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.

SH7145 Group

Acceleration/Deceleration Control for a Stepping Motor Using CMT

Introduction

This application uses the compare-match timer module (CMT) to control acceleration and deceleration of a stepping motor by 1-2 phase excitation. A stepping motor is repeatedly driven in this sequence: forward rotation → stop → reverse rotation → stop.

Target Device

SH7145F

Contents

1. Specifications

A two-phase stepping motor is controlled by the pulses which are generated using channel 0 of the CMT (Compare-Match Timer) and general ports (PC12 to PC15) of the SH7145F. In this sample task, control is through 1-2 phase excitation and the motor is driven repeatedly in the following sequence: forward rotation → stop → reverse rotation → stop. During the processes of motor rotation, slew-up and slew-down processing is performed.

Figure 1 shows the connections between the SH7145F and the stepping motor.

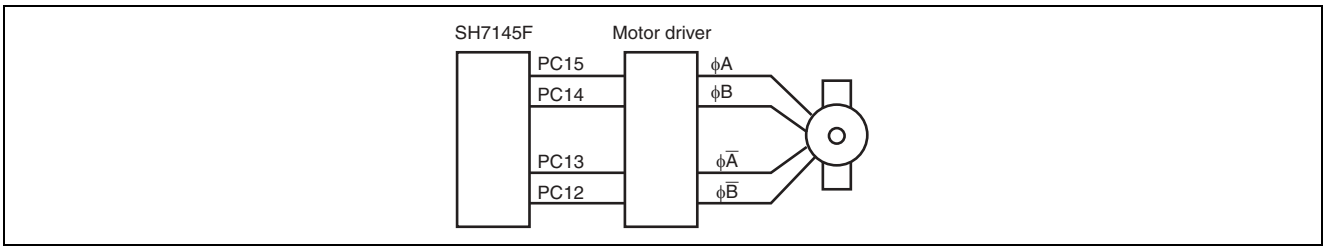


Figure 1 Connections Stepping Motor Control

2. Description of Functions

In this sample task, pulses for controlling the stepping motor is generated using the CMT and the general port (port C).

2.1 Stepping Motor

This sample task uses a permanent-type stepping motor (KP6P8-701 from JAPAN SERVO Co., Ltd.). Table 1 summarizes the standard specifications of the KP6P8-701.

Table 1 Standard Specifications of the Stepping Motor (KP6P8-701)

Item	Specification
Model	KP6P8-701
Number of phases	2
Step angle [deg./step]	7.5
Voltage [V]	12
Current [A/ϕ]	0.33
Resistance [Ω/ϕ]	36
Inductance [mH/ϕ]	28
Maximum static torque [mN • m (kgf • cm)]	78.4 (0.8)
Detent torque [mN • m(gf • cm)]	1.3 (180)
Rotor inertia [g • cm^2]	23.7

2.2 Compare-Match Timer (CMT)

The CMT generates an interrupt at the specified period. Figure 2 shows a block diagram of the CMT module channel 0 (ch0). Its functions are described below.

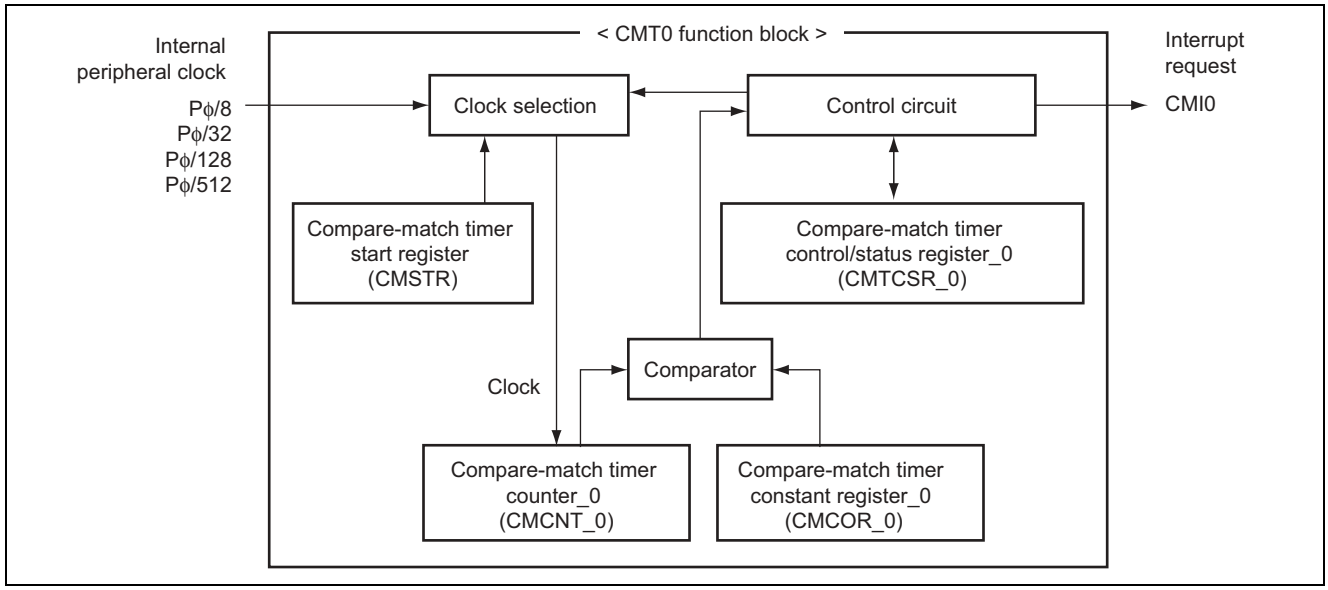


Figure 2 CMT (Channel 0) Block Diagram

- The CMT has a 16-bit counter and can generate an interrupt at a specified period.
- A clock generated by dividing the internal peripheral clock Pϕ can be selected, and the counter increments based on the selected clock.
- The compare-match timer start register (CMSTR) starts or stops counting.
- The compare-match timer control/status register (CMCSR_0) indicates a compare-match occurrence, sets up interrupts, and selects the clock for counting.
- The compare-match timer counter (CMCNT_0) is an up-counter used to generate interrupts.
- The compare-match timer constant register (CMCOR_0) specifies the period of compare-match generation.

2.3 General Port (Port C)

In this sample task, pulses for 1-2 phase excitation are output by general port (port C) control. PC12 to PC15 are used for pulse outputs. Figure 3 shows a block diagram of port C. The functions of port C are summarized below.

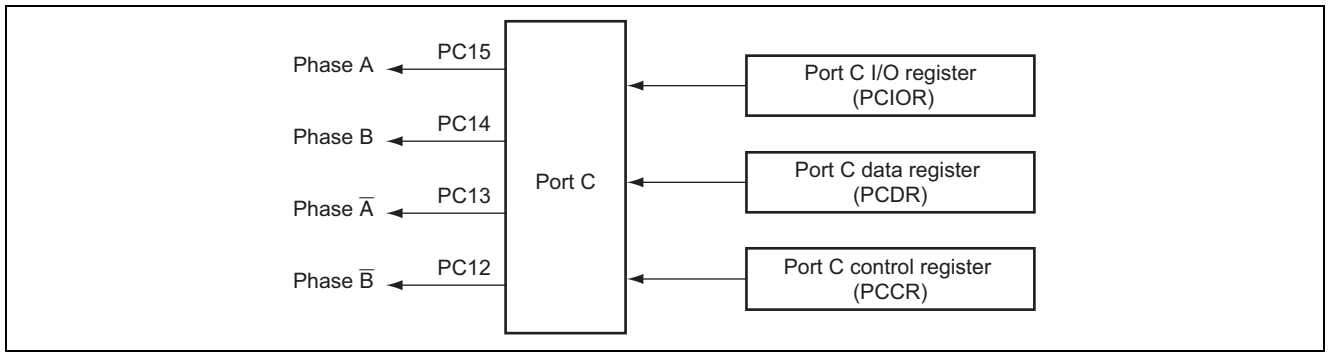


Figure 3 Port C Block Diagram

- Port C is a 16-bit general I/O port.
- The port C control register (PCCR) selects the functions of multiplexed pins.
- The port C I/O register (PCIOR) selects the input or output direction of the pins. The PCIOR is only valid when the pins of the port C function as general I/O pins; it is invalid otherwise.
- The port C data register (PCDR) stores data of port C. When general output function is selected, data written to PCDR is directly output from the corresponding pins. When general input function is selected, the states of the corresponding pins are directly read by reading the PCDR.

3. Description of Operation

3.1 Pulse Output for 1-2 Phase Excitation

Pulse are output through the use of CMT's interrupt function and port C general outputs. Four pins (PC12 to PC15) of the port C are used for pulse outputs. The port C outputs are changed at each compare-match interrupt generated by the CMT. Figure 4 (1) shows the functional block diagram, and figure 4 (2) shows the pulses for 1-2 phase excitation (output changes on port C).

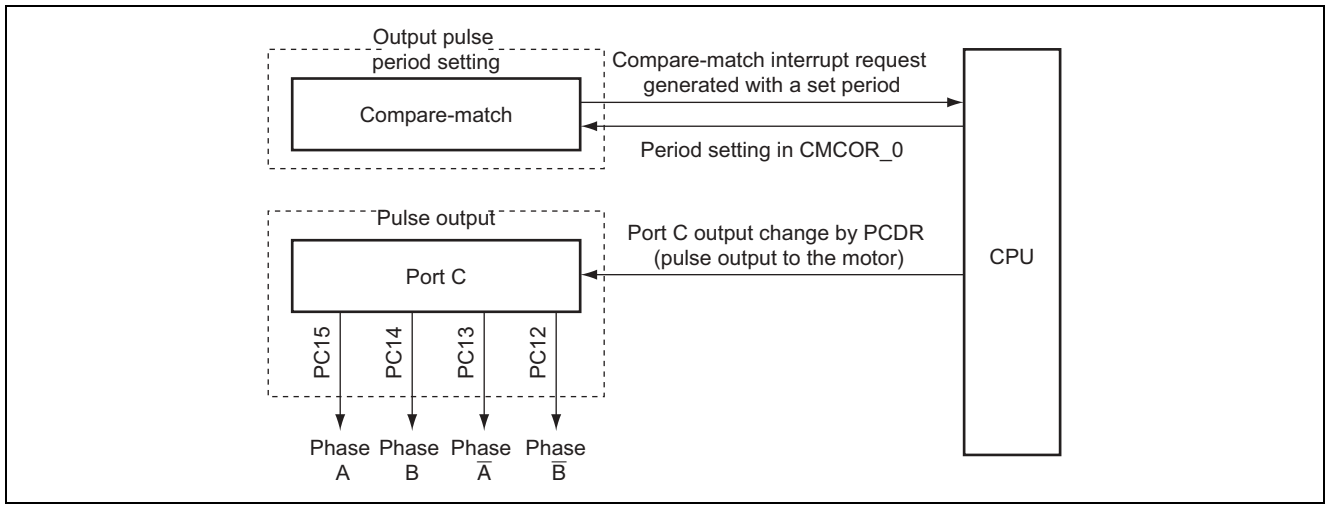


Figure 4 Functional Block Diagram (1)

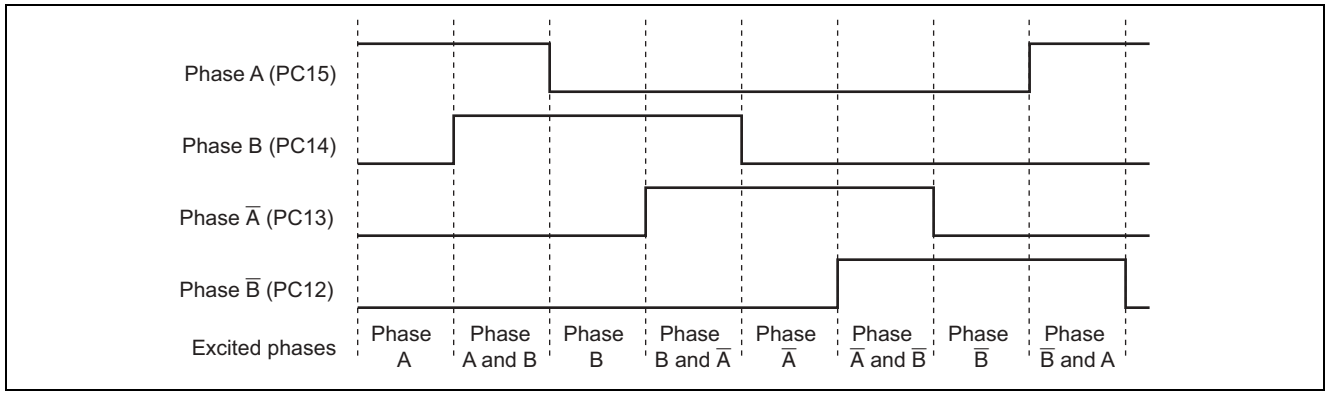


Figure 4 1-2 Phase Excitation Pulses (2)

3.2 Example of Stepping Motor Operation

Figure 5 shows an example of stepping motor operation by 1-2 phase excitation. The operation is summarized below.

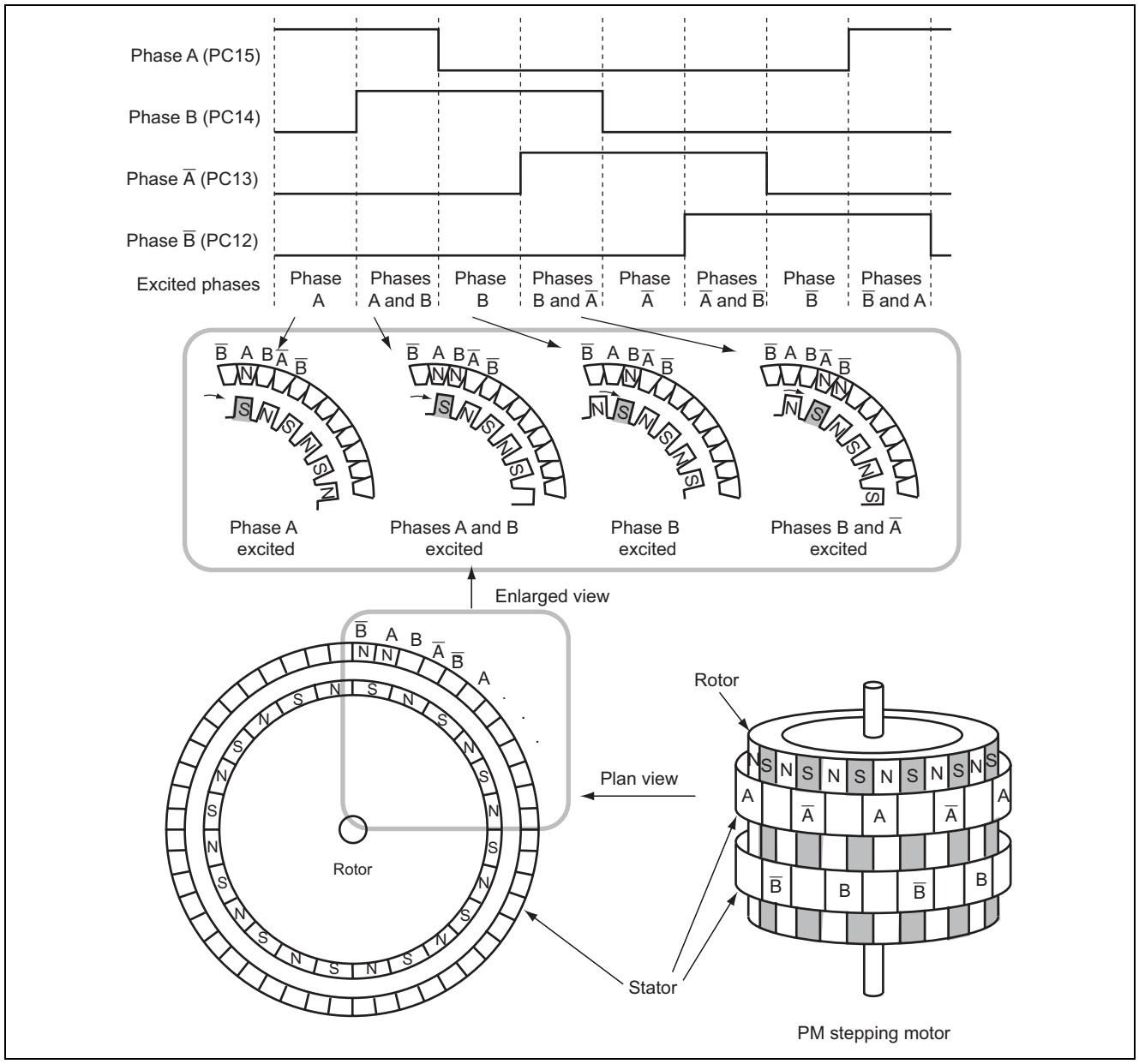


Figure 5 Stepping Motor Operation by 1-2 Phase Excitation

1. Firstly, phase A of the stator is excited, and the magnets on the rotor are positioned at phase A.
2. Next, phase A and phase B are excited simultaneously. The magnets on the rotor are then in the intermediate position of phase A and phase B. Subsequently, the phases are excited in the following sequence to cause the rotor to rotate: phase B → phases B and Ā → phase Ā → phases Ā and B̄.
3. Reverse rotation of the stepping motor is achieved by exciting phases in the following sequence: phases B and A → phase A → phases A and B̄ → phase B̄ → phases Ā and B̄.
4. The stepping motor is stopped by holding the excitation of the last phase for a specific period.

3.3 Motor Acceleration and Deceleration Control

The motor acceleration and deceleration is controlled by slew-up and slew-down operations. This prevents the motor from going out of synchnonization with the control pulses when the motor rotation is started or stopped. Figure 6 shows an example of stepping motor operation, which is explained below.

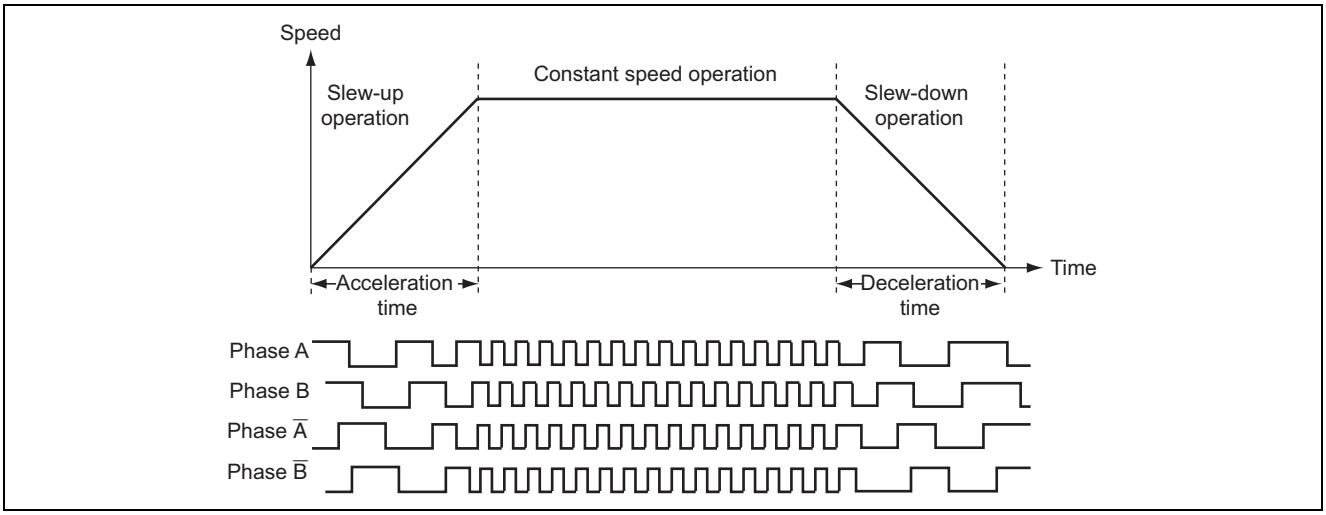


Figure 6 Slew-Up and Slew-Down Operations

1. The pulse period is gradually shortened to increase the motor rotation speed (slew-up).
2. The motor rotation speed is held constant by generating pulses of a constant period.
3. The pulse period is gradually extended to decrease the motor rotation speed (slew-down.)

Figure 7 shows the flow of acceleration and deceleration control for the stepping motor.

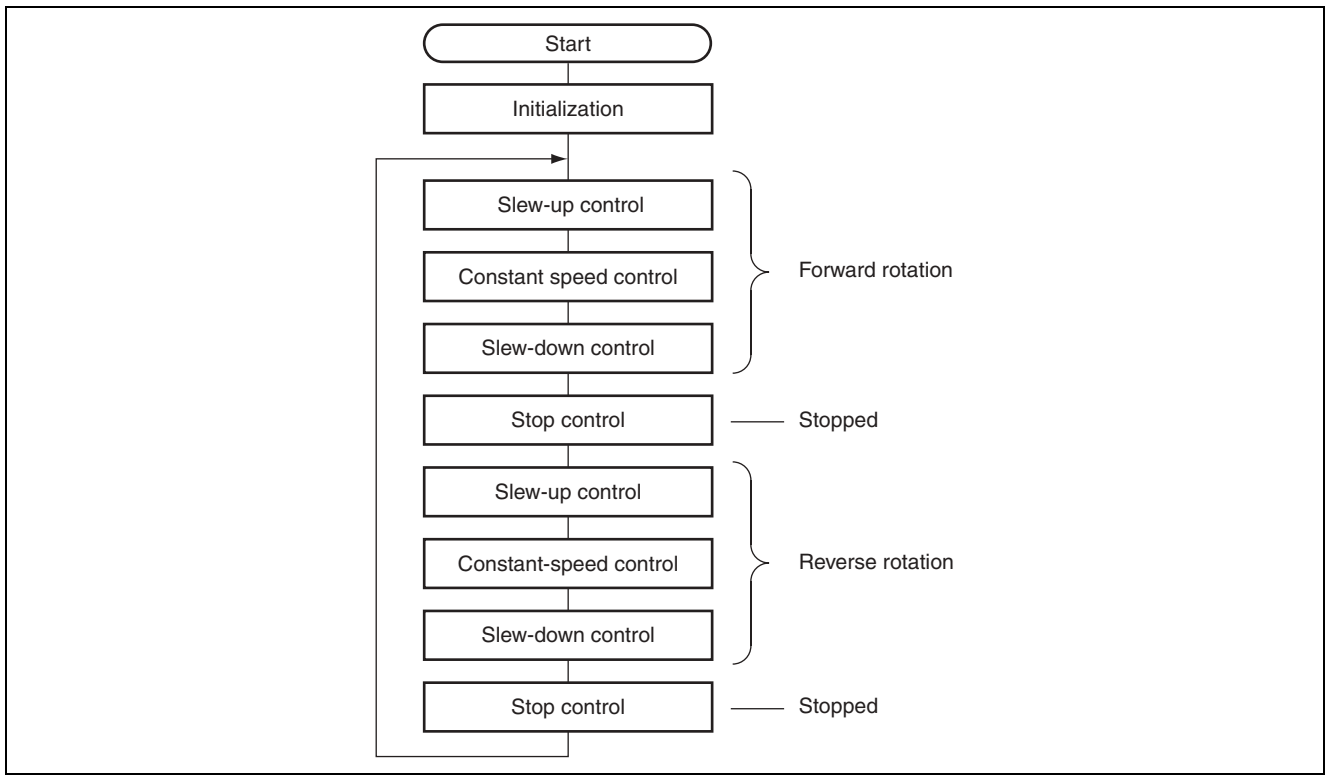


Figure 7 Flowchart of Acceleration and Deceleration Control

3.4 Slew-Up Control

Figure 8 shows the operation of slew-up control. Table 2 summarizes the software and hardware processing during slew-up control. Note that figure 8 shows the slew-up operation in forward rotation. In reverse rotation, the high-level outputs from port C are generated in the reverse sequence.

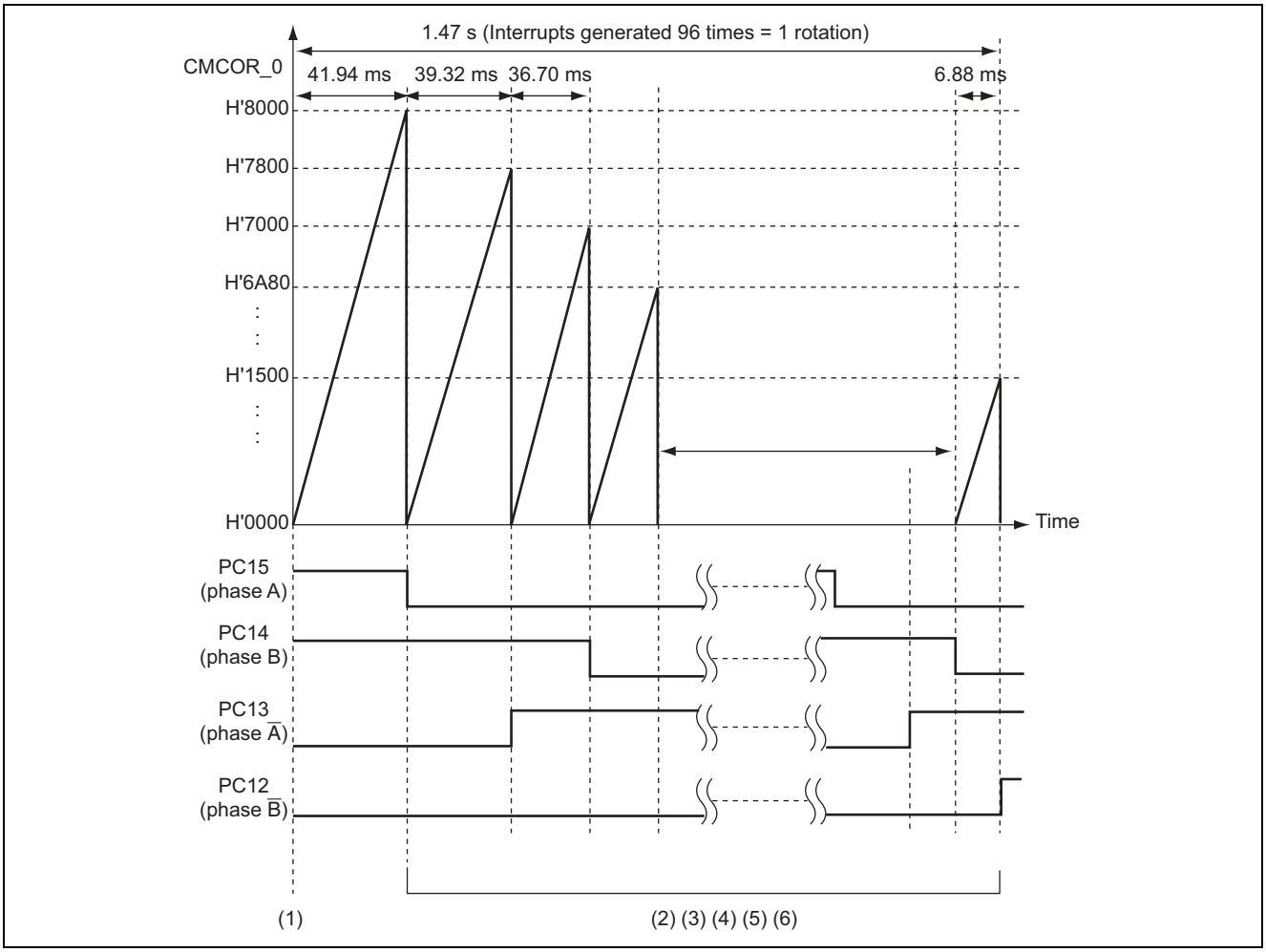


Figure 8 Operation in Slew-Up Control

Table 2 Processing in Slew-Up Control

No.	Software Processing	Hardware Processing
(1)	—	Start CMT0 counting.
(2)	—	Set the CMF flag (generate a compare-match interrupt).
(3)	Set the next excitation switching period in CMCOR_0.	—
(4)	Clear the CMF flag.	Start CMT0 counting.
(5)	Set the PCDR to switch the phase to be excited.	—
(6)	Repeat steps (2) to (5) above until the rotor rotates once.	Repeat steps (2) to (5) above until the rotor rotates once.

3.5 Slew-Down Control

Figure 9 shows the operation of slew-down control. Table 3 summarizes the software and hardware processing during slew-down control. Note that figure 9 shows the slew-down operation in forward rotation. In reverse rotation, the high-level outputs from port C are generated in the reverse sequence.

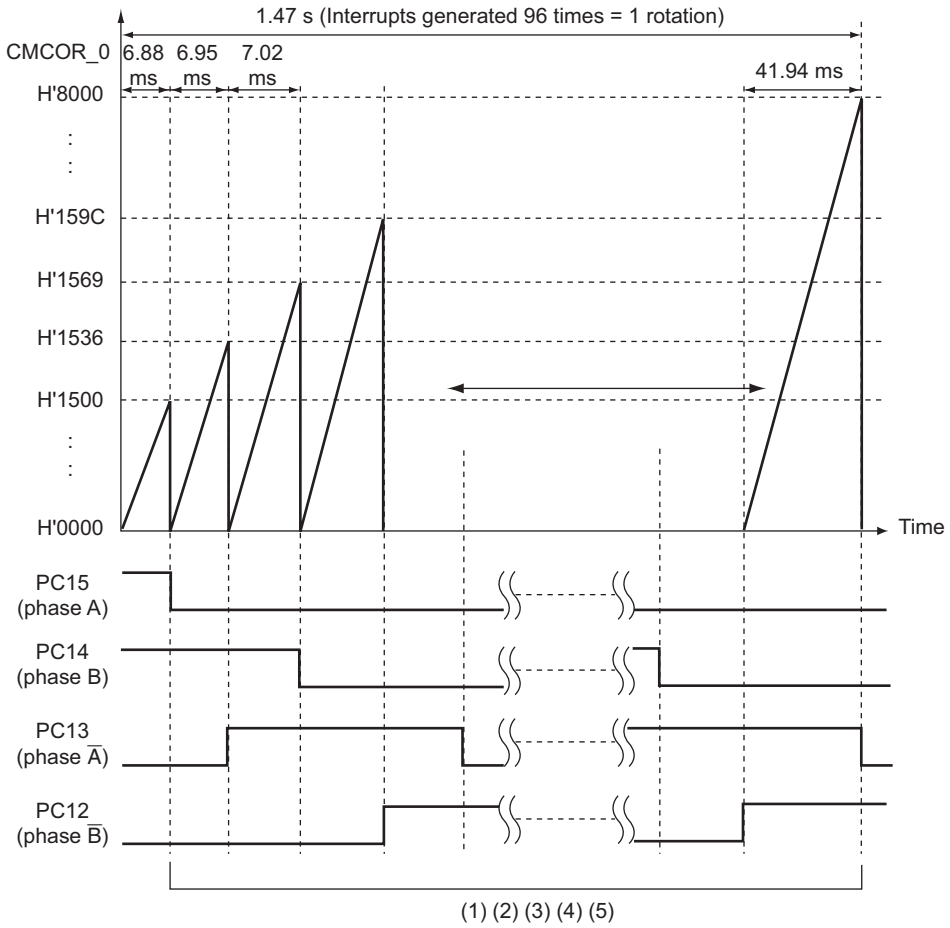


Figure 9 Operation in Slew-Down Control

Table 3 Processing in Slew-Down Control

No.	Software Processing	Hardware Processing
(1)	—	Set the CMF flag (generate a compare-match interrupt).
(2)	Set the next excitation switching period in CMCOR_0.	—
(3)	Clear the CMF flag.	Start CMT0 counting.
(4)	Set the PCDR to switch the phase to be excited.	—
(5)	Repeat steps (1) to (4) above until the rotor rotates once.	Repeat steps (1) to (4) above until the rotor rotates once.

3.6 Constant-Speed Control

Figure 10 shows the operation of constant-speed control. Table 4 summarizes the software and hardware processing during constant-speed control. Note that figure 10 shows the constant-speed operation in forward rotation. In reverse rotation, the high-level outputs from port C are generated in the reverse sequence.

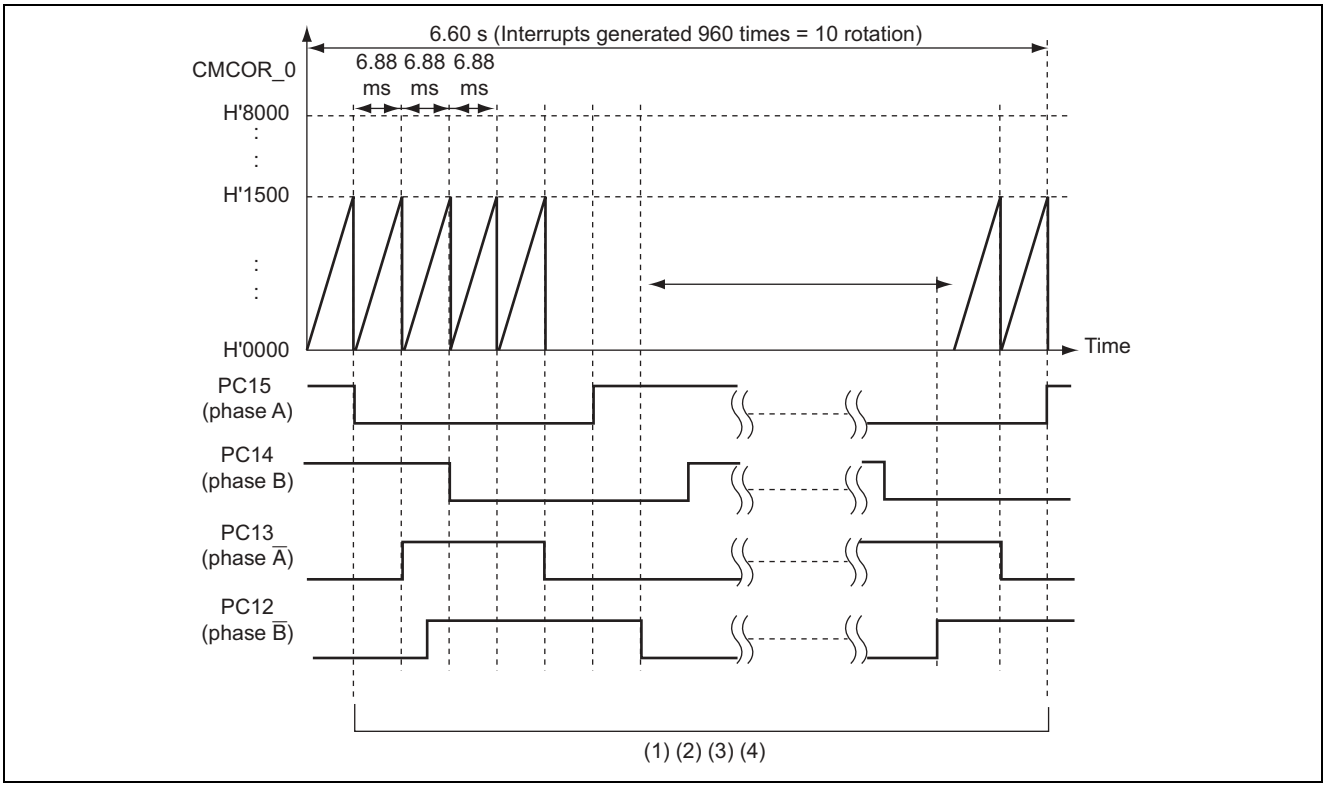


Figure 10 Operation in Constant-Speed Control

Table 4 Processing in Constant-Speed Control

No.	Software Processing	Hardware Processing
(1)	—	Set the CMF flag (generate a compare-match interrupt).
(2)	Clear the CMF flag.	Start CMT0 counting.
(3)	Set the PCDR to switch the phase to be excited.	—
(4)	Repeat steps (1) to (3) above a specified number of times.	Repeat steps (1) to (3) above a specified number of times.

4. Description of Software

4.1 Modules

Table 5 shows the modules used in this sample task.

Table 5 Description of Modules

Module Name	Label Name	Function
Main routine	main	Initializes the global variables, I/O ports, and compare-match timer and enables the interrupt.
Compare-match interrupt processing	cmt_int	Main routine of stepping motor control.
Slew-up control in forward rotation	fslueup	Performs slew-up control in forward rotation.
Slew-down control in forward rotation	fsluedwn	Performs slew-down control in forward rotation.
Constant-speed control in forward rotation	fconst	Performs constant-speed control in forward rotation.
Rotation stop	frstop	Stops forward or reverse rotation.
Slew-up control in reverse rotation	rslueup	Performs slew-up control in reverse rotation.
Slew-down control in reverse rotation	rsluedwn	Performs slew-down control in reverse rotation.
Constant-speed control in reverse rotation	rconst	Performs constant-speed control in reverse rotation.

4.2 Internal Registers

Tables 6 and 7 describe the internal registers used in this sample task. Note that the setting values in these tables are the values used in this sample task and not the initial values.

Table 6 Description of Internal Registers (1)

Register Name	Bit	Bit Name	Setting	Function
MSTCR2				Module standby control register 2
	12	MSTP12	0	CMT standby control bit When MSTP12 = 0, standby state of the CMT is cancelled.
CMSTR			H'01	Compare-match timer start register
	15 to 2	—	0	Reserved bits
	1	STR1	0	Count start 1 When STR1 = 0, CMCNT_1 stops counting.
	0	STR0	1	Count start 0 When STR0 = 1, CMCNT_0 starts counting.
CMCSR_0				Compare-match timer control/status register_0
	15 to 8	—	0	Reserved its
	7	CMF	*1	Compare-match flag When CMF = 1, CMCNT matches CMCOR.
	6	CMIE	1	Compare-match interrupt enable Enables or disables the compare-match interrupt. When CMIE = 1, the compare-match interrupt is enabled.
	5 to 2	—	0	Reserved its
	1, 0	CKS1 CKS0	0 1	CMCNT_0 input clock selection In this sample task, ϕP/32 is selected.
CMCNT_0			—	Compare-match timer counter_0 Up-counter used to generate interrupt requests.
CMCOR_0			*2	Compare-match timer constant register_0 This register is used to set the period of compare-match with CMCNT_0.

Notes: 1. This bit can only be cleared to 0 and is automatically set to 1 by hardware.
2. The value changes during slew-up or slew-down control.

Table 7 Description of Internal Registers (2)

Register Name	Bit	Bit Name	Setting	Function
IPRG			H'00F0	Interrupt priority register G Specifies the priority of interrupt sources.
	7 to 4	IPR7	1	These bits specify the priority level of CMT0 (0 to 15).
		IPR6	1	
		IPR5	1	
		IPR4	1	
PCCR			H'0000	Port C control register Specifies port C pin functions.
	15	PC15MD	0	When PC15MD = 0, the corresponding pin functions as a general port.
	14	PC14MD	0	When PC14MD = 0, the corresponding pin functions as a general port.
	13	PC13MD	0	When PC13MD = 0, the corresponding pin functions as a general port.
	12	PC12MD	0	When PC12MD = 0, the corresponding pin functions as a general port.
PCIOR			H'0F00	Port C I/O register Specifies input or output for the port C pins.
	15	PC15IOR	1	When PC15IOR = 1, PC15 functions as an output pin.
	14	PC14IOR	1	When PC14IOR = 1, PC14 functions as an output pin.
	13	PC13IOR	1	When PC13IOR = 1, PC13 functions as an output pin.
	12	PC12IOR	1	When PC12IOR = 1, PC12 functions as an output pin.
PCDR				Port C data register
	15	PC15DR	*	When PC15 function is general output, the value of PC15DR is output.
	14	PC14DR	*	When PC14 function is general output, the value of PC14DR is output.
	13	PC13DR	*	When PC13 function is general output, the value of PC13DR is output.
	12	PC12DR	*	When PC12 function is general output, the value of PC12DR is output.

Note: The setting values are changed everytime a compare-match interrupt occurs.

4.3 RAM Usage

Table 8 describes the RAM usage in this sample task.

Table 8 Description of RAM

Label Name	Function	Memory Size	Used in
ppcnt	Index of array pattb[], which stores excitation data of the stepping motor	1 byte	Main routine Compare-match interrupt processing Slew-up control in forward rotation Slew-down control in forward rotation Constant-speed control in forward rotation Rotation stop Slew-up control in reverse rotation Slew-down control in reverse rotation Constant-speed control in reverse rotation
sluecnt	Index of array int_cyc[], which is used in slew-up and slew-down processing	1 byte	Main routine Compare-match interrupt processing Slew-up control in forward rotation Slew-down control in forward rotation Slew-up control in reverse rotation Slew-down control in reverse rotation
nextmode	Specifies an operation mode of the stepping motor.	1 byte	Main routine Compare-match interrupt processing
modecnt	Specifies the number of interrupts for each operation mode of the stepping motor.	2 bytes	Main routine Compare-match interrupt processing
pattb[8]	Stepping motor excitation pattern data table	8 bytes	Main routine Slew-up control in forward rotation Slew-down control in forward rotation Constant-speed control in forward rotation Rotation stop Slew-up control in reverse rotation Slew-down control in reverse rotation Constant-speed control in reverse rotation
int_cyc[96]	Interrupt period data table used in slew-up and slew-down processing	8 bytes	Main routine Slew-up control in forward rotation Slew-down control in forward rotation Slew-up control in reverse rotation Slew-down control in reverse rotation

4.4 Data Table Variables

- Data table for switching the stepping motor excitation pattern

```
pattb[8] = {
  0x8000,          : Excites phase A (PC15).
  0xC000,          : Excites phases A (PC15) and B (PC14).
  0x4000,          : Excites phase B (PC14).
  0x6000,          : Excites phases B (PC14) and Ā (PC13).
  0x2000,          : Excites phase Ā (PC13).
  0x3000,          : Excites phases Ā (PC13) and B̄ (PC12).
  0x1000,          : Excites phase B̄ (PC12).
  0x9000,          : Excites phases B̄ (PC12) and A (PC15).
}
```

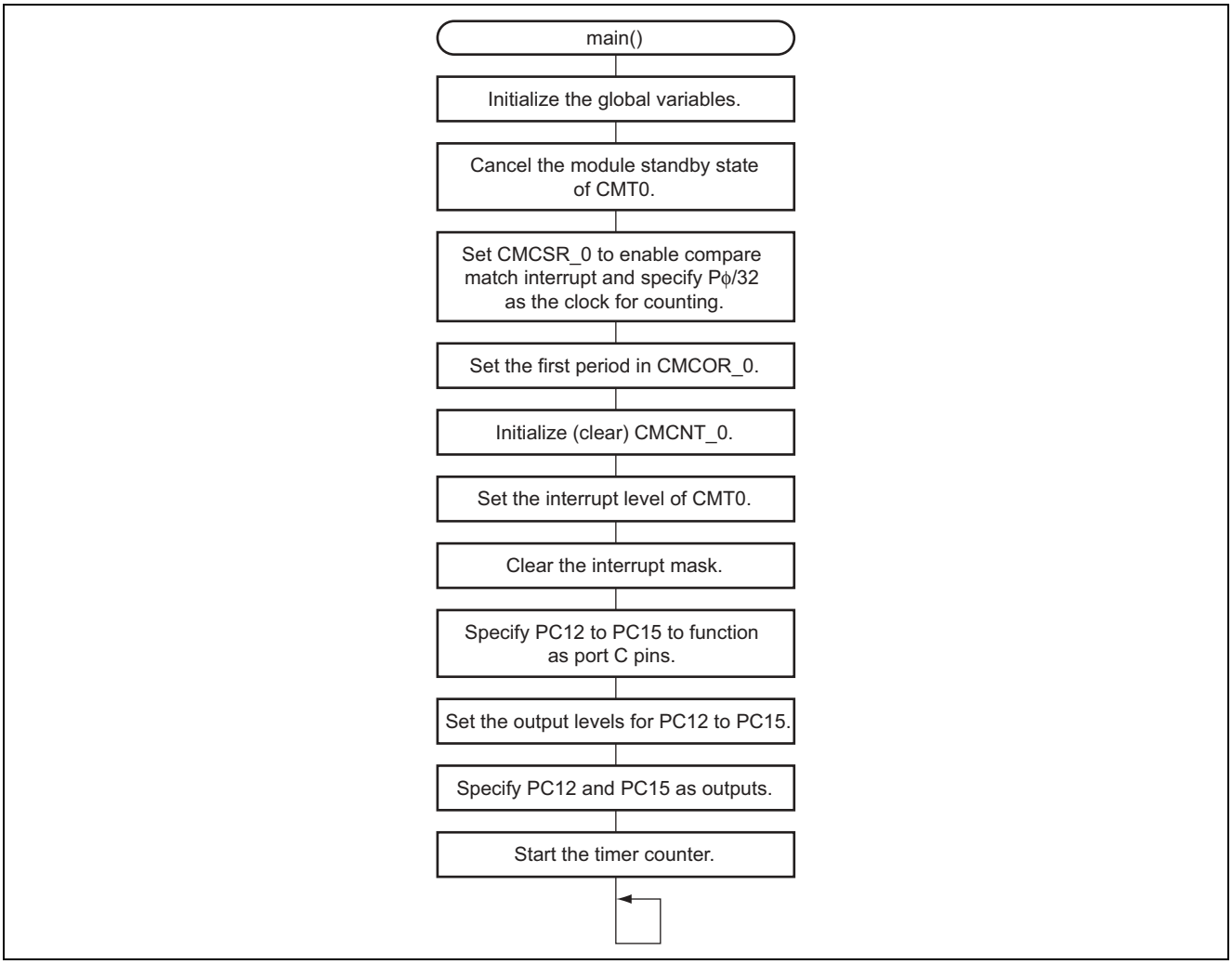
- Data table for period setting in slew-up and slew-down processing

```
int_cyc[96] = {
  0x8000,0x7800,0x7000,0x6A80,0x6720,0x639C,0x6018,0x5C94,0x5910,0x558C,
  0x5208,0x4EE8,0x4BC8,0x48A8,0x4650,0x43F8,0x41A0,0x4010,0x3E80,0x3D54,
  0x3C28,0x3AFC,0x3A5E,0x38A4,0x37DC,0x3714,0x364C,0x3584,0x34BC,0x33F4,
  0x335E,0x32C8,0x3232,0x319C,0x3106,0x30A2,0x303E,0x2FDA,0x2F76,0x2EE5,
  0x2E54,0x2DC3,0x2D32,0x2CA1,0x2C10,0x2B7F,0x2AEE,0x2A5D,0x29CC,0x293B,
  0x28AA,0x2819,0x2788,0x26F7,0x2666,0x25D5,0x2544,0x24B3,0x2422,0x2391,
  0x2300,0x226F,0x21DE,0x214D,0x20BC,0x202B,0x1F9A,0x1F09,0x1E78,0x1DE2,
  0x1D1A,0x1C48,0x1BCE,0x1AF4,0x1A4A,0x19EB,0x198C,0x192D,0x18CE,0x186F,
  0x1823,0x17F2,0x17B8,0x177B,0x173E,0x1701,0x16CE,0x169B,0x1668,0x1635,
  0x1602,0x15CF,0x159C,0x1569,0x1536,0x1500,
}
```

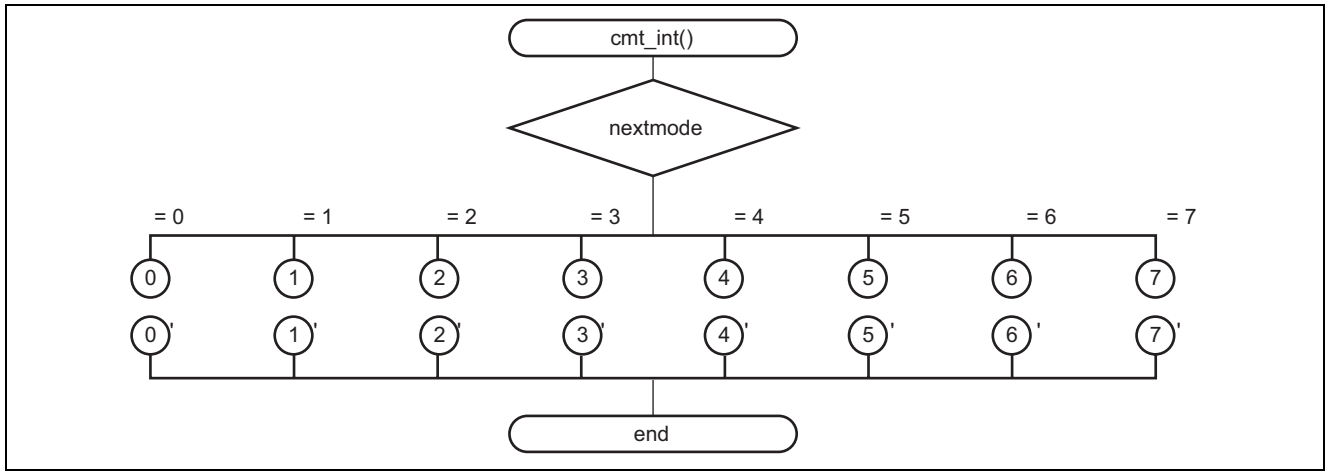
During slew-up and slew-down operations, data from this table is written to CMCOR_0 everytime a compare-match interrupt occurs to perform acceleration/deceleration control for the stepping motor.

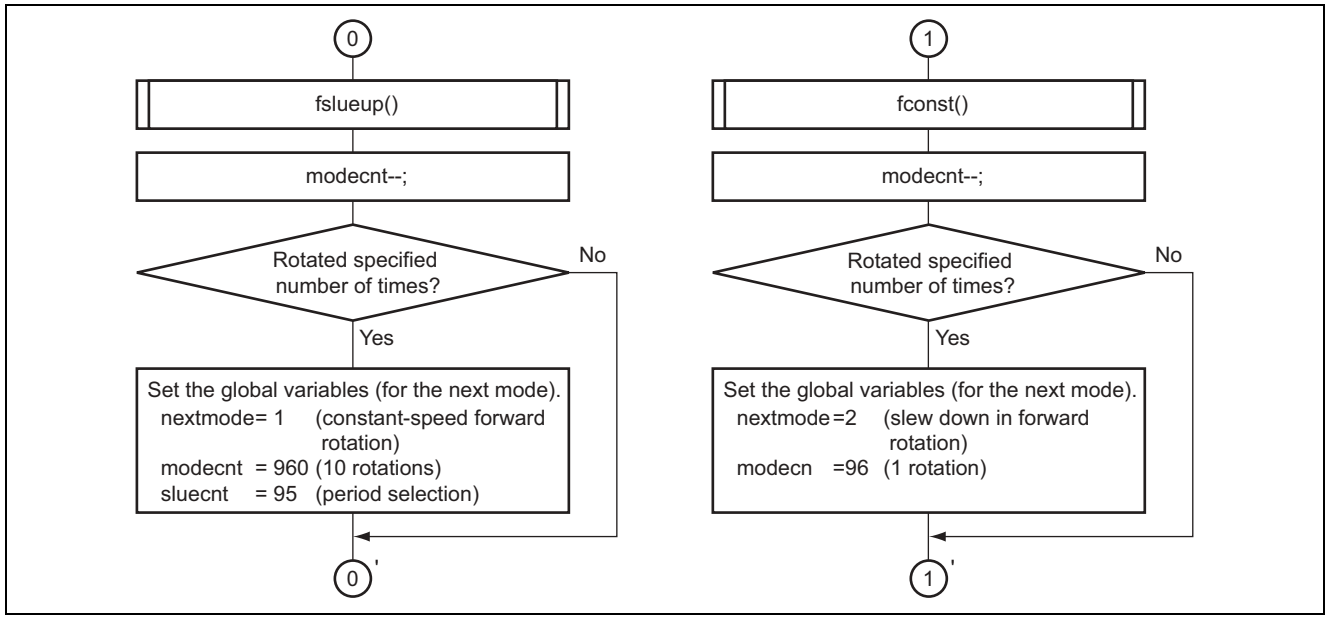
5. Flowchart

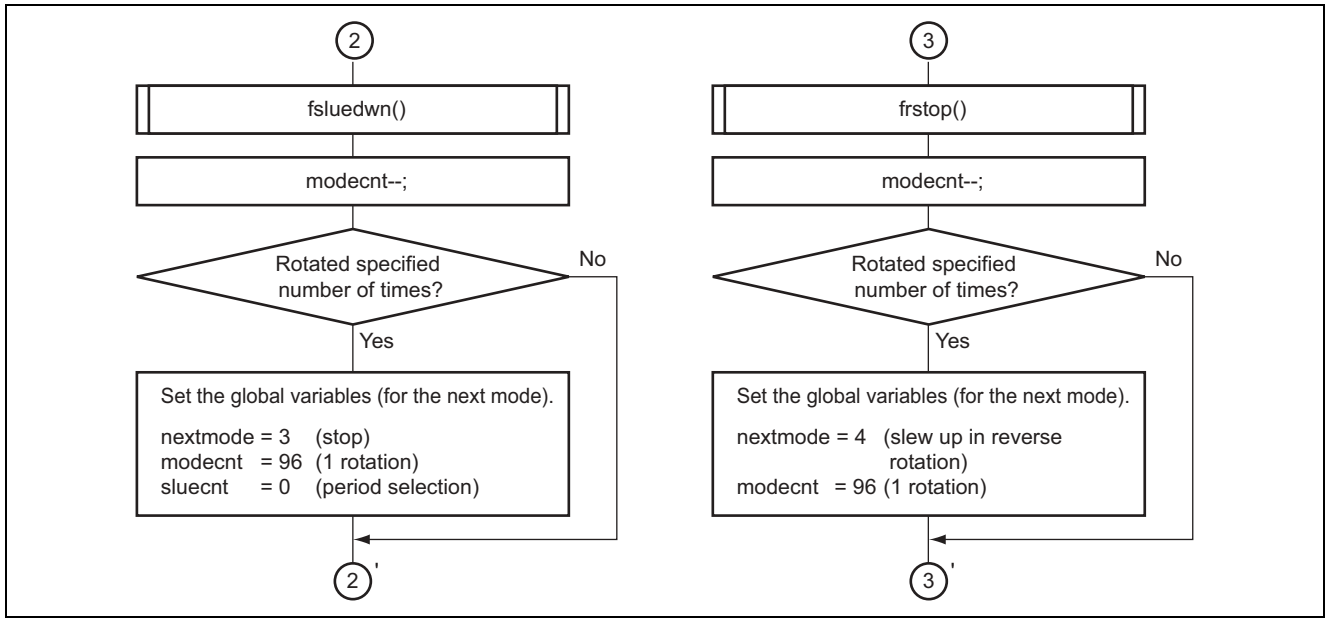
5.1 Main Routine

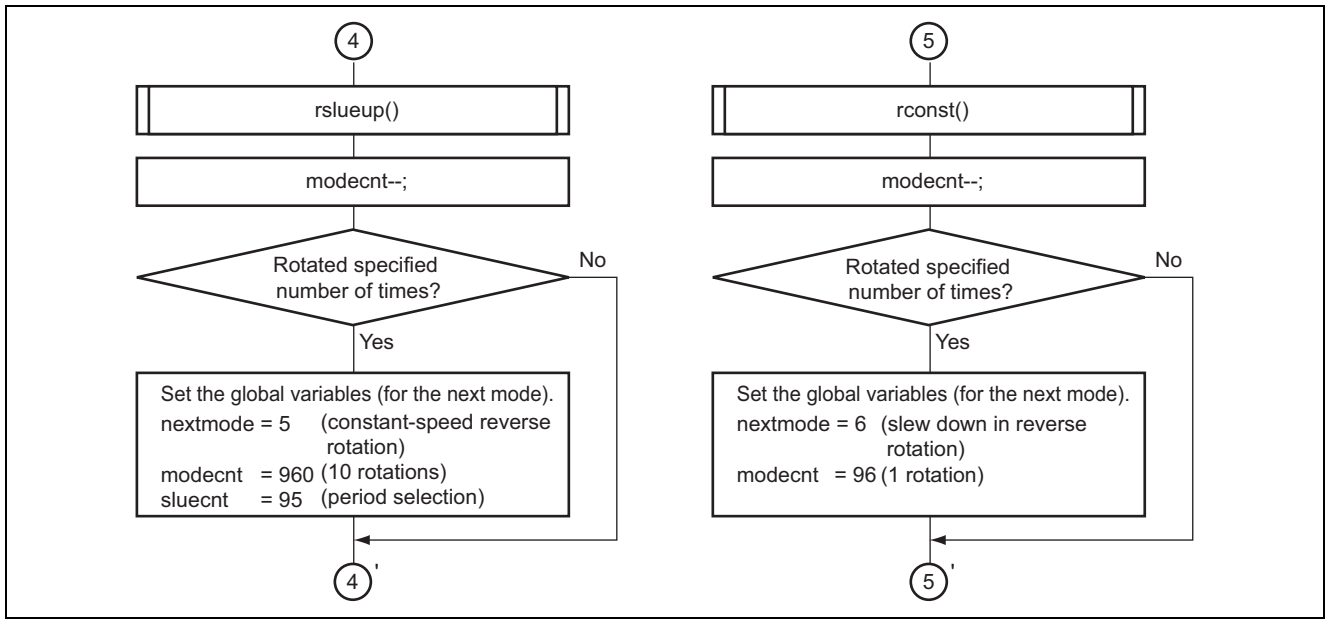


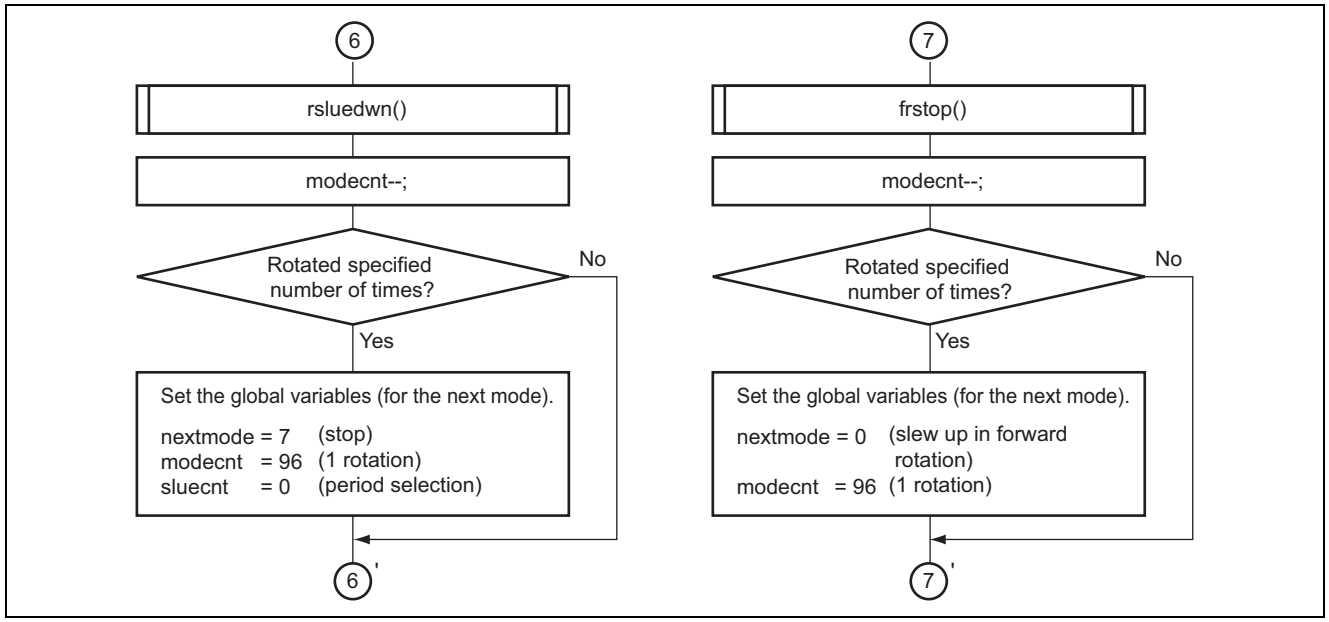
5.2 Compare-Match Interrupt Routine



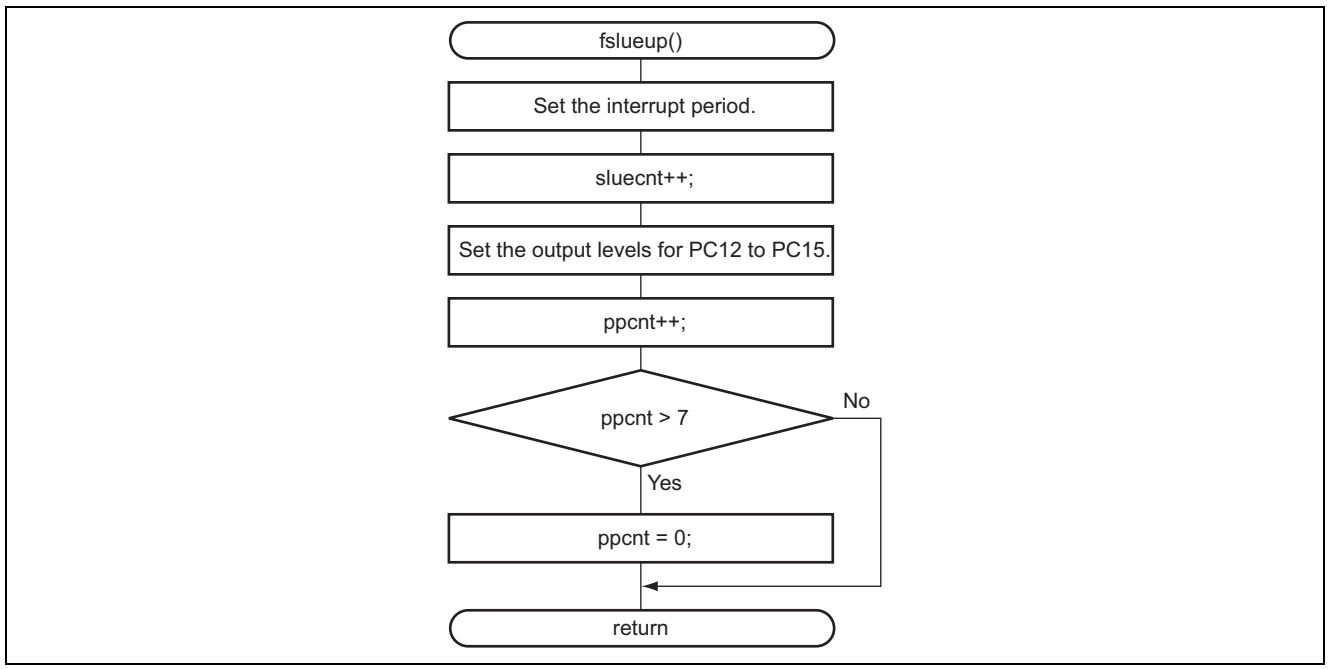



2
fsluedwn()
modecnt--;
Rotated specified number of times?
No
Yes
Set the global variables (for the next mode).
nextmode = 3 (stop)
modecnt = 96 (1 rotation)
sluecnt = 0 (period selection)
2'
3
frstop()
modecnt--;
Rotated specified number of times?
No
Yes
Set the global variables (for the next mode).
nextmode = 4 (slew up in reverse rotation)
modecnt = 96 (1 rotation)
3'

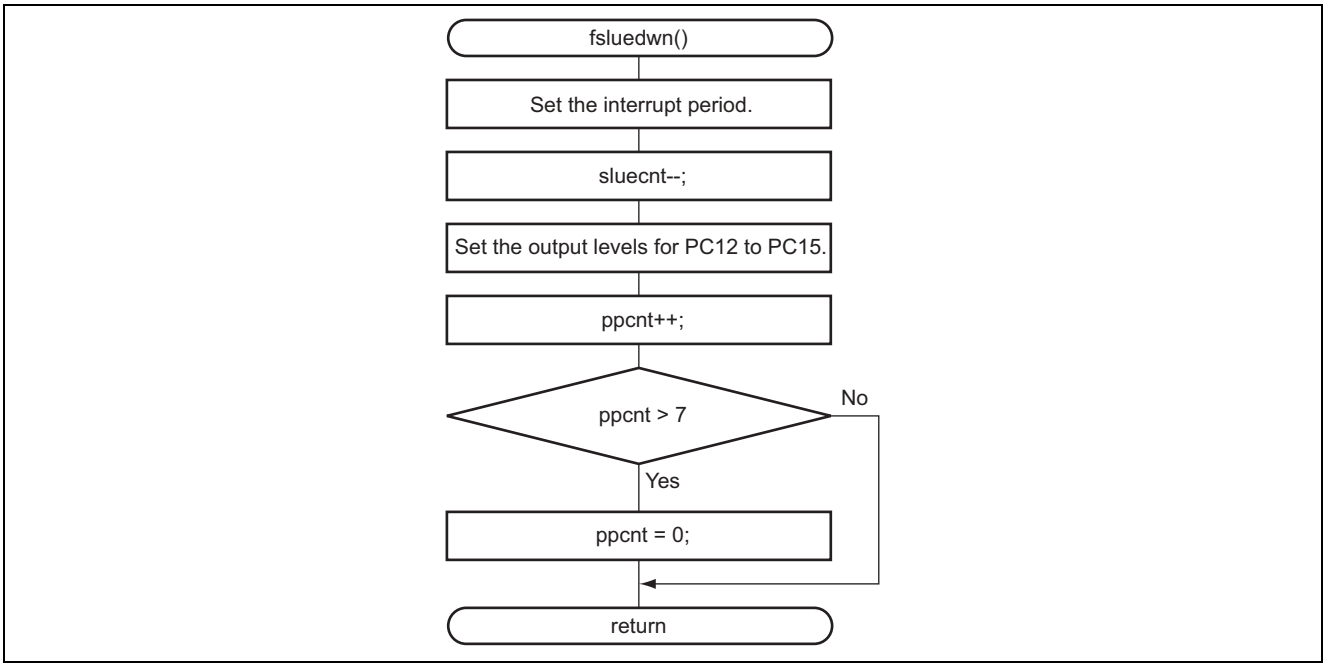

4
rslueup()
modecnt--;
Rotated specified number of times?
No
Yes
Set the global variables (for the next mode).
nextmode = 5 (constant-speed reverse rotation)
modecnt = 960 (10 rotations)
sluecnt = 95 (period selection)
4'
5
rconst()
modecnt--;
Rotated specified number of times?
No
Yes
Set the global variables (for the next mode).
nextmode = 6 (slew down in reverse rotation)
modecnt = 96 (1 rotation)
5'



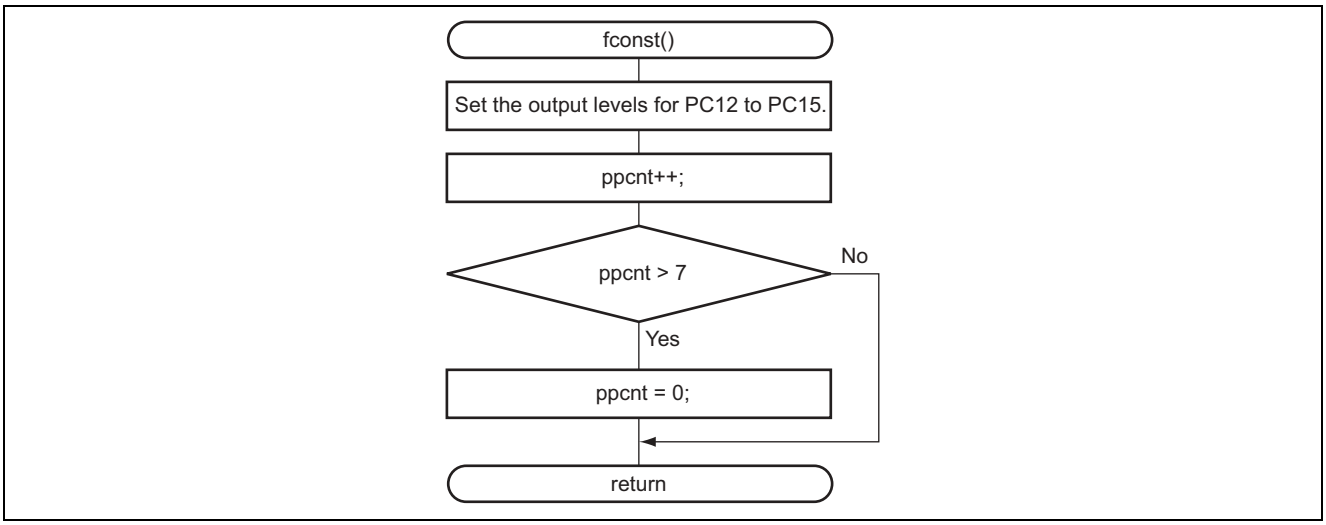
5.3 Slew-Up Routine for Forward Rotation



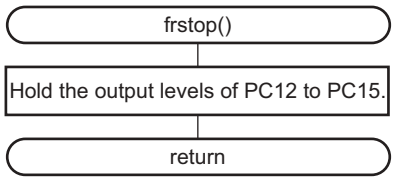
5.4 Slew-Down Routine for Forward Rotation



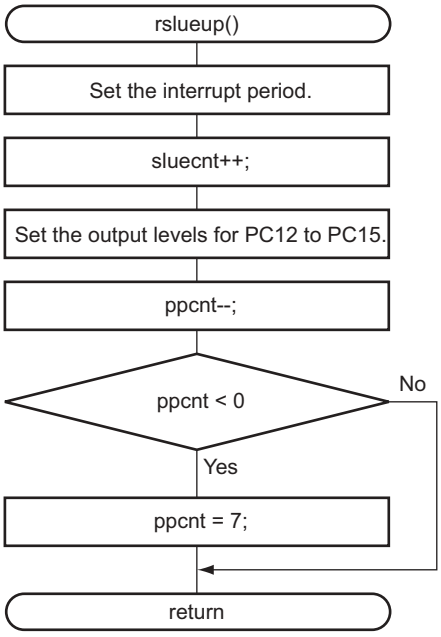
5.5 Routine for Constant-Speed Forward Rotation



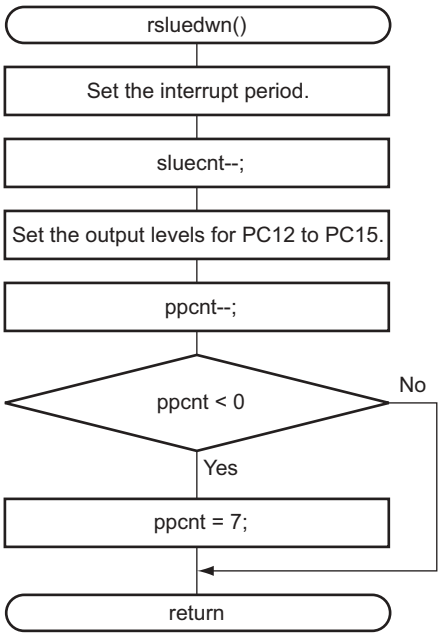
5.6 Stop Routine



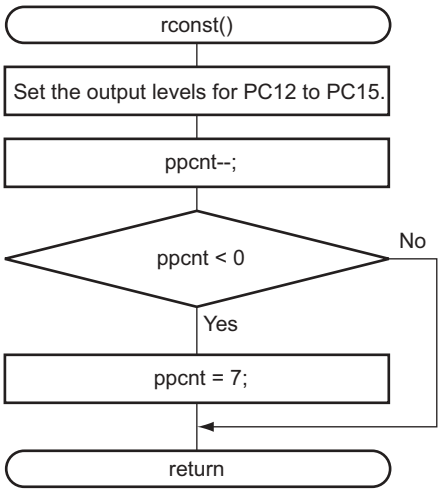
5.7 Slew-Up Routine for Reverse Rotation



5.8 Slew-Down Routine for Reverse Rotation



5.9 Routine for Constant-Speed Reverse Rotation



6. Program Listing

```
/************************************************************************************************/
/* SH7145F Application Note                                                                     */
/*                                                                                              */
/* Function                                                                                     */
/*    :CMT0(Stepping motor)                                                                     */
/*                                                                                              */
/* External input clock         :12.5MHz                                                        */
/* Internal CPU clock           :50MHz                                                          */
/* Internal peripheral clock    :25MHz                                                          */
/*                                                                                              */
/* Written        :2003/10      Rev.1.0                                                         */
/************************************************************************************************/

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

/************************************************************************************************/
/*  Function Define                                                                             */
/************************************************************************************************/
void main(void);
extern void _INITSCT(void);

void fslueup(void);
void fsluedwn(void);
void fconst(void);
void frstop(void);
void rslueup(void);
void rsluedwn(void);
void rconst(void);

void cmt_int(void);

void dummy_f(void);
/************************************************************************************************/
/*  Global Variable                                                                             */
/************************************************************************************************/
char ppcnt;
unsigned char sluecnt;
unsigned char nextmode;
long modecnt;
```

```
/*****************************************************************************************************/
/*  Stepping Motor Output Pattern Table                                                              */
/*****************************************************************************************************/

unsigned short pattb[8] = {                         /* Output Pulse Pattern Table               */
    0x8000,0xC000,0x4000,0x6000,0x2000,0x3000,0x1000,0x9000
};

unsigned short int_cyc[96] = {                      /* Interrupt Cycle Table                    */
    0x8000,0x7800,0x7000,0x6A80,0x6720,0x639C,0x6018,0x5C94,0x5910,0x558C,
    0x5208,0x4EE8,0x4BC8,0x48A8,0x4650,0x43F8,0x41A0,0x4010,0x3E80,0x3D54,
    0x3C28,0x3AFC,0x3A5E,0x38A4,0x37DC,0x3714,0x364C,0x3584,0x34BC,0x33F4,
    0x335E,0x32C8,0x3232,0x319C,0x3106,0x30A2,0x303E,0x2FDA,0x2F76,0x2EE5,
    0x2E54,0x2DC3,0x2D32,0x2CA1,0x2C10,0x2B7F,0x2AEE,0x2A5D,0x29CC,0x293B,
    0x28AA,0x2819,0x2788,0x26F7,0x2666,0x25D5,0x2544,0x24B3,0x2422,0x2391,
    0x2300,0x226F,0x21DE,0x214D,0x20BC,0x202B,0x1F9A,0x1F09,0x1E78,0x1DE2,
    0x1D1A,0x1C48,0x1BCE,0x1AF4,0x1A4A,0x19EB,0x198C,0x192D,0x18CE,0x186F,
    0x1823,0x17F2,0x17B8,0x177B,0x173E,0x1701,0x16CE,0x169B,0x1668,0x1635,
    0x1602,0x15CF,0x159C,0x1569,0x1536,0x1500,
};
/*****************************************************************************************************/
/*  Main Program                                                                                     */
/*****************************************************************************************************/
void main(void)
{
    ppcnt = 0;                                      /* Output Pulse Pattern table counter set   */
    sluecnt = 0;                                    /* Slow Up/Down table counter set           */
    nextmode = 0;
    modecnt = 95;                                   /* Motor Slue mode count set "95"           */

    P_STBY.MSTCR2.BIT.MSTP12 = 0;                   /* Disable CMT0 standby mode                */

    P_CMT.CMCSR_0.WORD = 0x0041;                    /* Set CMCSR_0                              */
             // [15-8] = 0;Reserve
             // [7]    = 0;CMF
             // [6]    = 1;CMT interrupt enable
             // [5-2]  = 0;Reserve
             // [1]    = 0:
             // [0]    = 1;count clock=P phi/32
    P_CMT.CMCOR_0 = int_cyc[sluecnt];               /* Set interrupt cycle table                */
    sluecnt++;

    P_CMT.CMCNT_0 = 0;                              /* Clear CMCNT_0                            */

    P_INTC.IPRG.BIT.CMT0 = 0xF;                     /* Set CMT0 interrupt level                 */
    set_imask(0);                                   /* Set interrupt mask level                 */

    P_PORTC.PCCR.WORD = 0;                          /* Set function                             */

    P_PORTC.PCDR.WORD = pattb[ppcnt];               /* Set portC output pattern table           */
    ppcnt++;

    P_PORTC.PCIOR.WORD |= 0xF000;                   /* PortC output                             */
```

```
    P_CMT.CMSTR.BIT.STR = 1;                    /* Timer count start                    */

    while(1);                                   /* LOOP                                 */
}
/****************************************************************************************/
/*  Interrupt Processing                                                                */
/****************************************************************************************/
#pragma interrupt(cmt_int)
void cmt_int(void)
{
    P_CMT.CMCSR_0.BIT.CMF = 0;                  /* Clear CMF                            */

switch(nextmode){
    case 0:
        fslueup();                              /* Forward Slow Up                      */
        modecnt--;
        if(modecnt <= 0){                       /* Next mode?                           */
            nextmode = 1;                       /* nextmode = 1 Constant Speed          */
            modecnt = 960;                      /* Next mode count set "960"            */
            sluecnt = 95;                       /* Slow Up/Down table counter set       */
        }
        break;

    case 1:
        fconst();                               /* Constant Speed                       */
        modecnt--;
        if(modecnt <= 0){                       /* Next mode?                           */
            nextmode = 2;                       /* nextmode = 2 Forward Slue Down       */
            modecnt = 96;                       /* Next mode count set "96"             */
        }
        break;

    case 2:
        fsluedwn();                             /* Forward Slow Down                    */
        modecnt--;
        if(modecnt <= 0){                       /* Next mode?                           */
            nextmode = 3;                       /* nextmode = 3 Slue Stop               */
            modecnt = 96;                       /* Next mode count set "96"             */
            sluecnt = 0;                        /* Slow Up/Down table counter set       */
        }
        break;

    case 3:
        frstop();                               /* Rotation Stop                        */
        modecnt--;
        if(modecnt <= 0){                       /* Next mode?                           */
            nextmode = 4;                       /* nextmode = 4 Reverse Slow Up         */
            modecnt = 96;                       /* Next mode count set "96"             */
        }
        break;
```

```
    case 4:
        rslueup();                              /* Reverse Slow Up                  */
        modecnt--;
        if(modecnt <= 0){                       /* Next mode?                       */
            nextmode = 5;                       /* nextmode = 5 Constant Speed      */
            modecnt = 960;                      /* Next mode count set "960"        */
            sluecnt = 95;                       /* Slow Up/Down table counter set   */
        }
        break;

    case 5:
        rconst();                               /* Constant Speed                   */
        modecnt--;
        if(modecnt <= 0){                       /* Next mode?                       */
            nextmode = 6;                       /* nextmode = 6 Reverse Slow Down   */
            modecnt = 96;                       /* Next mode count set "96"         */
        }
        break;

    case 6:
        rsluedwn();                             /* Reverse Slow Down                */
        modecnt--;
        if(modecnt <= 0){                       /* Next mode?                       */
            nextmode = 7;                       /* nextmode = 7 Slue Stop           */
            modecnt = 96;                       /* Next mode count set "96"         */
            sluecnt = 0;                        /* Slow Up/Down table counter set   */
        }
        break;

    case 7:
        frstop();                               /* Slue Stop                        */
        modecnt--;
        if(modecnt <= 0){                       /* Next mode?                       */
            nextmode = 0;                       /* nextmode = 0 Forward Slow Up     */
            modecnt = 96;                       /* Next mode count set "96"         */
        }
        break;
    }
}
/*****************************************************************************************/
/*  Forward Slow Up                                                                      */
/*****************************************************************************************/
void fslueup(void)
{
    P_CMT.CMCOR_0 = int_cyc[sluecnt];           /* Set interrupt cycle table        */
    sluecnt++;

    P_PORTC.PCDR.WORD = pattb[ppcnt];           /* Set portC output pattern table   */
    ppcnt++;

    if(ppcnt>7)
        ppcnt = 0;
}
```

```
/*****************************************************************************************************/
/*  Forward Slow Down                                                                                */
/*****************************************************************************************************/
void fsluedwn(void)
{
    P_CMT.CMCOR_0 = int_cyc[sluecnt];              /* Set interrupt cycle table                */
    sluecnt--;

    P_PORTC.PCDR.WORD = pattb[ppcnt];              /* Set portC output pattern table           */
    ppcnt++;

    if(ppcnt>7)
        ppcnt = 0;
}
/*****************************************************************************************************/
/*  Forward Constant Speed                                                                           */
/*****************************************************************************************************/
void fconst(void)
{
    P_PORTC.PCDR.WORD = pattb[ppcnt];              /* Set portC output pattern table           */
    ppcnt++;

    if(ppcnt>7)
        ppcnt = 0;
}
/*****************************************************************************************************/
/*  Slue/Reverse Stop                                                                                */
/*****************************************************************************************************/
void frstop(void)
{
    P_PORTC.PCDR.WORD = pattb[ppcnt];               /* Set portC output pattern table          */
}
/*****************************************************************************************************/
/*  Reverse Slow Up                                                                                  */
/*****************************************************************************************************/
void rslueup(void)
{
    P_CMT.CMCOR_0 = int_cyc[sluecnt];              /* Set interrupt cycle table                */
    sluecnt++;

    P_PORTC.PCDR.WORD = pattb[ppcnt];              /* Set portC output pattern table           */
    ppcnt--;

    if(ppcnt < 0)
        ppcnt = 7;
}
```

```
/*****************************************************************************************************/
/*  Reverse Slow Down                                                                                */
/*****************************************************************************************************/
void rsluedwn(void)
{
    P_CMT.CMCOR_0 = int_cyc[sluecnt];              /* Set interrupt cycle table                */
    sluecnt--;

    P_PORTC.PCDR.WORD = pattb[ppcnt];              /* Set portC output patarn table            */
    ppcnt--;

    if(ppcnt < 0)
        ppcnt = 7;
}
/*****************************************************************************************************/
/*  Reverse Constant                                                                                 */
/*****************************************************************************************************/
void rconst(void)
{
    P_PORTC.PCDR.WORD = pattb[ppcnt];               /* Set portC output patarn table           */
    ppcnt--;

    if(ppcnt < 0)
        ppcnt = 7;
}
/*****************************************************************************************************/
/*  Other Interruption Program                                                                       */
/*****************************************************************************************************/
#pragma interrupt(dummy_f)
void dummy_f(void)
{
    /* Other Interrupt */
}
```

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sep.16.04	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corp. 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 Corp. 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 Corp. or a third party.
2. Renesas Technology Corp. 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 Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. 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 Corp. 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 Corp. by various means, including the Renesas Technology Corp. 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 Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. 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 Corp. or an authorized Renesas Technology Corp. 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 Corp. 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 Corp. for further details on these materials or the products contained therein.