Setting minimum number of revolutions */
       default_rot = n_default_rot;

       set_flag = 0;                    /* Clearing parameter set flag */

       num_carrier= 0;                  /* Clearing number of carrier waves */
       num_nc = 0;                      /* Clearing number of carrier wave counts */
       term = 1;                        /* Clearing 1/4 of number of periods */

}

/******************************************************************************/
/*     Calculating inverter frequency, carrier wave, and amplitude value     */
/******************************************************************************/
void calc_parameter( void )
{

/* Updating inverter frequency 7 Hz to 70 Hz */
       if ( f_ref > n_f_ref_temp ) {
              n_f_ref_temp = n_f_ref_temp + 1;
       } else if ( f_ref < n_f_ref_temp ) {
              n_f_ref_temp = n_f_ref_temp - 1;
       }
       if ( n_f_ref_temp < MIN_FREQ ) n_f_ref_temp = MIN_FREQ;
       if ( n_f_ref_temp > MAX_FREQ ) n_f_ref_temp = MAX_FREQ;
```

```
/* Setting sin and cos value offsets */
       n_offset = set_carrier[n_f_ref_temp - MIN_FREQ];      /* Equivalent to 1/4 of carrier wave */

/* Setting amplitude value */
       n_v_amp = VBYF_RATE * n_f_ref_temp + VBYF_OFFSET;     /* V/f ratio × 2 */

/* Setting minimum number of revolutions */
       n_default_rot = (n_f_ref_temp >> 2) & 0x3fff;  /* 1/4 times of frequency to approx. 0.25 s */

       if ( n_f_ref_temp < F_INV1 ) {                 /* Up to 20 Hz */
              n_num_pulse = FCBYF1;                   /* Setting number of carrier waves */
              n_ptr_rate = 1;
              n_v_amp += 25;
       } else if ( n_f_ref_temp < F_INV2 ) {          /* 21 Hz to 40 Hz */
              n_num_pulse = FCBYF2;                   /* Setting number of carrier waves */
              n_ptr_rate = 2;
              if ( n_f_ref_temp < 44 ) n_v_amp += 55;
       } else if ( n_f_ref_temp < F_INV3) {           /* 41 Hz to 60 Hz */
              n_num_pulse = FCBYF3;                   /* Setting number of carrier waves */
              n_ptr_rate = 3;
       } else {                                       /* 61 Hz to 80 Hz */
              n_num_pulse = FCBYF4;                   /* Setting number of carrier waves */
              n_ptr_rate = 4;
              if ( n_v_amp > 325 ) n_v_amp = 325;
       }
       n_ptr_rate *= NUM_POLE;                        /* Polarity logarithm */

}
```

4.1.7 PWM interrupt servicing function

```
/********************************************************************************/
/*     PWM interrupt servicing                                                  */
/********************************************************************************/
_ _interrupt void int_pwm(void)
{
       DI();

       if ( process == ROT_STOP ) {      /* Stop */
              CE0 = 0;                   /* Disabling output of all phases of PWM */
       } else {

              if ( (rot_num == 0) && (set_flag == 1) ) set_parameter();/*Updating control parameters*/

              sv = sinwt[ num_carrier ];
              cv = coswt[ num_carrier ];
              v_r = (unsigned short)(((unsigned long)v_amp * sv) >> 10);
              v_s = (unsigned short)(((unsigned long)v_amp * cv) >> 10);
```

```
            if ( ( v_r < offset ) || ( v_s < offset ) ) {
                TW0BFCM0 = offset;                          /* C phase: Constant value */
                TW0BFCM1 = offset + sgn_sin_amp * v_r;      /* R phase: Basic sine wave */
                TW0BFCM2 = offset + sgn_cos_amp * v_s;      /* S phase: π/2 delay */

                ex_set_value_c = offset;                    /* C phase: Constant value */
                ex_set_value_r = offset + sgn_sin_amp * v_r;  /* R phase: Basic sine wave */
                ex_set_value_s = offset + sgn_cos_amp * v_s;  /* S phase: π/2 delay */
            } else {

                TW0BFCM0 =ex_set_value_c;                   /* C phase: Constant value */
                TW0BFCM1 =ex_set_value_r;                   /* R phase: Basic sine wave */
                TW0BFCM2 =ex_set_value_s;                   /* S phase: π/2 delay */
            }

            if ( (num_nc + ptr_rate) >= TERM ) {    /* 1/4 period */
                sgn_sin_amp = sgn_ph * sgn_sin_amp;
                sgn_cos_amp = - sgn_ph * sgn_cos_amp;
                sgn_ph = - sgn_ph;
                term++;             /* Incrementing number of times of 1/4 period */
                ADS = 0x00;         /* Selecting ANI0 */
                ADM = ADM_START;    /* Starting A/D conversion */
                num_nc = 0;
            } else {
                num_carrier += sgn_ph * ptr_rate;   /* Incrementing carrier wave number */
                num_nc += ptr_rate;                 /* Incrementing number of carrier wave counts */
            }

            if ( term > 4 ) {               /* 1 period or less */
                num_carrier= 0;             /* Clearing number of carrier waves */
                num_nc = 0;                 /* Clearing number of carrier wave counts */
                term = 1;                   /* Clearing number of times of 1/4 period */
                rot_num++;                  /* Incrementing number of revolutions */
            }

            if ( rot_num == default_rot ) {
                rot_num = 0;        /* Minimum number of revolutions completed, clearing counter */
                set_flag = 1;       /* Parameter set flag */
            }
}

IF1L &= ~0x04;
EI();

}
```

4.1.8 A/D converter interrupt servicing function

```
/******************************************************************************/
/*     Set frequency A/D converter interrupt servicing                        */
/******************************************************************************/
_ _interrupt void int_ad(void)
{
signed short      iua;                  /* Current value */
signed short      iva;                  /* Current value */
unsigned short    vol;                  /* Set value */

    DI();

    ADM = ADM_STOP;
    switch( ADS & 0x07) {
        case 0x00:
            iua = (( ( ADCR >> 6 ) & 0x3ff ) - 0x200);
            if  ( abs(iua) > MAX_I ) {
                CE0 = 0;                      /* Disabling output of all phases of PWM */
                RTPM01 = 0x3f;
                error_flag = ERROR1;          /* Setting error number */
            }
            ADS = 0x01;                       /* Selecting ANI1 */
            ADM = ADM_START;                  /* Starting A/D conversion */
            break;
        case 0x01:
            iva = (( ADCR & 0x3ff ) - 0x200);
            if  ( abs(iva) > MAX_I ) {
                CE0 = 0;                      /* Disabling output of all phases of PWM */
                RTPM01 = 0x3f;
                error_flag = ERROR1;          /* Setting error number */
            }

            ADS = 0x04;                       /* Selecting ANI4 */
            ADM = ADM_START;                  /* Starting A/D conversion */
            break;
        default:
            vol = ~( ADCR >> 6 ) & 0x3ff;           /* Reading set value */
            f_ref = ((vol + 1) >> 4 ) + MIN_FREQ;   /* Calculating set frequency */
            EI();
            calc_parameter();                 /* Calculating control parameter */
            P01 = 0;                          /* Interrupt start output */
            break;
    }

    IF1H &= ~0x10;
    EI();
}
```

4.1.9 LED indication function

```
/*****************************************************************************/
/*      LED value indication subroutine                                      */
/*              no:   Indication area number (1 to 4)                       */
/*              data: Indication data (0 to 99)                              */
/*****************************************************************************/
void led( unsigned short led_number, unsigned short led_data )
{
unsigned short   led_i;
unsigned short   led_ro;
unsigned short   led_ha;

     switch ( led_number ) {
          case LED_1:
               led_i = led_data / 10;
               led_ro = led_i / 10;
               led_ha = led_ro / 10;
               led_data = led_data - led_i * 10;
               led_i = led_i - led_ro * 10;
               led_ro = led_ro - led_ha * 10;

               output_data( LED11, ~led_pat[led_ha] );
               output_data( LED12, ~led_pat[led_ro] );
               output_data( LED13, ~led_pat[led_i] );
               output_data( LED14, ~led_pat[led_data] );
               break;

          case LED_2:
               led_i = led_data/10;
               led_ro = led_data - 10 * led_i;

               output_data( LED21, ~led_pat[led_i] );
               output_data( LED22, ~led_pat[led_ro] );
               break;

          case LED_3:
               led_i = led_data/10;
               led_ro = led_data - 10 * led_i;

               output_data( LED31, ~led_pat[led_i] );
               output_data( LED32, ~led_pat[led_ro] );
               break;

          case LED_4:
               led_i = led_data/10;
               led_ro = led_data - 10 * led_i;
```

```
                output_data( LED41, ~led_pat[led_i] );
                output_data( LED42, ~led_pat[led_ro] );
                break;
        }
}

/*****************************************************************************/
/*      External I/O output                                                  */
/*              reg:  Output register number                                 */
/*              data: Output data                                            */
/*****************************************************************************/
void output_data( unsigned short reg, unsigned short data )
{
        P5 = (unsigned char)reg;
        P4 = (unsigned char)(data & 0xff);
        P7 = (unsigned char)(((data >> 8) & 0x07) | (P7 & 0xf8));
        PM4 = 0x00;
        PM7 = PM7 & 0xf8;
        P66 = 0;
        P66 = 1;
}

/*****************************************************************************/
/*      External I/O input                                                   */
/*              reg:  Input register number                                  */
/*              data: Input data                                             */
/*****************************************************************************/
unsigned short input_data( unsigned short reg )
{
        unsigned short data ;

                P5 = (unsigned char)reg;
                PM4 = 0xff;
                PM7 |= 0x07;
                P65 = 0;
                data = ((P7 << 8) & 0x07) | P4;
                P65 = 1;
                PM4 = 0x00;
                PM7 = PM7 & 0xf8;
        return data;
}
```

4.1.10 Hardware initialization processing function

```
/*****************************************************************************/
/*      Hardware (peripheral I/O) initialization                             */
/*****************************************************************************/
void hw_init (){
```

```
        IMS = 0xc8;                 /* Setting memory area */

        /* Setting main clock oscillation and high-speed mode */
        while(OSTC != 0x1f);        /* Waiting for oscillation stabilization time */
        WDTM = 0x77;
        MCM = 0x03;                 /* Selecting X1 input clock */

        /* Initializing FPGA access port mode register */
        PM7 = 0xf8;                 /* P70-P72 output */
        PM6 = 0x9f;                 /* P65,P66 output */
        PM5 = 0xe0;                 /* P50-P54 output */
        PM4 = 0x00;                 /* P40-P47 output */

        /* TMW0 */
        TW0C = 0x10;                /* Stopping, fX/4 -> 5 MHz, generated each time TW0UDC underflows */
        TW0M = 0x04;
        TW0OC = 0x00;               /* TW0TO0 to TW0TO5 enabled to output */

        /* A/D */
        ADM = 0x1d;                 /* Stopping A/D conversion, select mode */
                                    /* A/D conversion time = 3.6 us, A/D conversion standby mode */
        ADS = 0x00;
        PFM = 0x00;
        PFT = 0x00;

        /* External interrupt */
        EGP = 0x00;                 /* Rising  edge disable */
        EGN = 0x00;                 /* Falling edge disable */

        /* Masking interrupt */
        MK0L = 0xff;
        MK0H = 0xfd;
        MK1L = 0xff;
        MK1H = 0xcf;

        /* Priority */
        PR1H = 0xef;
}
```

4.1.11 Software initialization processing function

```
/******************************************************************************/
/*      Software (variable) initialization                                    */
/******************************************************************************/
void sw_init (){

        num_carrier = 0;                    /* Number of carrier waves */
```

```
        num_nc = 0;                         /* Number of carrier waves */
        term = 1;                           /* Number of 1/4 revolutions */
        rot_num = 0;                        /* Number of revolutions */
        offset = 96;                        /* sin and cos value offset */
        n_ptr_rate = 1;

        sgn_sin_amp = 1;
        sgn_cos_amp = 1;
        sgn_ph = 1;

        f_ref_temp = MIN_FREQ;              /* Target frequency */
        f_ref = MIN_FREQ;                   /* Set frequency */

        process = ROT_STOP;                 /* Revolution status flag */
        sw_mode = SW_STOP;                  /* Specified revolution flag */

        error_flag = 0;                     /* Error flag */
        set_flag = 1;

        n_f_ref_temp = MIN_FREQ;
        n_offset = 96;                      /* Equivalent to 1/4 of carrier wave */
        n_num_pulse = 384;                  /* Setting number of carrier waves */
        n_ptr_rate = 1;
        n_v_amp = 71;                       /* Setting amplitude value */
        n_default_rot = 4;                  /* Setting minimum number of revolutions */

        output_data( LED11, 0 );
        output_data( LED12, 0 );
        output_data( LED13, 0 );
        output_data( LED14, 0 );
        output_data( LED21, 0 );
        output_data( LED22, 0 );
        output_data( LED31, 0 );
        output_data( LED32, 0 );
        output_data( LED41, 0 );
        output_data( LED42, 0 );

}
```