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.

To all our customers

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

Renesas Technology Corp.

Cautions

Keep safety first in your circuit designs!

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

Notes regarding these materials

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

APPLICATION NOTE

1-2 Phase Excitation Control for a Stepping Motor

Introduction

Applies pins P83 to P80 and the timer W compare-match function of the H8/3664 to control a two-phase stepping motor. Control of the stepping motor is through 1-2 phase excitation.

Target Device

H8/300H Tiny Series H8/3664

Contents

Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

1. Specifications

1. Applies pins P83 to P80 and the timer W compare-match function of the H8/3664 to control a two-phase stepping motor. Control of the stepping motor is through 1-2 phase excitation.
2. This task repeatedly drives a stepping motor in the following sequence: forward rotation → stop → reverse rotation → stop.
3. Control of the stepping motor is through 1-2 phase excitation.
4. Processing for slue-up and slue-down control is carried out by software.

Figure 1.1 shows the connections for two-phase stepping-motor control.

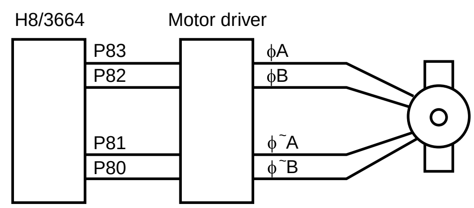


Figure 1.1 Connections for Two-Phase Stepping-Motor Control

2. Principles of Motor Control

1. Example of stepping-motor operation

   Figure 2.1 shows an example of two-phase stepping motor operation through 1-2 phase excitation where one step for the motor is 7.5 degrees of rotation. The operation is summarized below:

   1) When the pulse for excitation of a given phase is high, that phase of the stator is excited, as is shown in figure 2.1.
   2) Phase A of the stator is excited, and the permanent magnets on the rotor are positioned at phase A.
   3) Next, phase A and phase B are excited simultaneously. The permanent 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 ˜A → phase ˜A → phases ˜A and ˜B → phase ˜B → phases ˜B and A.
   4) Reverse rotation of the stepping motor is achieved by exciting the phases in the following sequence: phases ˜B and A → phase ˜B → phases ˜A and ˜B → phase ˜A → phases B and ˜A → phase B → phases A and B → phase A.
   5) The stepping motor is stopped by holding the phase excitation for a specified period at the last phase of forward or reverse rotation.

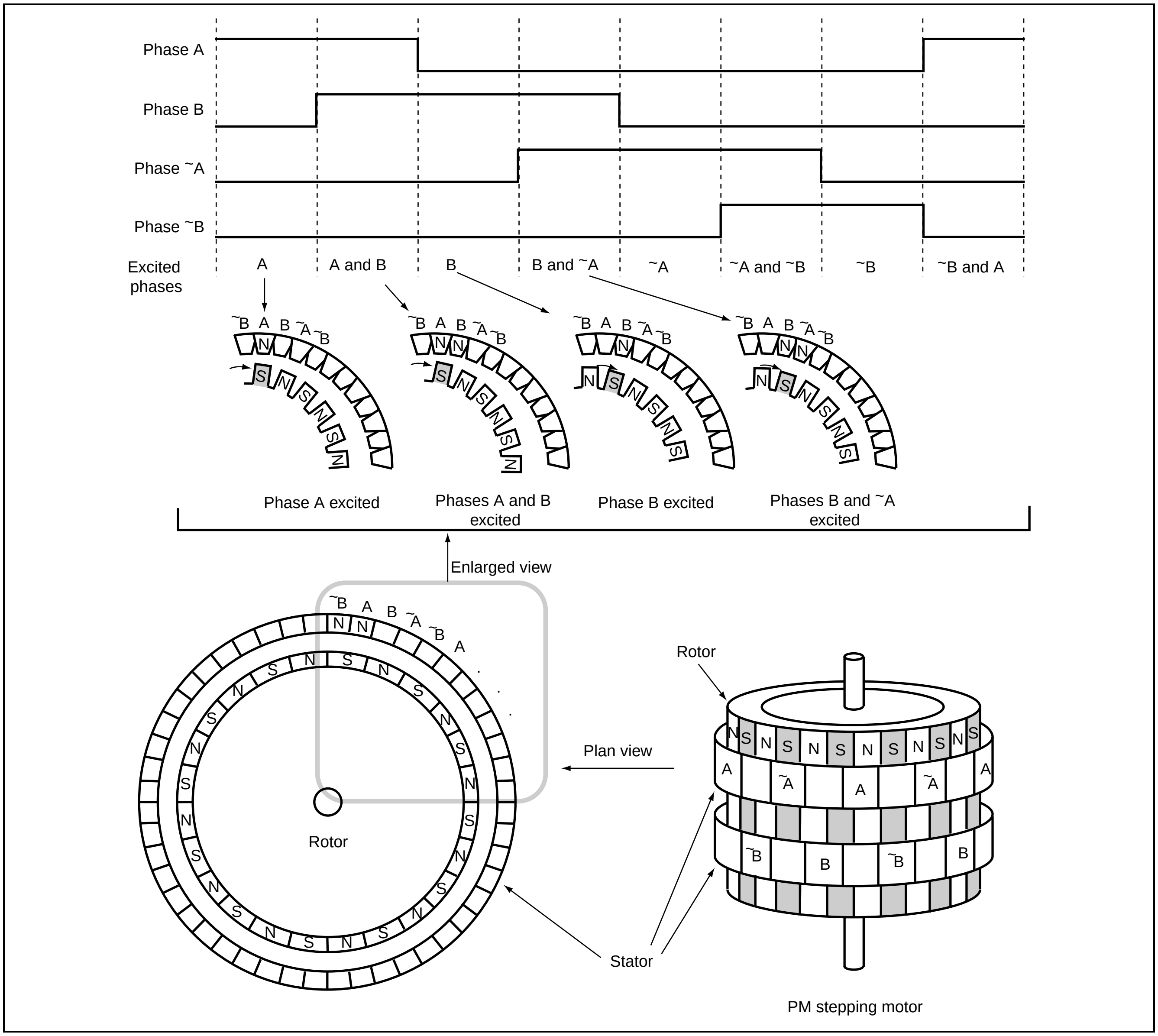


Figure 2.1 Stepping Motor Operation

2. Slue-up and slue-down operation

During slue-up/slue-down operation, pulse output is controlled to achieve acceleration or deceleration. Slue-up/slue-down operation keeps the motor in time with the control signals. In particular, if a train of short-cycle pulses is suddenly output, the motor may not be able to handle the load and does not rotate. Slue-up and slue-down operation control is applied to avoid this problem.

The control sequence is described below.

1) The pulse cycle is gradually shortened until the specified number of pulses has been output (slue up).
2) The specified number of pulses is output on a regular cycle.
3) The pulse cycle is gradually extended until the specified number of pulses has been output (slue down).

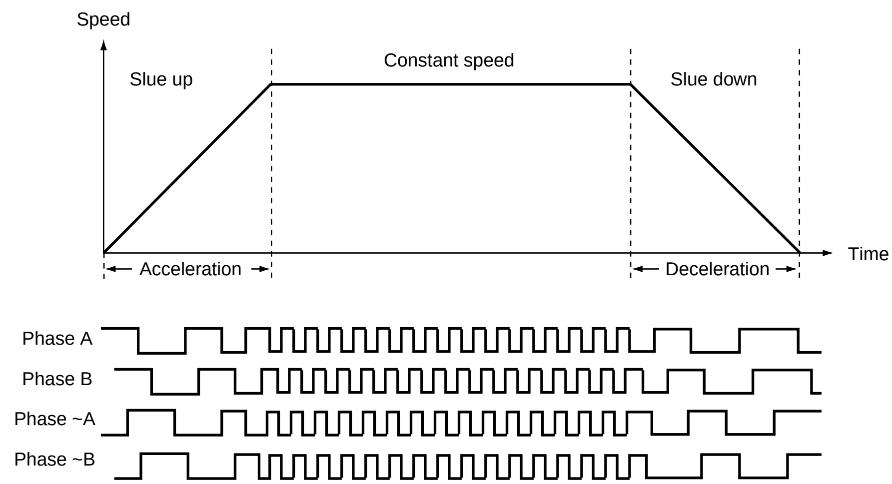


Figure 2.2 Slue-up and Slue-down Operation

3. Functions Used

1. A stepping motor of the permanent-magnet type (part number: KP6P8-701 from Japan Servo Co., Ltd.) is used in this sample task. Table 3.1 gives the standard specifications of the KP6P8-701.

Table 3.1 KP6P8-701 Standard Specifications

Item	Value
Model	KP6P8-701
Number of phases	2
Stepping angle [deg./step]	7.5
Voltage [V]	12
Current [A/phase]	0.33
Resistance of windings [Ω/phase]	36
Inductance [mH/phase]	28
Maximum static torque [mN•m]	78.4
Detent torque [mN•m]	1.3
Rotor inertia [g•cm^2]	23.7

2. The H8/3664 functions used to control the stepping motor are described below. Figure 3.1 is a block diagram of the functions used in this task.

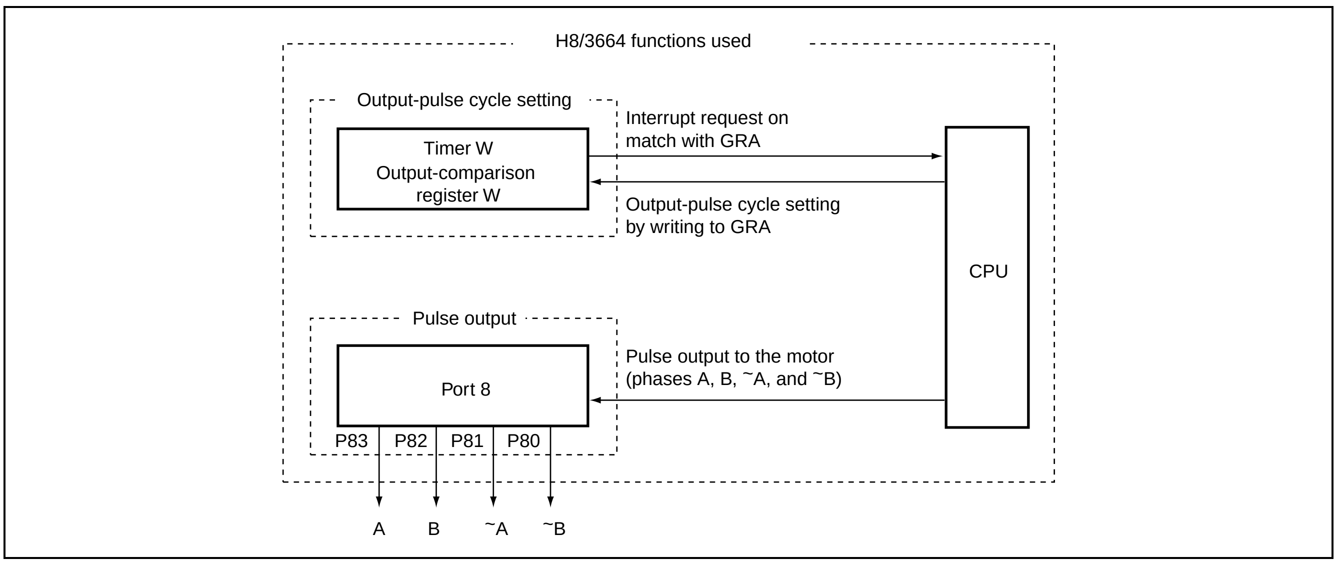


Figure 3.1 H8/3664 Functions Used

3. The timer W functions are described below.
   1) Figure 3.2 is a block diagram of timer W.

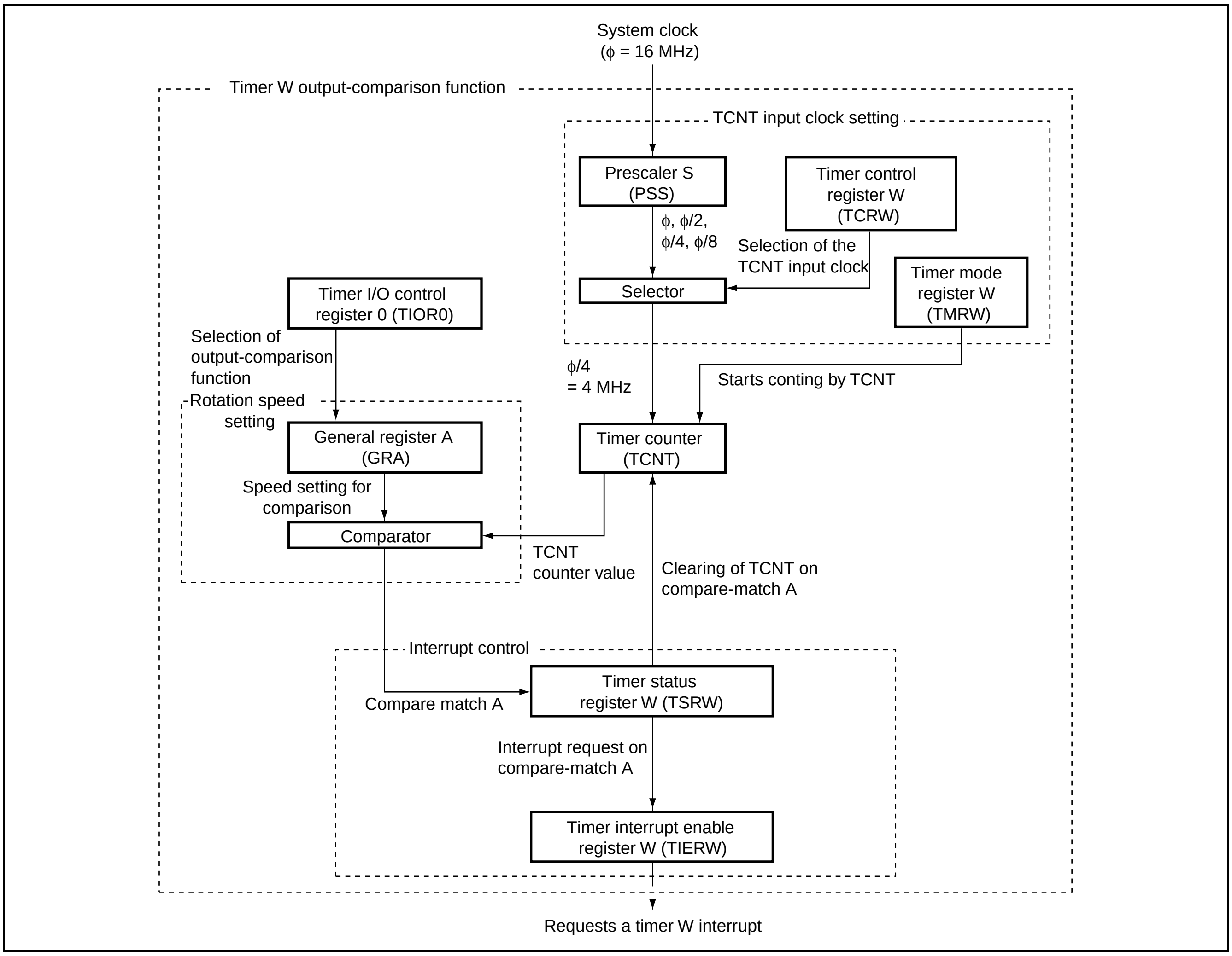


Figure 3.2 Timer W Block Diagram

2) Timer W is a 16-bit multiple-function timer that has output-comparison and input-capture functions. Output-comparison is used in this sample task. A description of the block diagram of timer W is given below.

— System clock (ϕ)

16-MHz OSC clock; supplied to the CPU and peripheral functions as the reference clock.

— Prescaler S (PSS)

13-bit counter receiving ϕ as input; incremented every cycle.

— Timer mode register W (TMRW)

Starts counting by CNT.

— Timer control register W (TCRW)

Eight-bit readable/writable register; selects the input clock for TCNT and specifies clearing of TCNT on compare-match A.

— Timer interrupt enable register W (TIERW)

Eight-bit readable/writable register; enables and disables timer interrupt requests.

— Timer status register W (TSRW)

Eight-bit register; controls the timer interrupt request signals.

— Timer I/O control register 0 (TIOR0)

Eight-bit readable/writable register; sets up the timer's output-comparison function.

— Timer counter (TCNT)

16-bit readable/writable up-counter; incremented by the input clock signal. This signal is selected from among five signals: ϕ, ϕ/2, ϕ/4, ϕ/8, and an external clock signal. In this sample task, ϕ/4 is selected.

— General register A (GRA)

16-bit readable/writable register. The value in GRA is constantly compared with the value of TCNT; when the values match, IMFA in TSRW is set to 1 and, if IMIEA in TIERW is 1, an interrupt request is issued to the CPU.

4. The port 8 functions are described below.
   1) Figure 3.3 is a block diagram of port 8.

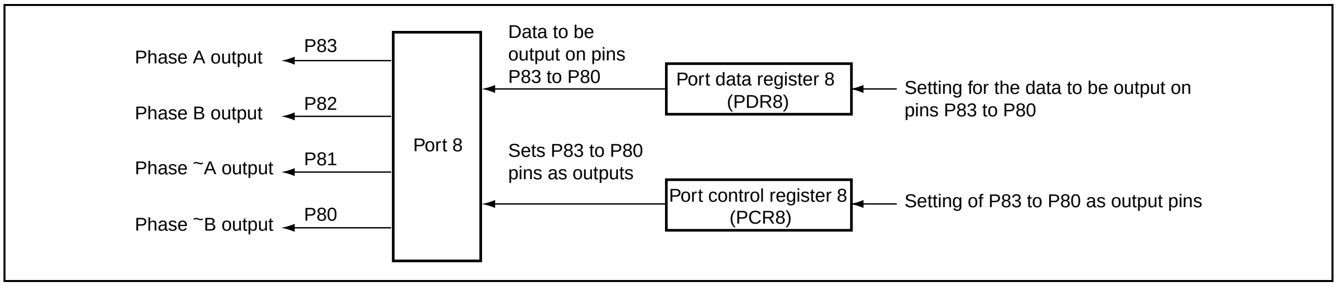


Figure 3.3 Block Diagram of Port 8

   2) Port 8 is an eight-bit I/O port. Pins P83 to P80 of port 8 are used in this sample task. The following describes the port 8 functions.

   — Port data register 8 (PDR8)

   P83 to P80 are used to excite the phases of the stepping motor.

   — Port control register 8 (PCR8)

   Sets the P83 to P80 pins as outputs.

5. The function assignments of this sample task are summarized in table 3.2.

Table 3.2 Function Assignment

Name	Assigned Function
System clock PSS TCNT	Reference clock for stepping motor control
TMRW	Starts the TCNT counter.
TCRW	Sets up TCNT operation.
TIERW	Enables/disables interrupt requests.
TSRW	Controls the interrupt request signals.
TIOR0	Sets up the output-comparison function.
GRA	Sets the duration of one step of the stepping motor.
PDR8 PCR8	Used to output the alternating phase-excitation signals for driving the stepping motor.

4. Operation

1. Figure 4.1 is the flowchart of stepping motor control.

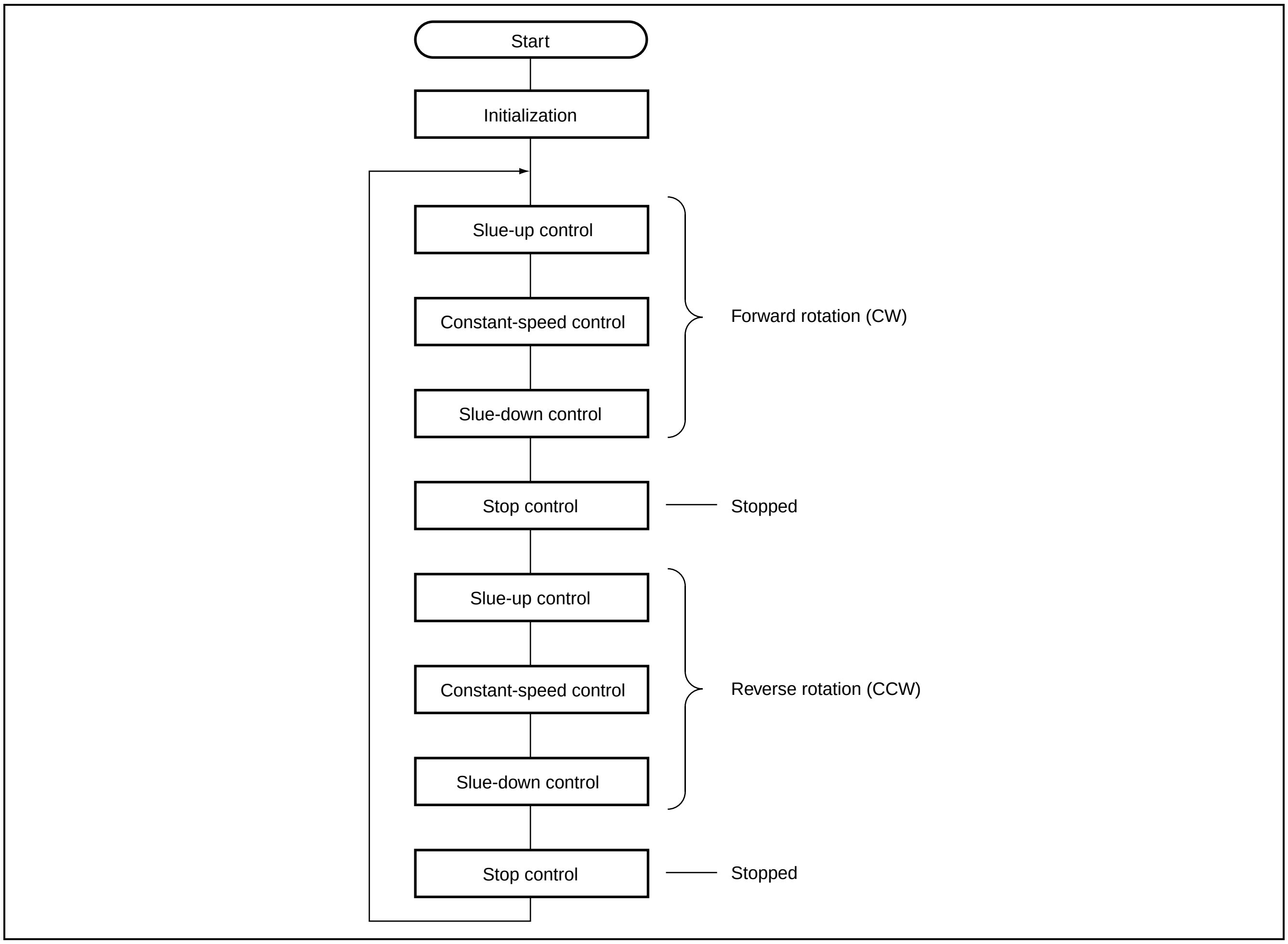


Figure 4.1 Flowchart of Stepping Motor Control

2. Calculation of timer W interrupt timing

   The timing of timer W interrupts is calculated as shown below from the setting of the GRA register, which is used as an output-comparison register:

   Timer W interrupt time = GRA/(ϕ/4)

   = GRA/(16 MHz/4)

   = GRA/4 [μs]

   where ϕ is the system-clock frequency.

3. Figure 4.2 shows the principle of slue-up control during forward rotation.

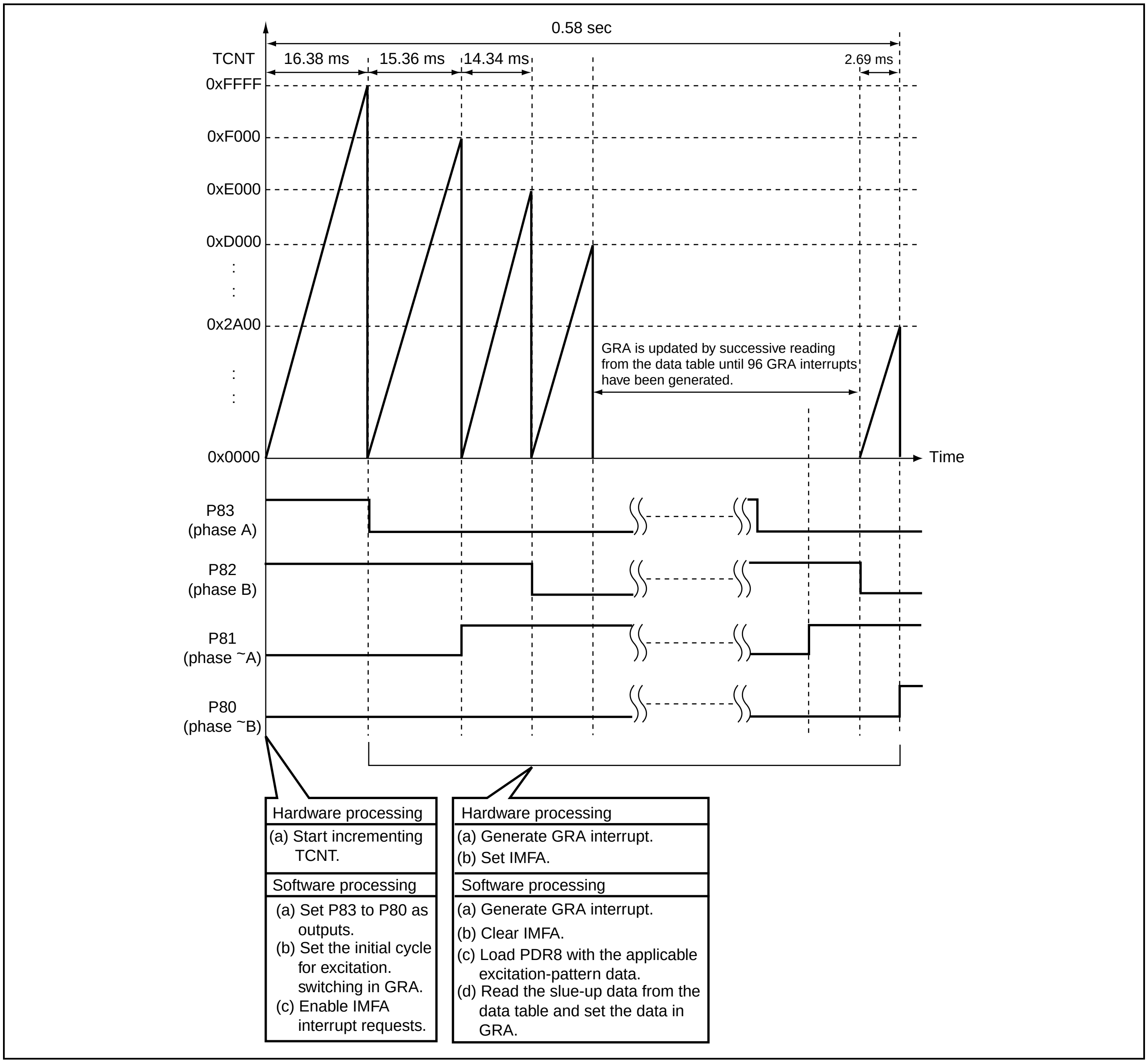


Figure 4.2 Principle of Operation: Slue-up Control During Forward Rotation

4. Figure 4.3 shows the principle of constant-speed control during forward rotation.

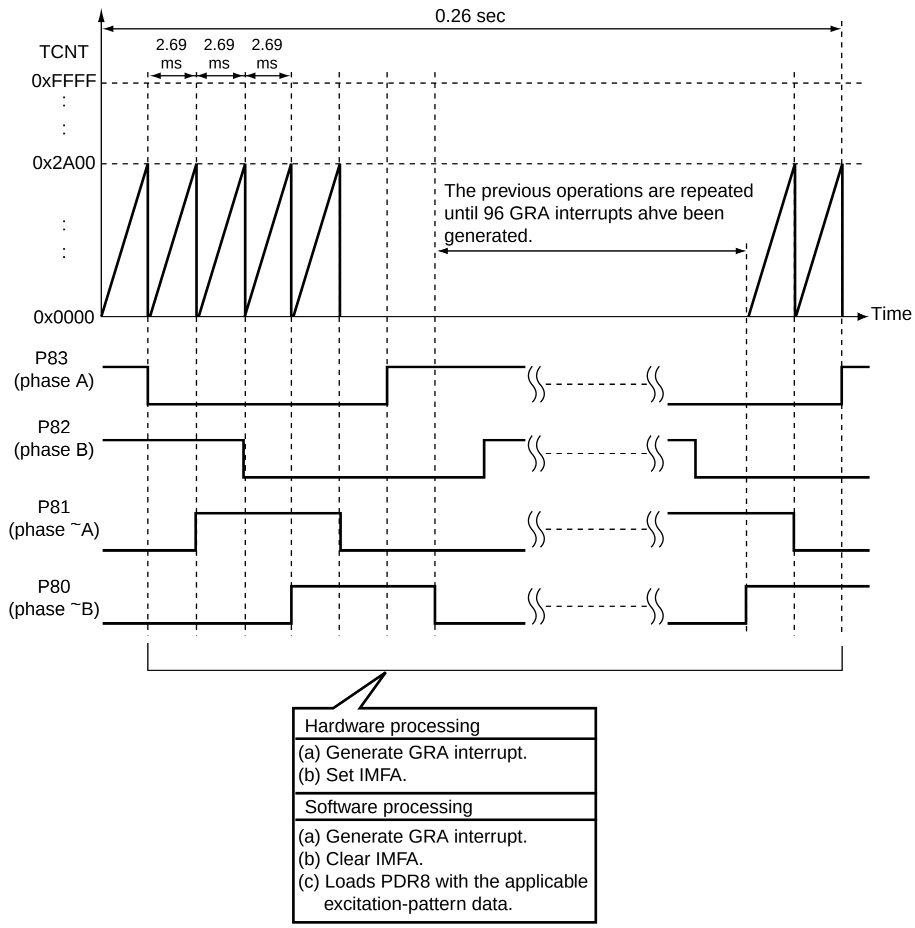


Figure 4.3 Principle of Operation: Constant-Speed Control During Forward Rotation

5. Figure 4.4 shows the principle of slue-down control during forward rotation.

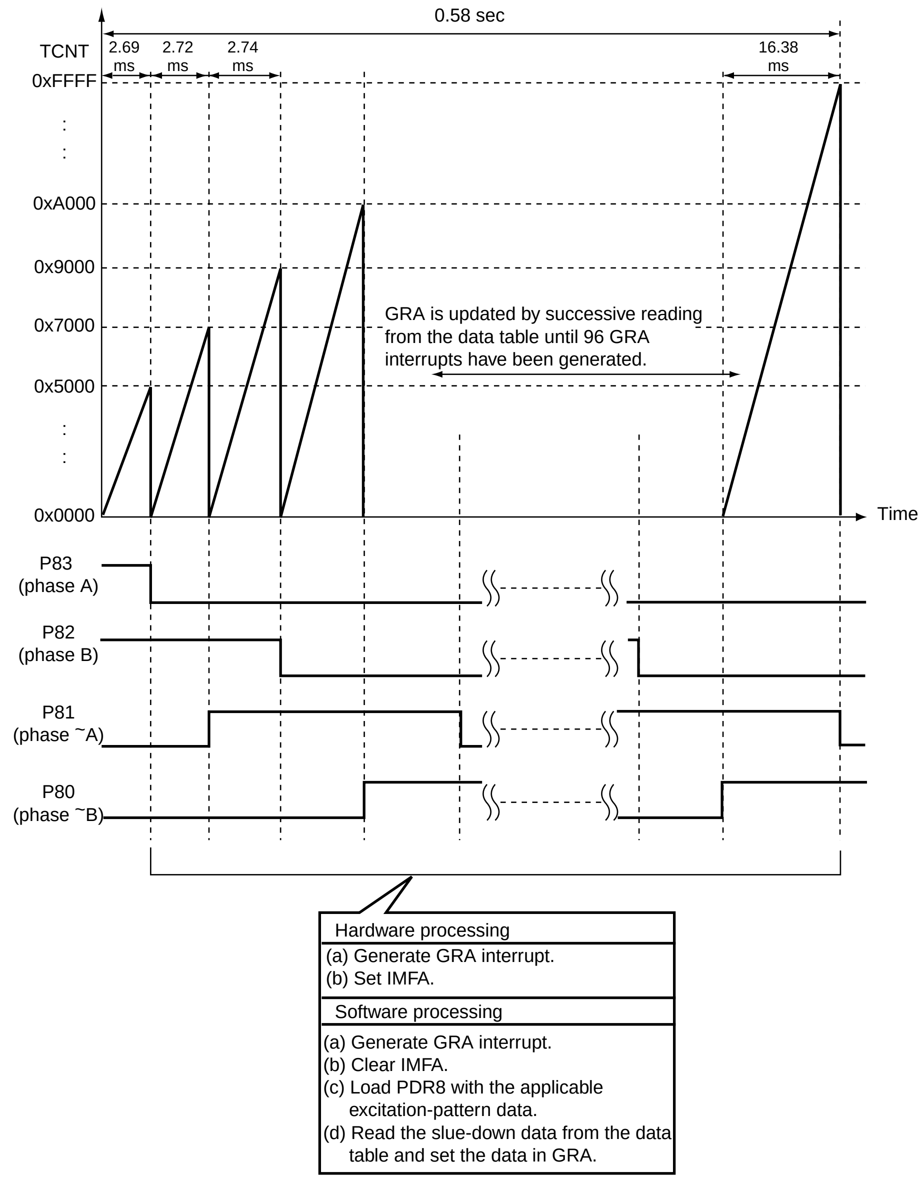


Figure 4.4 Principle of Operation: Slue-down Control During Forward Rotation

6. Figure 4.5 shows the principle of stop control.

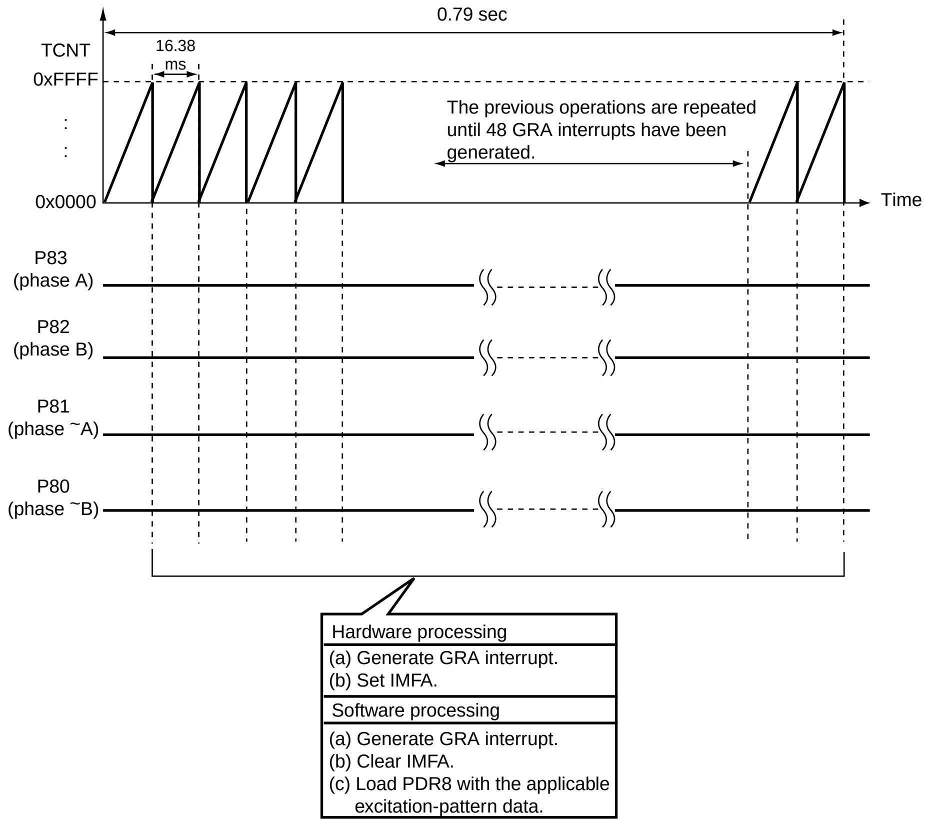


Figure 4.5 Principle of Operation: Stop Control

7. Figure 4.6 shows the principle of slue-up control during reverse rotation.

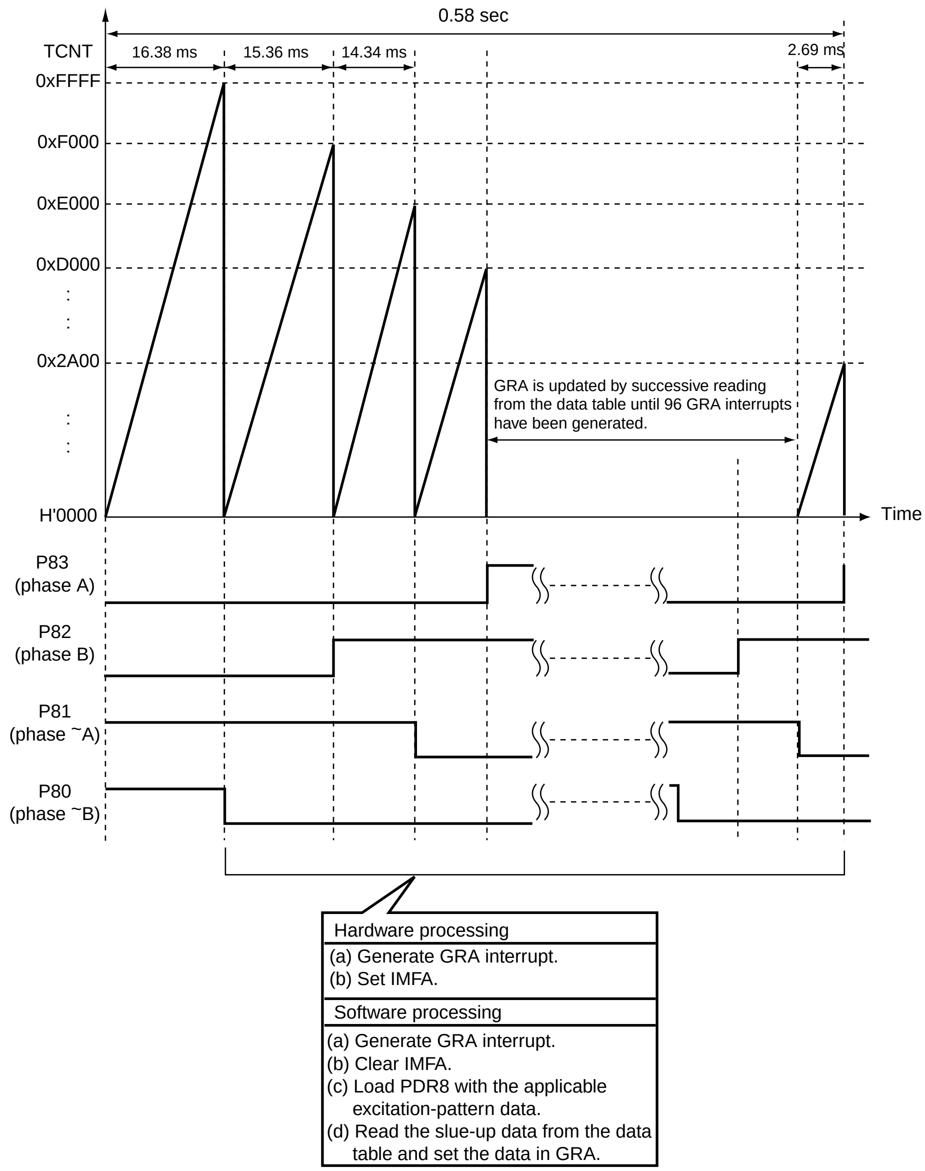


Figure 4.6 Principle of Operation: Slue-up Control During Reverse Rotation

8. Figure 4.7 shows the principle of constant-speed control during reverse rotation.

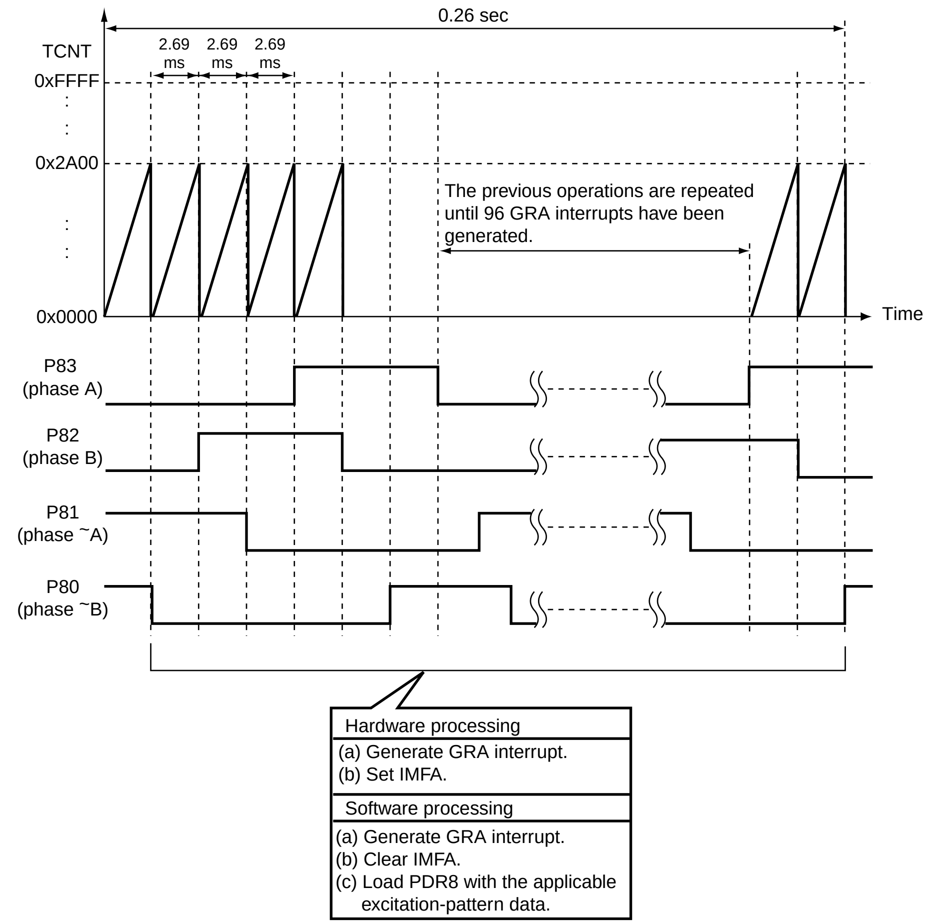


Figure 4.7 Principle of Operation: Constant-Speed Control During Reverse Rotation

9. Figure 4.8 shows the principle of slue-down control during reverse rotation.

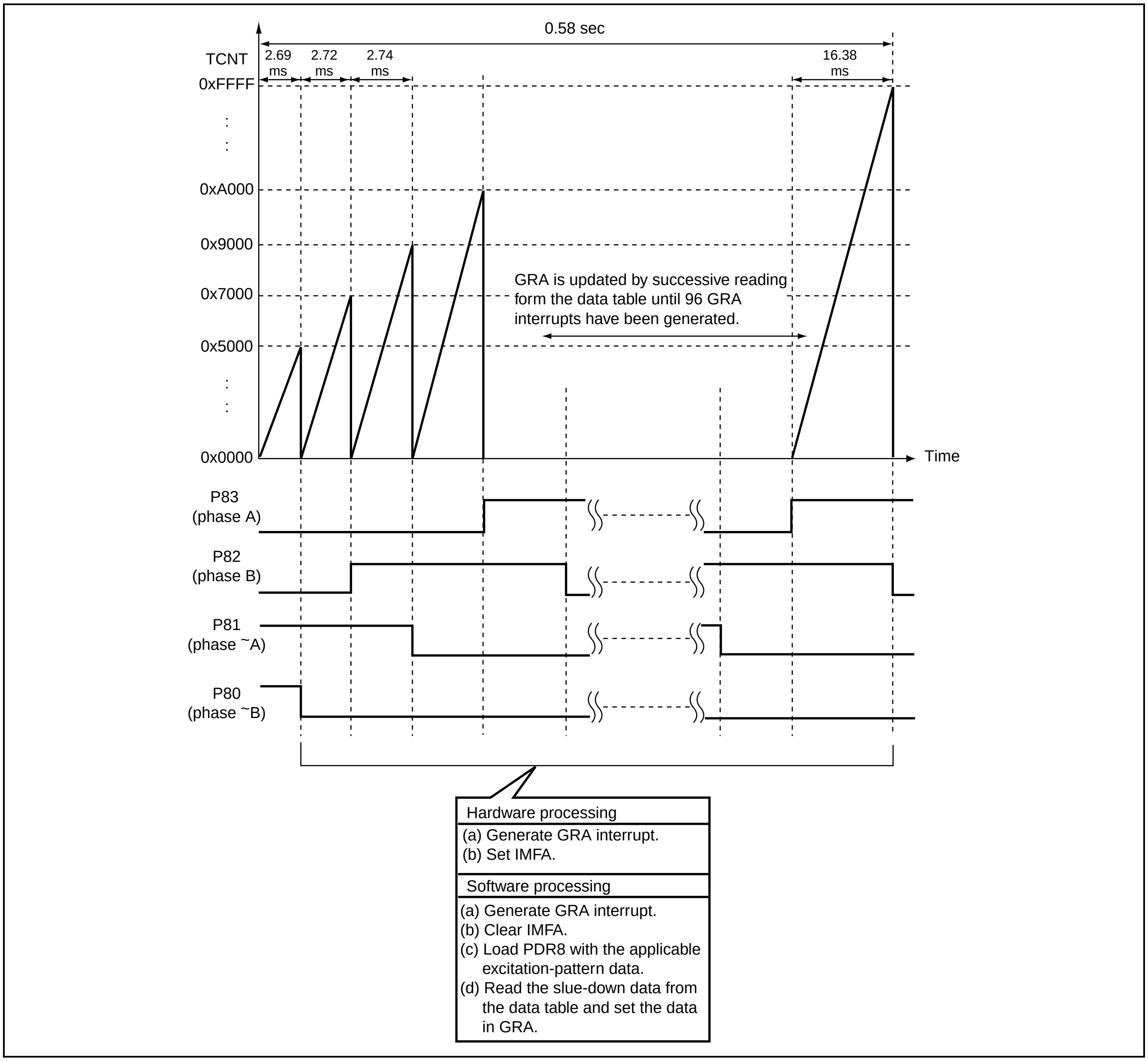


Figure 4.8 Principle of Operation: Slue-down Control During Reverse Rotation

5. Description of the Software

5.1 Modules

Table 5.1 specifies the modules used in this sample task.

Table 5.1 Description of Modules

Module Name	Label Name	Function
Main routine	main	Initializes the global variables, I/O ports, and timer W; enables interrupts.
timer W interrupt processing	twint	Core routine in handling of the stepping motor's operation.
Slue-up control during forward rotation	fslueup	Executes slue-up control during forward rotation
Slue-down control during forward rotation	fsluedwn	Executes slue-down control during forward rotation.
Constant-speed control during forward rotation	fconst	Executes constant-speed control during forward rotation.
Rotation stop	frstop	Stops forward/reverse rotation.
Slue-up control during reverse rotation	rslueup	Executes slue-up control during reverse rotation.
Slue-down control during reverse rotation	rsluedwn	Executes slue-down control during reverse rotation.
Constant-speed control during reverse rotation	rconst	Executes constant-speed control during reverse rotation.

5.2 Arguments

No arguments are used in this task.

5.3 Internal Registers Used

Table 5.2 describes the usage of internal registers in this sample task.

Table 5.2 Internal Registers Used

Register Name		Description	Address	Setting
TMRW	CTS	Timer mode register W (timer counter start): When CTS = 0, TCNT counter is stopped. When CTS = 1, TCNT counter is started.	0xFF80 Bit7	1
TCRW	CCLR	Timer control register W (counter clear): When CCLR = 0, TCNT is not cleared on compare-match A. When CCLR = 1, TCNT is cleared on compare-match A.	0xFF81 Bit7	1
	CKS2 CKS1 CKS0	Timer control register W (clock select 2 to 0): When CKS2 = 0, CKS1 = 1, and CKS0 = 0, system clock ϕ/4 is set as the input clock signal for TCNT.	0xFF81 Bit 6 Bit 5 Bit 4	CKS2 = 0 CKS1 = 1 CKS0 = 0
TIEWR	IMIEA	Timer interrupt enable register W (input capture/compare-match interrupt enable A): When IMIEA = 0, IMFA interrupt requests are disabled. When IMIEA = 1, IMFA interrupt requests are enabled.	0xFF82 Bit 0	1
TSRW	IMFA	Timer status register W (input capture/compare-match flag A): When IMFA = 0, TCNT and GRA do not match. When IMFA = 1, TCNT and GRA match.	0xFF83 Bit 0	0
TIOR0	IOA2	Timer I/O control register 0 (I/O control A2): When IOA2 = 0, GRA is used as an output-comparison register. When IOA2 = 1, GRA is used as an input-capture register.	0xFF84 Bit 2	0
	IOA1 IOA0	Timer I/O control register 0 (I/O control A1 and A0): When IOA1 = 0 and IOA0 = 0, output from pins upon compare-match is disabled.	0xFF84 Bit 1 Bit 0	IOA1 = 0 IOA0 = 0
TCNT		Timer counter: 16-bit counter driven by input system clock ϕ/4.	0xFF86	0x0000
GRA		General register A: When the value set in GRA matches that in the TCNT counter, a compare-match A signal is generated.	0xFF88	0xF000
PDR8		Port data register 8: P83 to P80 provide phase excitation signals for driving the stepping motor.	0xFFD8	0x08
PCR8		Port control register 8: When PCR8 = 0x0F, P83 to P80 are set as output pins.	0xFFE8	0x0F

5.4 Global Variables

Table 5.3 describes the global variables used in this task example.

Table 5.3 Global Variables

Variable Name	Description	Data Type/Size	Used in
Twcnt	An element of array pattbl[ ], which holds excitation-pattern data for the stepping motor.	Char/1 byte	main, fslueup, fsluedwn, fconst, frstop, rslueup, rsluedwn, rconst
Sluecnt	An element of array uptbl[ ], which is used for slue-up and slue-down control.	Char/1 byte	main, twint, fslueup, fsluedwn, rslueup, rsluedwn
Nextmode	Setting of the stepping motor's operating mode.	Char/1 byte	main, twint
Modecnt	Setting of the number of interrupts for the current operating mode	Short/2 bytes	main, twint
pattbl[8]	Excitation-pattern data table for the stepping motor.	Unsigned char/ 8 bytes	main, fslueup, fsluedwn, fconst, frstop, rslueup, rsluedwn, rconst
uptbl[96]	Interrupt time data table for slue-up and slue-down control.	Unsigned short/ 192 bytes	main, fslueup, fsluedwn, rslueup, rsluedwn

5.5 Data Table Variables

- Data table for switching the stepping motor's excitation patterns

```
pattbl[8]={
    0x08;  Excite phase A (P83).
    0x0C;  Excite phases A and B (P83 and P82).
    0x04;  Excite phase B (P82).
    0x06;  Excite phases B and ~A (P82 and P81).
    0x02;  Excite phase ~A (P81).
    0x03;  Excite phases ~A and ~B (P81 and P80).
    0x01;  Excite phase ~B (P80).
    0x09;  Excite phases ~B and A (P80 and P83).
    }
```

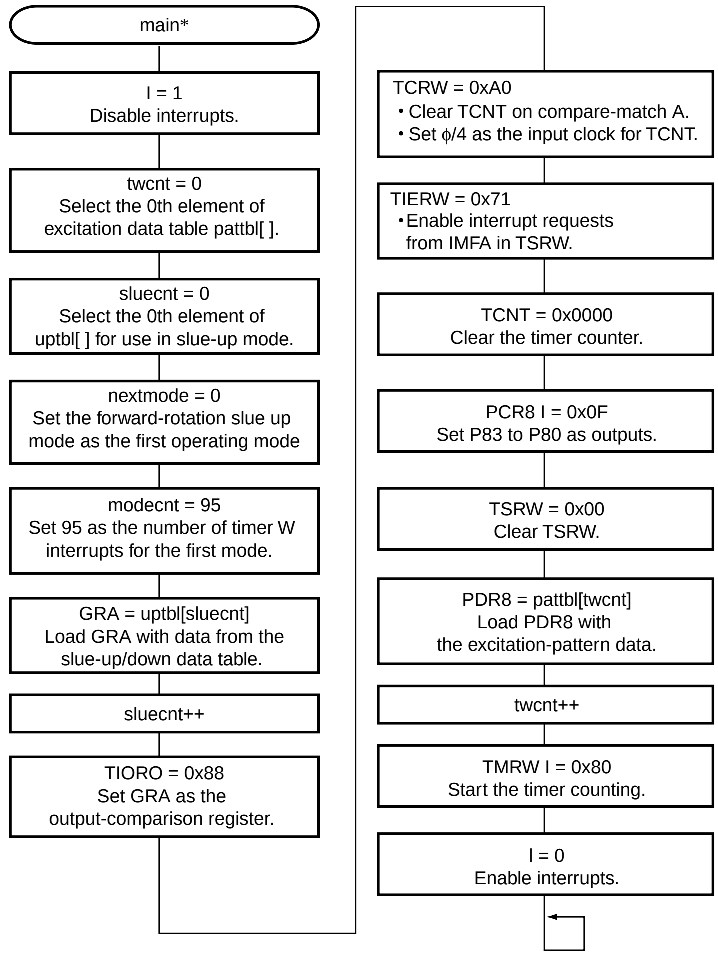
- Data table of settings for slue up and slue down

```
uptbl[96]={
    0xFFFF,0xF000,0xE000,0xD500,0xCE40,0xC738,0xC030,0xB928,0xB220,0xAB18,
    0xA410,0x9DD0,0x9790,0x9150,0x8CA0,0x87F0,0x8340,0x8020,0x7D00,0x7AA8,
    0x7850,0x75F8,0x74BD,0x7148,0x6FB8,0x6E28,0x6C98,0x6B08,0x6978,0x67E8,
    0x66BC,0x6590,0x6464,0x6338,0x620C,0x6144,0x607C,0x5FB4,0x5EEC,0x5DCA,
    0x5CA8,0x5B86,0x5A64,0x5942,0x5820,0x56FE,0x55DC,0x54BA,0x5398,0x5276,
    0x5154,0x5032,0x4F10,0x4DEE,0x4CCC,0x4BAA,0x4A88,0x4966,0x4844,0x4722,
    0x4600,0x44DE,0x43BC,0x429A,0x4178,0x4056,0x3F34,0x3E12,0x3CF0,0x3BC4,
    0x3A34,0x3890,0x373C,0x35E8,0x3494,0x33D6,0x3318,0x325A,0x319C,0x30DE,
    0x3064,0x2FEA,0x2F70,0x2EF6,0x2E7C,0x2E02,0x2D9C,0x2D36,0x2CD0,0x2C6A,
    0x2C04,0x2B9E,0x2B38,0x2AD2,0x2A6C,0x2A00,
    }
```

Data in uptbl[ ] is sequentially written to GRA each time a GRA interrupt is generated during slue-up and slue-down operations until the stepping motor has rotated once (96 steps).

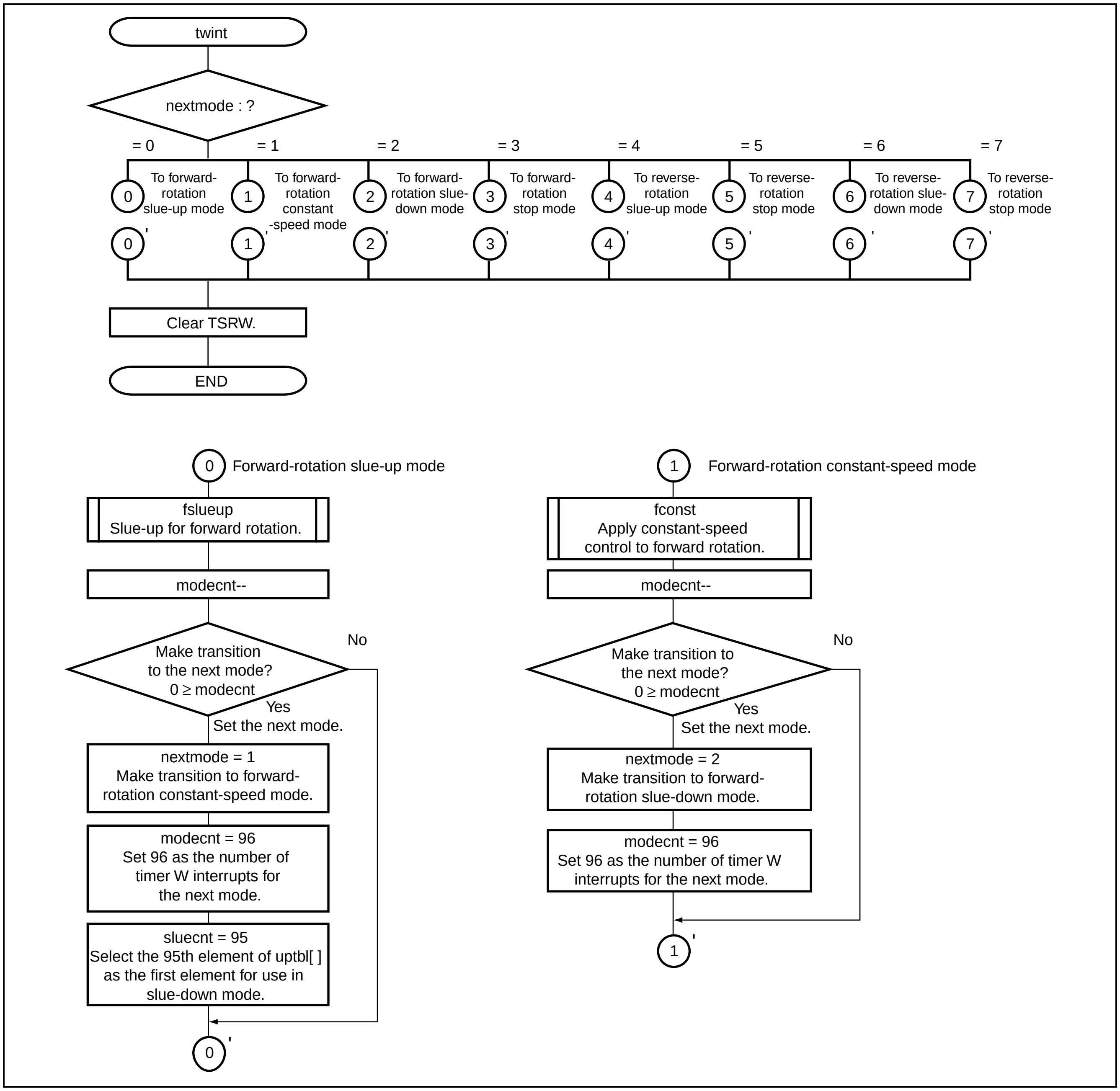
6. Flowchart

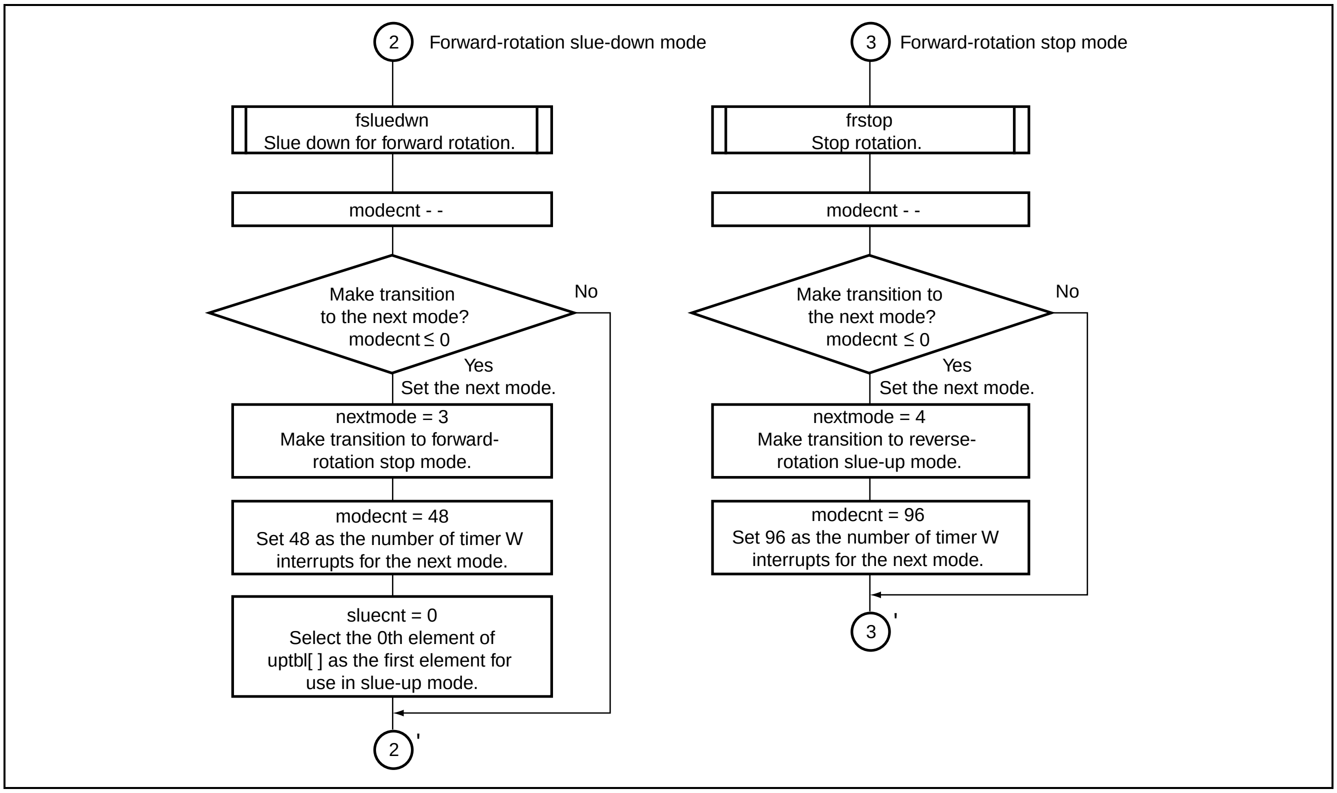
1. Function main

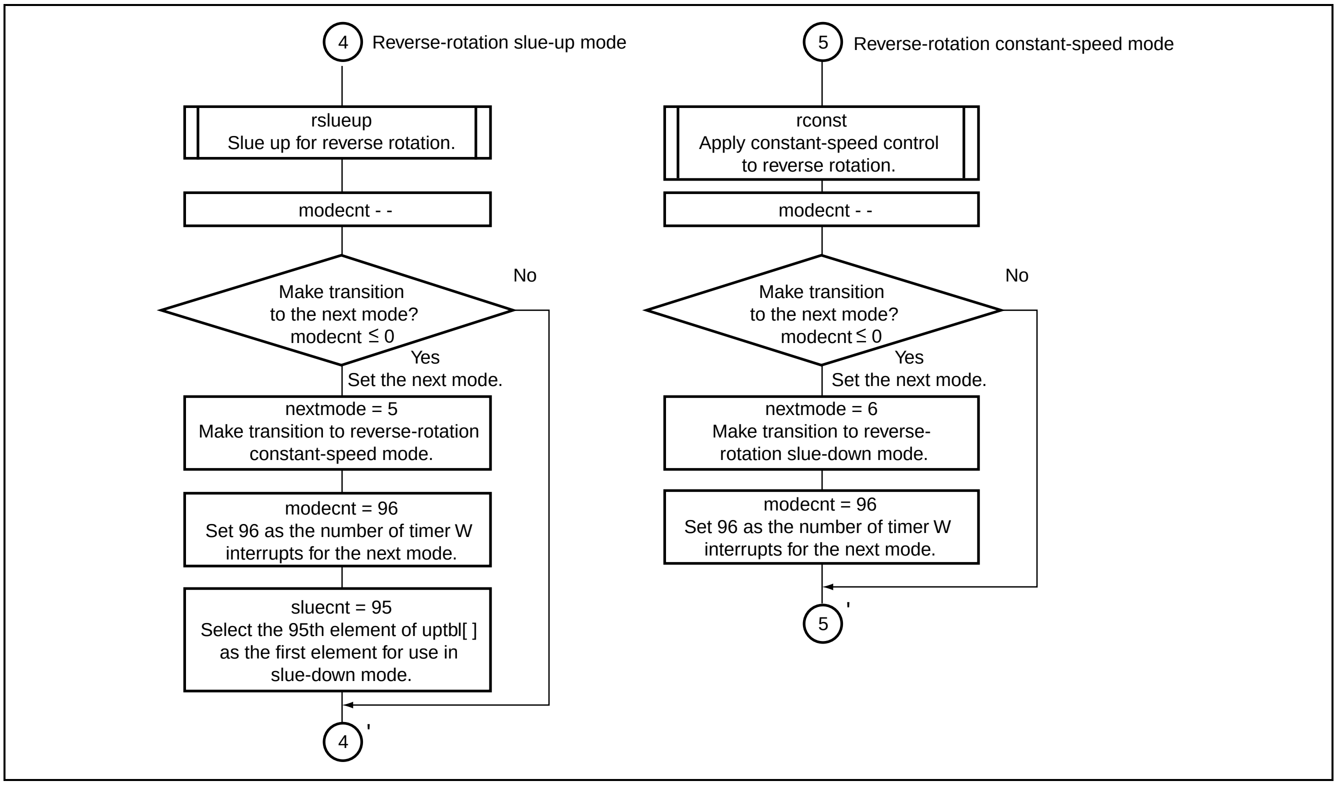


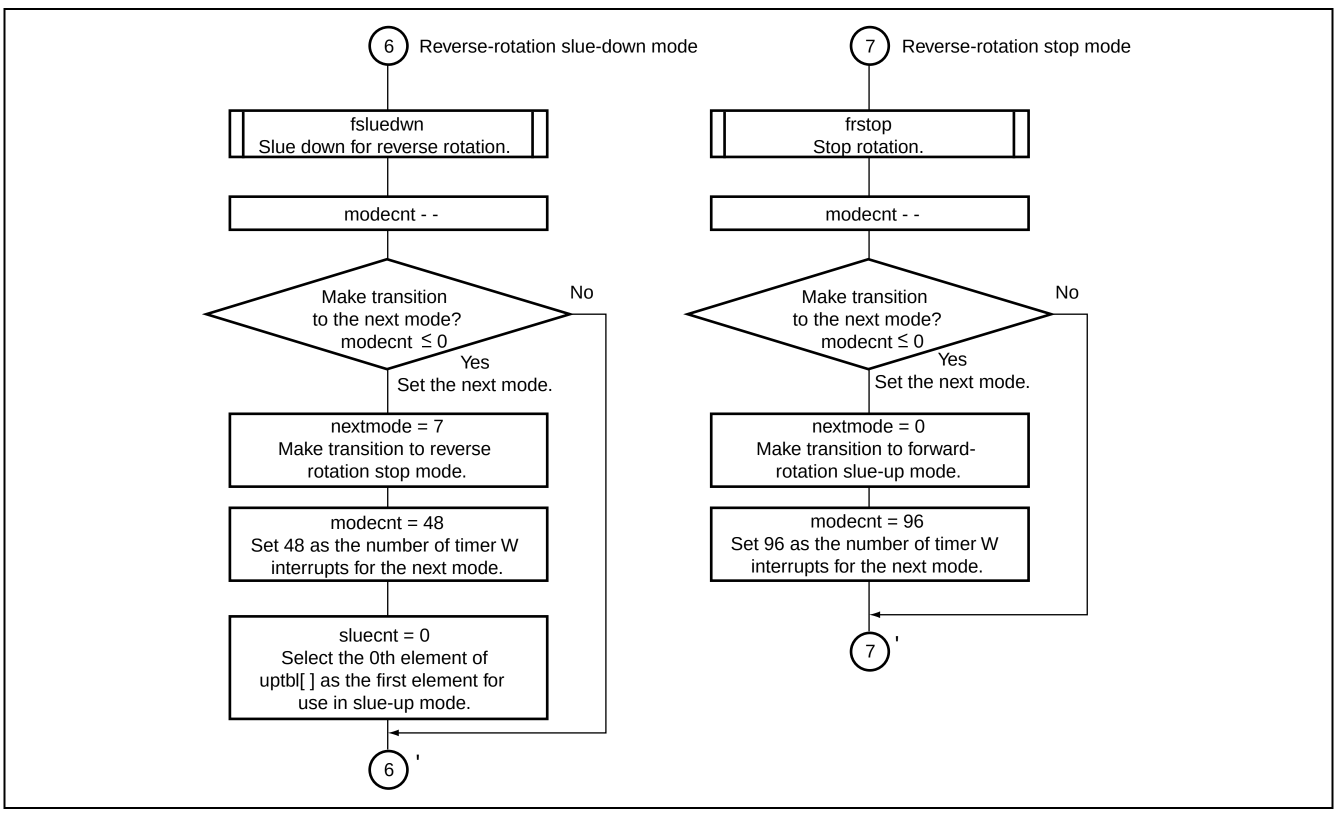
Note: * In this example, the stack pointer is set by INIT.SRC (assembly language).

2. Timer interrupts

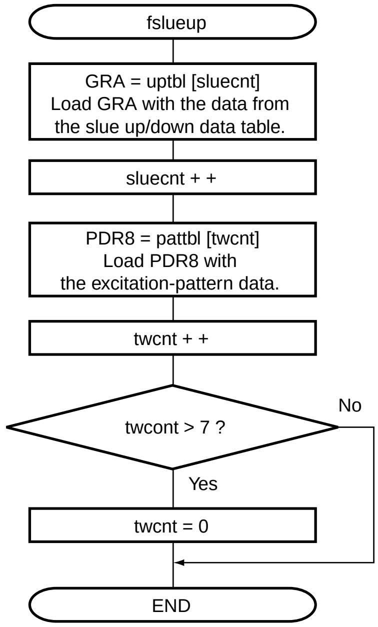



2
Forward-rotation slue-down mode
fsluedwn
Slue down for forward rotation.
modecnt - -
Make transition
to the next mode?
modecnt ≤ 0
No
Yes
Set the next mode.
nextmode = 3
Make transition to forward-
rotation stop mode.
modecnt = 48
Set 48 as the number of timer W
interrupts for the next mode.
sluecnt = 0
Select the 0th element of
uptbl[ ] as the first element for
use in slue-up mode.
2'
3
Forward-rotation stop mode
frstop
Stop rotation.
modecnt - -
Make transition to
the next mode?
modecnt ≤ 0
No
Yes
Set the next mode.
nextmode = 4
Make transition to reverse-
rotation slue-up mode.
modecnt = 96
Set 96 as the number of timer W
interrupts for the next mode.
3'

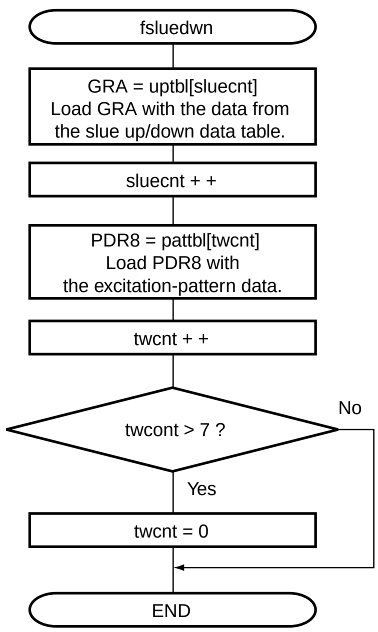

4
Reverse-rotation slue-up mode
rslueup
Slue up for reverse rotation.
modecnt - -
Make transition
to the next mode?
modecnt ≤ 0
No
Yes
Set the next mode.
nextmode = 5
Make transition to reverse-rotation
constant-speed mode.
modecnt = 96
Set 96 as the number of timer W
interrupts for the next mode.
sluecnt = 95
Select the 95th element of uptbl[ ]
as the first element for use in
slue-down mode.
4 '
5
Reverse-rotation constant-speed mode
rconst
Apply constant-speed control
to reverse rotation.
modecnt - -
Make transition
to the next mode?
modecnt ≤ 0
No
Yes
Set the next mode.
nextmode = 6
Make transition to reverse-
rotation slue-down mode.
modecnt = 96
Set 96 as the number of timer W
interrupts for the next mode.
5 '


6
Reverse-rotation slue-down mode
fsluedwn
Slue down for reverse rotation.
modecnt - -
Make transition
to the next mode?
modecnt ≤ 0
No
Yes
Set the next mode.
nextmode = 7
Make transition to reverse
rotation stop mode.
modecnt = 48
Set 48 as the number of timer W
interrupts for the next mode.
sluecnt = 0
Select the 0th element of
uptbl[ ] as the first element for
use in slue-up mode.
6 '
7
Reverse-rotation stop mode
frstop
Stop rotation.
modecnt - -
Make transition
to the next mode?
modecnt ≤ 0
No
Yes
Set the next mode.
nextmode = 0
Make transition to forward-
rotation slue-up mode.
modecnt = 96
Set 96 as the number of timer W
interrupts for the next mode.
7 '

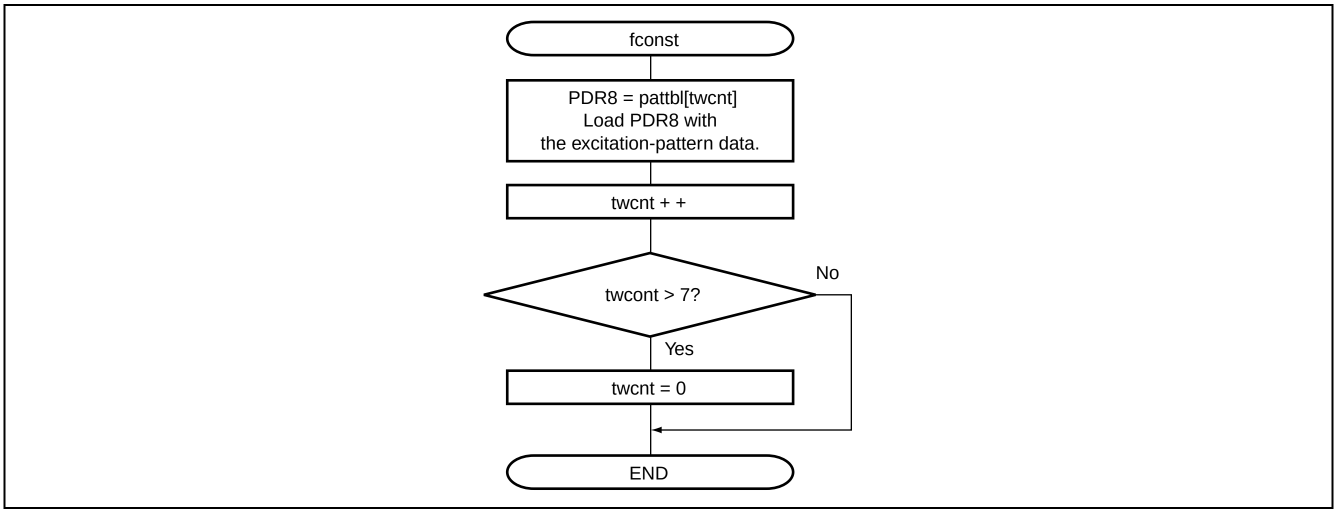
3. Slue-up control during forward rotation



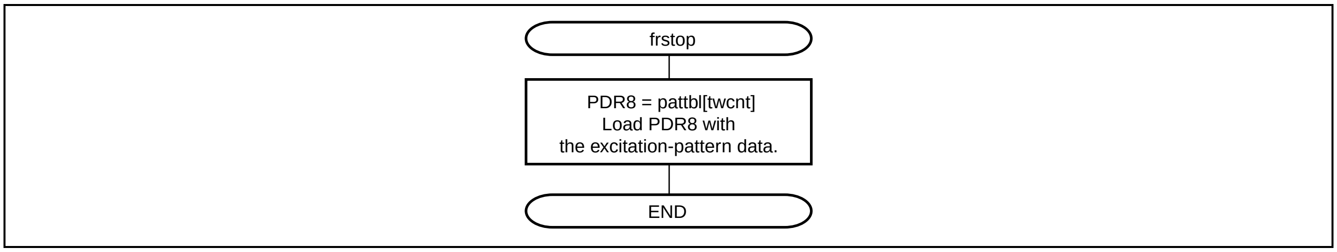
4. Slue-down control during forward rotation



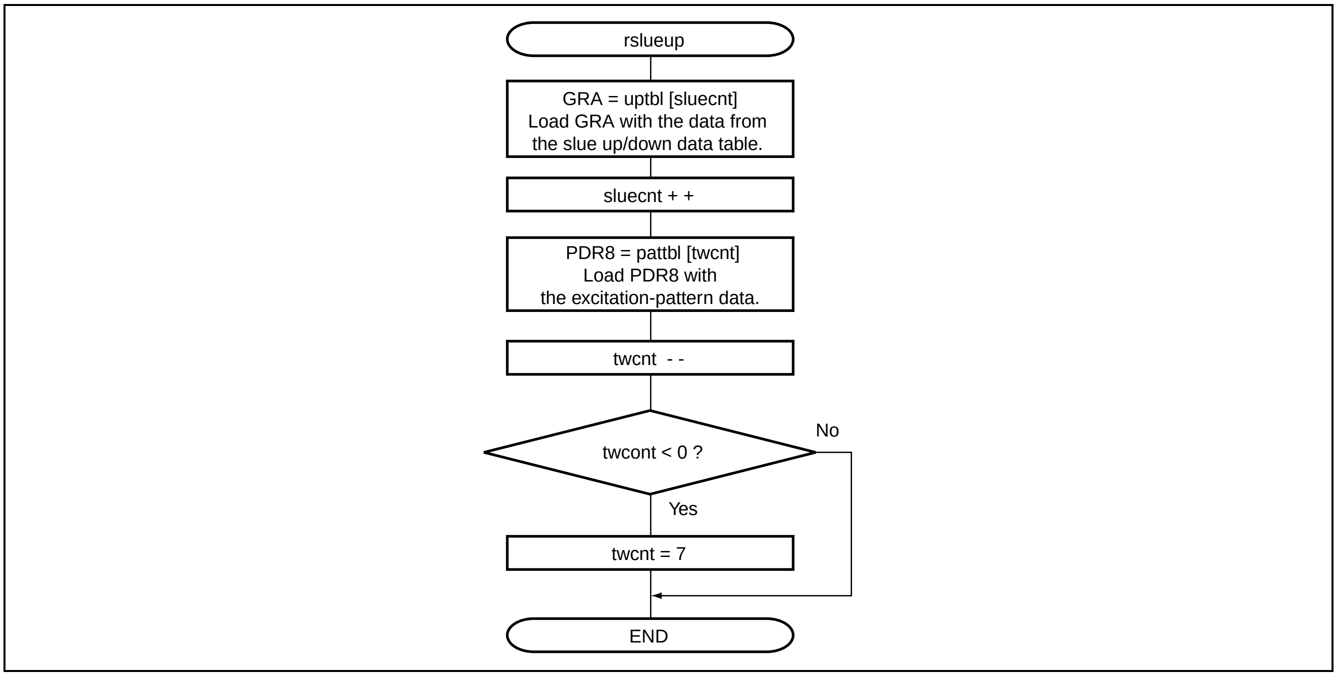
5. Constant-speed control during forward rotation



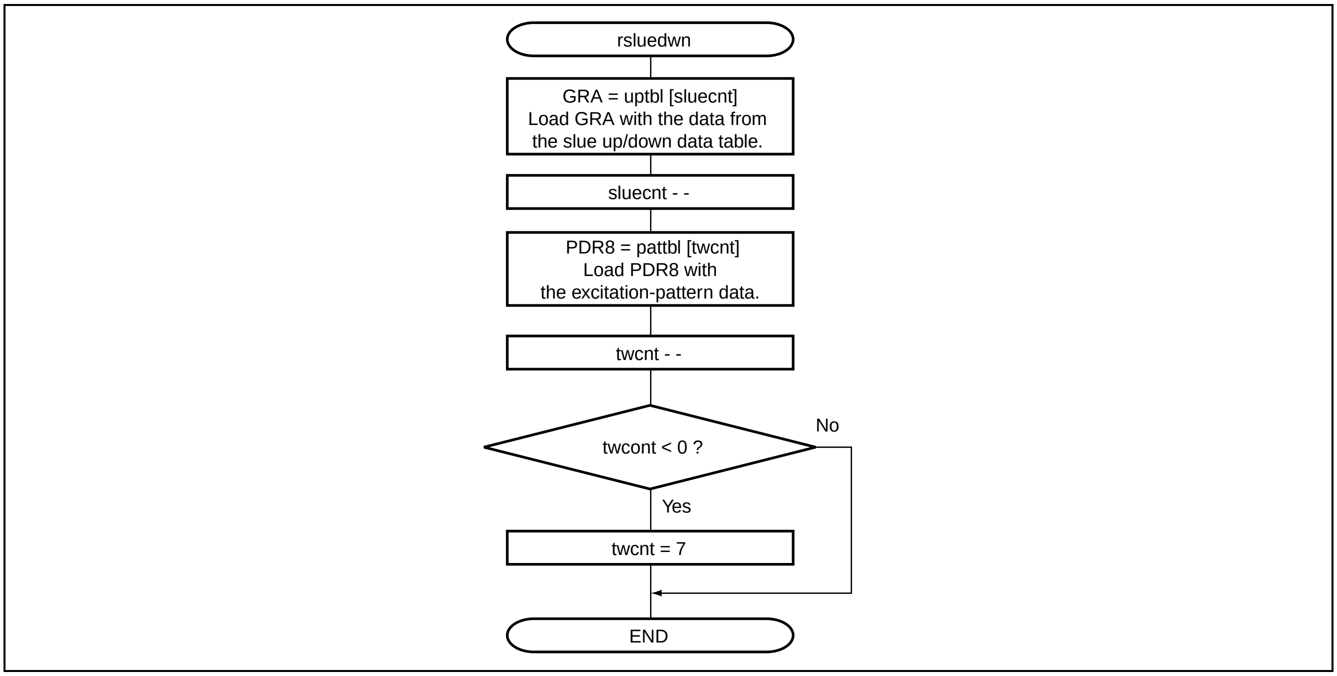
6. Stop control



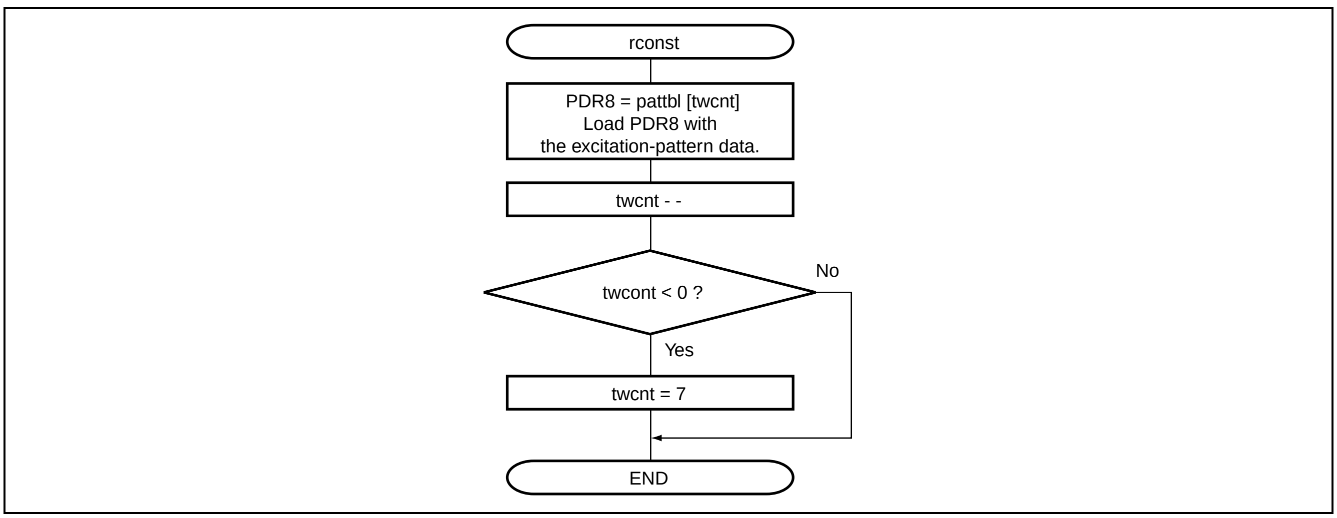
7. Slue-up control during reverse rotation



8. Slue-down control during reverse rotation



9. Constant-speed control during reverse rotation



7. Program Listing

INIT.SRC (program listing)

```
        .EXPORT  _INIT
        .IMPORT  _main
   ;
        .SECTION    P,CODE
   _INIT:
        MOV.W    #0xFF80,R7
        LDC.B    #B'10000000,CCR
        JMP      @_main
   ;
        .END
```

```
/****************************************************************************************************/
/*                                                                                                  */
/*   H8/300HN Series -H8/3664-                                                                      */
/*   Application Note                                                                               */
/*                                                                                                  */
/*   '1-2 phase excitation Stepping Motor                                                           */
/*                                                                                                  */
/*   Function                                                                                       */
/*   : Timer W Output Compare                                                                       */
/*                                                                                                  */
/*   External Clock : 16MHz                                                                         */
/*   Internal Clock : 16MHz                                                                         */
/*   Sub-Clock      : 32.768kHz                                                                     */
/*                                                                                                  */
/****************************************************************************************************/

#include    <machine.h>

/****************************************************************************************************/
/*   Symbol Definition                                                                              */
/****************************************************************************************************/
struct BIT {
    unsigned char   b7:1;                               /* bit7                                     */
    unsigned char   b6:1;                               /* bit6                                     */
    unsigned char   b5:1;                               /* bit5                                     */
    unsigned char   b4:1;                               /* bit4                                     */
    unsigned char   b3:1;                               /* bit3                                     */
    unsigned char   b2:1;                               /* bit2                                     */
    unsigned char   b1:1;                               /* bit1                                     */
    unsigned char   b0:1;                               /* bit0                                     */
};
```

```
#define     TMRW        *(volatile unsigned char *)0xFF80   /* Timer Mode Register W            */
#define     TCRW        *(volatile unsigned char *)0xFF81   /* Timer Control Register W         */
#define     TCRW_BIT    (*(struct BIT *)0xFF81)             /* Timer Control Register W         */
#define     CCLR        TCRW_BIT.b7                         /* Counter Clear                    */
#define     CKS2        TCRW_BIT.b6                         /* Clock Select 2                   */
#define     CKS1        TCRW_BIT.b5                         /* Clock Select 1                   */
#define     CKS0        TCRW_BIT.b4                         /* Clock Select 0                   */
#define     TIERW       *(volatile unsigned char *)0xFF82   /* Timer Interrupt Enable Register  */
#define     TIERW_BIT   (*(struct BIT *)0xFF82)             /* Timer Interrupt Enable Register  */
#define     IMIEB       TIERW_BIT.b1                        /* Output Compare Interrupt B Enable*/
#define     IMIEA       TIERW_BIT.b0                        /* Output Compare Interrupt A Enable*/
#define     TSRW        *(volatile unsigned char *)0xFF83   /* Timer Status Register W          */
#define     TSRW_BIT    (*(struct BIT *)0xFF83)             /* Timer Status Register W          */
#define     IMFB        TSRW_BIT.b1                         /* Output Compare Flag B            */
#define     IMFA        TSRW_BIT.b0                         /* Output Compare Flag A            */
#define     TIOR0       *(volatile unsigned char *)0xFF84   /* Timer I/O Control Register 0     */
#define     TIOR0_BIT   (*(struct BIT *)0xFF84)             /* Timer I/O Control Register 0     */
#define     IOB2        TIOR0_BIT.b6                        /* I/O Control Register B2          */
#define     IOB1        TIOR0_BIT.b5                        /* I/O Control Register B1          */
#define     IOB0        TIOR0_BIT.b4                        /* I/O Control Register B0          */
#define     TCNT        *(volatile unsigned int *)0xFF86    /* Time Counter                     */
#define     GRA         *(volatile unsigned int *)0xFF88    /* General Register A               */
#define     GRB         *(volatile unsigned int *)0xFF8A    /* General Register B               */
#define     PDR8        *(volatile unsigned int *)0xFFDB    /* Port Data Register 8             */
#define     PCR8        *(volatile unsigned int *)0xFFEB    /* Port Control Register 8          */

#pragma interrupt   (twint)
/*************************************************************/
/*  Function define                                          */
/*************************************************************/
extern void INIT ( void );                                   /* SP Set                           */
void        main ( void );
void        twint ( void );

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

```
/************************************************************/
char   twcnt,sluecnt,nextmode;
short  modecnt;

/************************************************************/
#pragma section OUTDT
unsigned char   pattbl[8] = {                               /* Stepping Motor Output Pattern Table */
    0x08,0x0C,0x04,0x06,0x02,0x03,0x01,0x09,
};

unsigned short  uptbl[96] = {                               /* Stepping Motor Output Pattern Table */
    0xFFFF,0xF000,0xE000,0xD500,0xCE40,0xC738,0xC030,0xB928,0xB220,0xAB18,
    0xA410,0x9DD0,0x9790,0x9150,0x8CA0,0x87F0,0x8340,0x8020,0x7D00,0x7AA8,
    0x7850,0x75F8,0x74BD,0x7148,0x6FB8,0x6E28,0x6C98,0x6B08,0x6978,0x67E8,
    0x66BC,0x6590,0x6464,0x6338,0x620C,0x6144,0x607C,0x5FB4,0x5EEC,0x5DCA,
    0x5CA8,0x5B86,0x5A64,0x5942,0x5820,0x56FE,0x55DC,0x54BA,0x5398,0x5276,
    0x5154,0x5032,0x4F10,0x4DEE,0x4CCC,0x4BAA,0x4A88,0x4966,0x4844,0x4722,
    0x4600,0x44DE,0x43BC,0x429A,0x4178,0x4056,0x3F34,0x3E12,0x3CF0,0x3BC4,
    0x3A34,0x3890,0x373C,0x35E8,0x3494,0x33D6,0x3318,0x325A,0x319C,0x30DE,
    0x3064,0x2FEA,0x2F70,0x2EF6,0x2E7C,0x2E02,0x2D9C,0x2D36,0x2CD0,0x2C6A,
    0x2C04,0x2B9E,0x2B38,0x2AD2,0x2A6C,0x2A00,
};

/************************************************************/
/*  Vector Address                                          */
/************************************************************/
#pragma section     V1                                      /* VECTOR SECTION SET              */
void (*const VEC_TBL1[])(void) = {                          /* 0x00 - 0x0f                     */
    INIT                                                    /* 00 Reset                        */
};
#pragma section     V2                                      /* VECTOR SECTION SET              */
void (*const VEC_TBL2[])(void) = {
    twint                                                   /* 2A Timer W Interrupt            */
};

#pragma section                                             /* P                               */
```

```
/************************************************************/
/*  Main Program                                            */
/************************************************************/
void main ( void )
{
    unsigned char tmp;

    set_imask_ccr(1);                                        /* Disable interrupts                     */

    twcnt = 0;                                               /* Output pattern table counter set       */
    sluecnt = 0;                                             /* Slue up/down table counter set         */
    nextmode = 0;
    modecnt = 95;                                            /* Motor slue mode count set "96"         */

    GRA = uptbl[sluecnt];                                    /* Initialize GRA                         */
    sluecnt++;

    TIOR0 = 0x88;                                           /* Initialize output compare function      */
    TCRW = 0xA0;                                            /* Initialize TCNT input clock period      */
    TIERW = 0x71;                                           /* Initialize IMIEA/IMIEB interrupt enable */

    TCNT = 0x0000;                                          /* Initialize TCNT                         */

    PCR8 |= 0x0F;                                           /* Port8 output                            */

    tmp = TSRW;                                             /* TSRW clear                              */
    TSRW = 0x00;

    PDR8 = pattbl[twcnt];                                   /* PDR8 set output pattern                 */
    twcnt++;

    TMRW |= 0x80;                                           /* Initialize timer mode register          */
    set_imask_ccr(0);                                       /* Interrupt enable                        */

    while(1);
}
```

```
/***********************************************************/
/*  Timer W Interrupt                                      */
/***********************************************************/
void twint ( void )
{
    unsigned char tmp;

    switch(nextmode){
        case 0:
            fslueup();                              /* Forward slue up                 */
            modecnt--;
            if(modecnt <= 0){                       /* Next mode?                      */
                nextmode = 1;                       /* nextmode = 1 constant speed     */
                modecnt = 96;                       /* Next mode countset "96"         */
                sluecnt = 95;                       /* Slue up/down table counter set  */
            }
            break;

        case 1:
            fconst();                               /* Constant speed                  */
            modecnt--;
            if(modecnt <= 0){                       /* Nextmode?                       */
                nextmode = 2;                       /* nextmode = 2 (forward slue down) */
                modecnt = 96;                       /* Nextmode countset "96"          */
            }
            break;

        case 2:
            fsluedwn();                             /* Forward slue down               */
            modecnt--;
            if(modecnt <= 0){                       /* Next mode?                      */
                nextmode = 3;                       /* nextmode = 3 (slue stop)        */
                modecnt = 48;                       /* Next mode countset "48"         */
                sluecnt = 0;                        /* Slue up/down table counter set  */
            }
            break;

        case 3:
            frstop();                               /* Slue stop                       */
            modecnt--;
            if(modecnt <= 0){                       /* Next mode?                      */
                nextmode = 4;                       /* nextmode = 4 (reverse slue up)  */
                modecnt = 96;                       /* Next mode countset "96"         */
            }
            break;
```

```
        case 4:
            rslueup();                              /* Reverse slue up              */
            modecnt--;
            if(modecnt <= 0){                       /* Next mode?                   */
                nextmode = 5;                       /* nextmode = 5 (constant speed)    */
                modecnt = 96;                       /* Next mode countset "96"      */
                sluecnt = 95;                       /* Slue up/down table counter set   */
            }
            break;

        case 5:
            rconst();                               /* Constant speed               */
            modecnt--;
            if(modecnt <= 0){                       /* Next mode?                   */
                nextmode = 6;                       /* nextmode = 6 (reverse slue down) */
                modecnt = 96;                       /* Next mode countset "96"      */
            }
            break;

        case 6:
            rsluedwn();                             /* Reverse slue down            */
            modecnt--;
            if(modecnt <= 0){                       /* Next mode?                   */
                nextmode = 7;                       /* nextmode = 7 (slue stop)     */
                modecnt = 48;                       /* Next mode countset "48"      */
                sluecnt = 0;                        /* Slue up/down table counter set   */
            }
            break;

        case 7:
            frstop();                               /* Slue stop                    */
            modecnt--;
            if(modecnt <= 0){                       /* Next mode?                   */
                nextmode = 0;                       /* nextmode = 0 (forward slue up)   */
                modecnt = 96;                       /* Next mode countset "96"      */
            }
            break;
    }

    tmp = TSRW;
    TSRW = 0x00;
}
```

```
/************************************************************/
/*  Forward slue up                                         */
/************************************************************/
void fslueup ( void )
{
    GRA = uptbl[sluecnt];                           /* GRA set slue up/down table       */
    sluecnt++;

    PDR8 = pattbl[twcnt];                           /* PDR8 set output pattern          */
    twcnt++;

    if(twcnt>7)
        twcnt = 0;
}

/************************************************************/
/*  Forward slue down                                       */
/************************************************************/
void fsluedwn ( void )
{
    GRA = uptbl[sluecnt];                           /* GRA set slue up/down table       */
    sluecnt--;

    PDR8 = pattbl[twcnt];                           /* PDR8 set output pattern          */
    twcnt++;

    if(twcnt>7)
        twcnt = 0;
}

/************************************************************/
/*  Forward constant speed                                  */
/************************************************************/
void fconst ( void )
{
    PDR8 = pattbl[twcnt];                           /* PDR8 set output pattern          */
    twcnt++;

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

```
/************************************************************/
/*  Slue/reverse stop                                       */
/************************************************************/
void frstop ( void )
{
    PDR8 = pattbl[twcnt];                                  /* PDR8 set output pattern          */
}

/************************************************************/
/*  Reverse slue up                                         */
/************************************************************/
void rslueup ( void )
{
    GRA = uptbl[sluecnt];                                  /* GRA set slue up/down table       */
    sluecnt++;

    PDR8 = pattbl[twcnt];                                  /* PDR8 set output pattern          */
    twcnt--;

    if(twcnt < 0)
        twcnt = 7;
}

/************************************************************/
/*  Reverse slue down                                       */
/************************************************************/
void rsluedwn ( void )
{
    GRA = uptbl[sluecnt];                                  /* GRA set slue up/down table       */
    sluecnt--;

    PDR8 = pattbl[twcnt];                                  /* PDR8 set output pattern          */
    twcnt--;

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

```
/*************************************************************/
/*  Reverse constant speed                                   */
/*************************************************************/
void rconst ( void )
{
    PDR8 = pattbl[twcnt];                                 /* PDR8 set output pattern           */
    twcnt--;

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

Link address specification

Section Name	Address
CV1	0x0000
CV2	0x002A
P	0x0100
C	0x0500
DOUTDT	0x0510
B	0xFB80