RX63T Group

Stepping Motor Employing Two-Phase Excitation

R01AN1240EJ0100
Rev.1.00
May 14, 2014

Introduction

This application note describes how to control a two-phase stepping motor by using the multi-function timer pulse unit 3, data transfer controller, and compare match timer functions of the RX63T Group.

This application note utilizes sample code from the following application note.

RX63T Initial Settings

RX63T Group: RX63T Initialization Example, rev. 1.00 (R01AN1252EJ0100)

Target Device

RX63T Group

When applying this application note to MCUs other than the target device, make changes as necessary to match the MCU to be used and evaluate operation carefully.

Contents

1. Specifications

The multi-function timer pulse unit 3 (MTU3), data transfer controller (DTC), and compare match timer (CMT) functions of the RX63T Group are used to control a two-phase stepping motor.

Note that this application note assumes a stepping motor with a step angle of 7.5 [degrees/step]. It describes a method for generating stepping motor control pulses that are output to the motor driver.

- The stepping motor is controlled by two-phase excitation, which involves repeatedly performing the following sequence of operations: forward → stop → reverse → stop.
- Stepping motor control pulses are generated as PWM output by the MTU3, and acceleration and deceleration processing are performed by means of DTC transfers.
- During constant control, the PWM output after acceleration control is maintained for a duration set by the CMT.
- During the motor stop period, the motor is stopped for a duration equal to the constant period.
- Forward and reverse control of the stepping motor is accomplished by inverting stepping motor control pulse waveforms that correspond to the B-phase and $\overline{\text{B}}$-phase output waveforms of the motor driver.

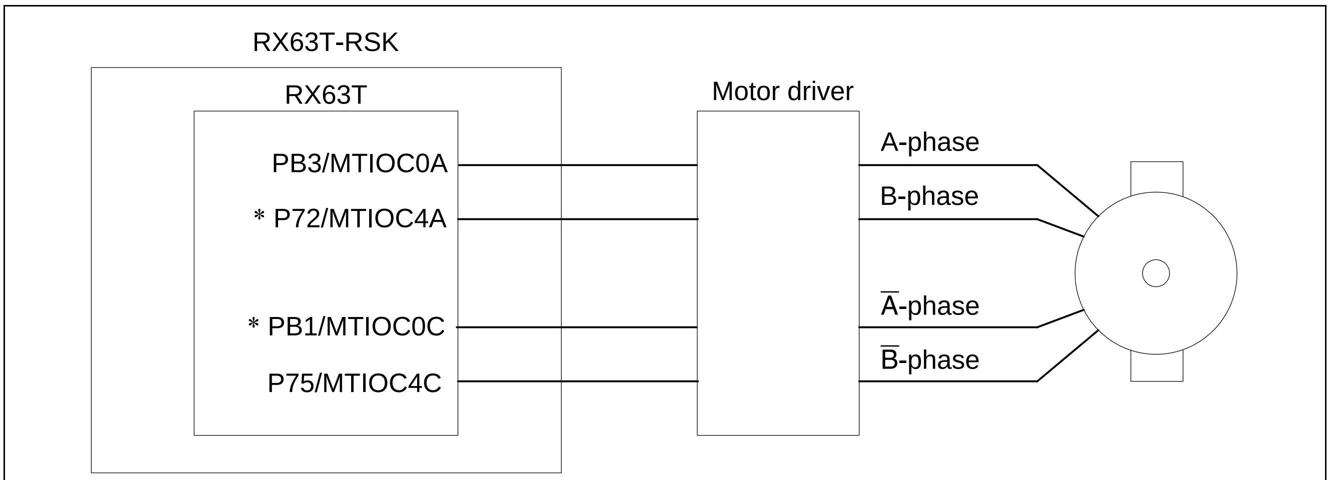
Note that a shoot-through current prevention period is interpolated between the stepping motor control pulses to prevent damage to the driver.

A wiring diagram for two-phase stepping motor control is shown below.

In the sample application, the stepping motor control pulses are output on pins MTIOC0A, MTIOC4A, MTIOC0C, and MTIOC4C.

Table 1.1 Peripheral Functions and Their Uses

Peripheral Function	Use
Multi-function timer pulse unit 3 (MTU3)	Stepping motor control pulse output
Data transfer controller (DTC)	Acceleration and deceleration control
Compare match timer (CMT)	Measuring the constant period and motor stop period



Note: * The following components are connected to the RSK ports:
P72: LED1
PB1: 4.7 kΩ pull-up resistor

Figure 1.1 Wiring Diagram for Two-Phase Stepping Motor Control

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F563T6EDFM (RX63T Group)
Operating frequency	Main clock: 16.0 MHz PLL: 192 MHz (main clock divided by 1 and multiplied by 12) System clock (ICLK): 96 MHz (PLL divided by 2) Timer module clock (PCLKA): 96 MHz (PLL divided by 2) Peripheral module clock (PCLKB): 48 MHz (PLL divided by 4) S12AD clock (PCLKD): 48 MHz (PLL divided by 4) Flash IF clock (FCLK): 48 MHz (PLL divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics High-performance Embedded Workshop Version 4.09.01.007
C compiler	RX Standard Toolchain (V1.2.1.0) RX Family C/C++ Compiler Driver V.1.02.01.000 RX Family C/C++ Compiler V.1.02.01.000 RX Family Assembler V.1.02.00.000 Optimizing Linkage Editor (V.10.02.00.000) RX Family C/C++ Standard Library Generator V.1.02.00.000
	Compiler options -cpu=rx600 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo (The integrated development environment default settings are used.)
iodefine.h version	Version 1.0F
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit+ for RX63T (Product No. R0K50563TS000BE)

3. Reference Application Note

For additional information associated with this document, refer to the following application note.

RX63T Group Initialization Example, rev. 1.00 (R01AN1252EJ0100)

4. Hardware

4.1 List of Pins

Table 4.1 lists the pins used by the sample application.

Table 4.1 Pins and Their Functions

Pin Name	I/O	Description
PB3/MTIOC0A	Output	Stepping motor operation control output (A-phase)
PB1/MTIOC0C	Output	Stepping motor operation control output ($\overline{A}$-phase)
P72/MTIOC4A	Output	Stepping motor operation control output (B-phase)
P75/MTIOC4C	Output	Stepping motor operation control output ($\overline{B}$-phase)

5. Software

5.1 Operation

5.1.1 Operating Principle of Stepping Motor

(1) Stepping Motor Operation Using Two-Phase Excitation

Figure 5.1 shows an example of the operation of a two-phase stepping motor with a step angle of 7.5 [degrees/step] by using two-phase excitation.

- As shown in figure 5.1, when a pulse is high the corresponding phase is excited.
- First, the $\overline{\text{B}}$-phase and A-phase are excited. This positions the rotor between the $\overline{\text{B}}$-phase and A-phase.
- Next, the A-phase and B-phase are excited simultaneously. This positions the rotor between the A-phase and B-phase.

  Subsequently, two-phase excitation involves exciting the adjacent two phases in sequence ($\overline{\text{B}}$-phase, A-phase → A-phase, B-phase → B-phase, $\overline{\text{A}}$-phase → $\overline{\text{A}}$-phase, $\overline{\text{B}}$-phase), causing the rotor to turn.
- In reverse operation, the excitation sequence is $\overline{\text{A}}$-phase, $\overline{\text{B}}$-phase → B-phase, $\overline{\text{A}}$-phase → A-phase, B-phase → $\overline{\text{B}}$-phase, A-phase, causing the rotor of the stepping motor to turn in the opposite direction.
- In stop operation, the final phases of the preceding forward or reverse operation continue to be excited for a fixed duration, during which the rotor of the stepping motor is stationary.

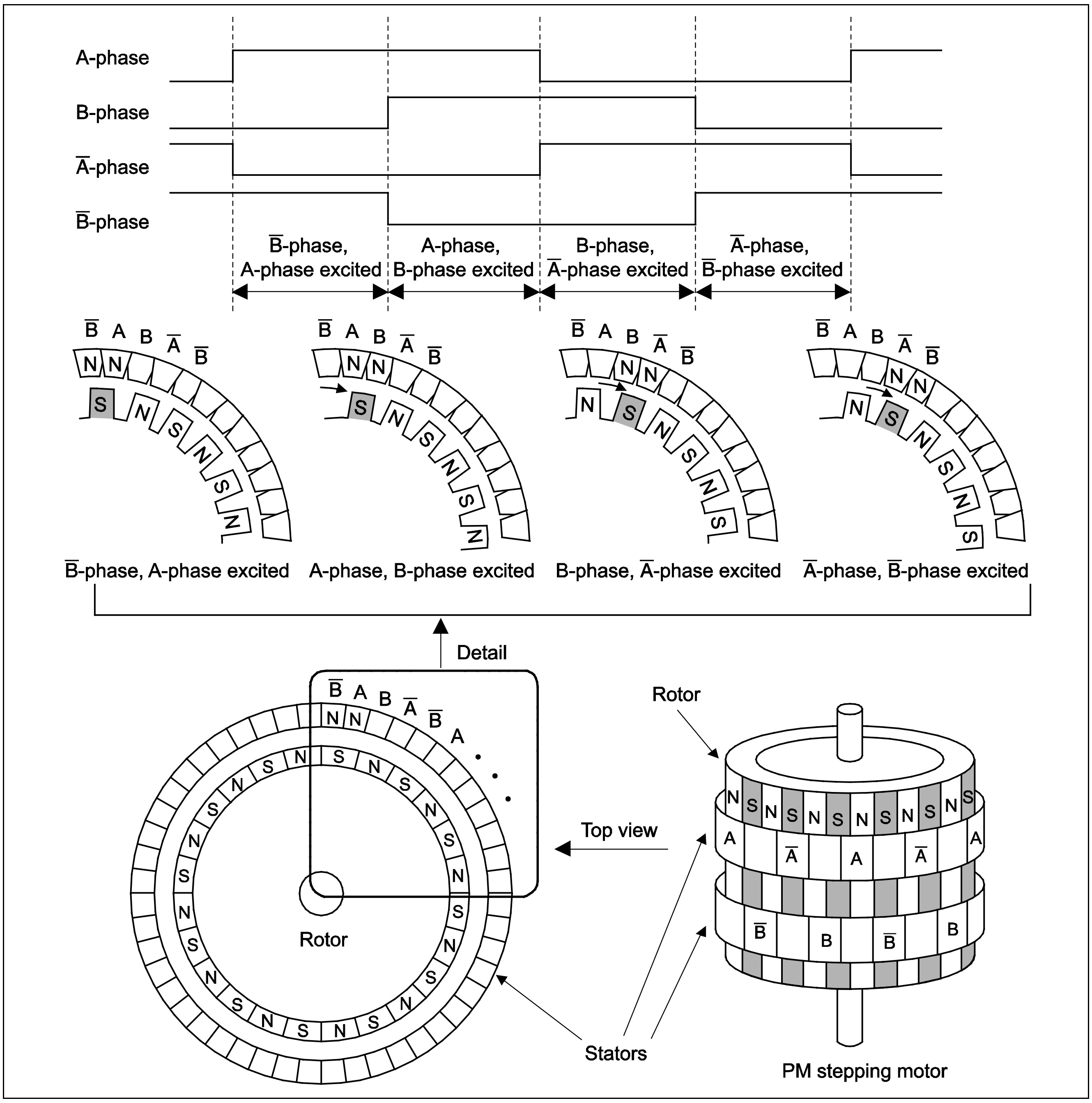


Figure 5.1 Stepping Motor Operation Example

(2) Shoot-Through Current Prevention Period

Shoot-through current prevention periods (no-overlap durations) are inserted into the sequence to prevent damage to the driver due to turn-off delay when switching among the excited phases.

Figure 5.2 shows an example with no-overlap duration output.

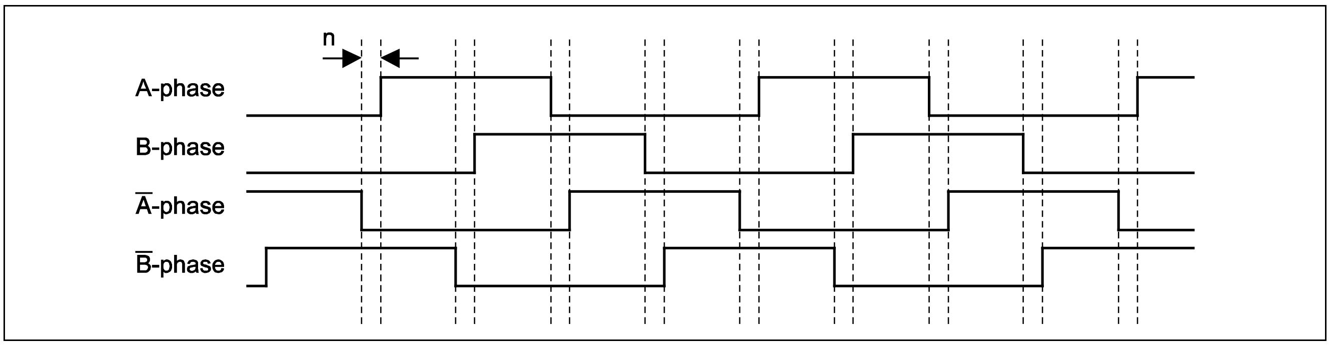


Figure 5.2 No-Overlap Duration Output Example

(3) Stepping Motor Speed Control

If short pluses are output suddenly when operating the stepping motor, the motor will be unable to keep up with the load and the rotor will stop turning. In such a case the motor is said to be "out of step." Acceleration and deceleration operation are necessary to prevent the motor from getting out of step. The operating principle of stepping motor speed control is as follows:

- To speed up the rotation of the stepping motor's rotor, the pulse period is shortened little by little (acceleration).
- To maintain the rotation of the stepping motor's rotor at an unchanging speed, the pulse period is kept fixed (constant).
- To slow down the rotation of the stepping motor's rotor, the pulse period is lengthened little by little (deceleration).
- To halt the rotation of the stepping motor's rotor, the pulse periods are discontinued and the same output state is maintained (stop).

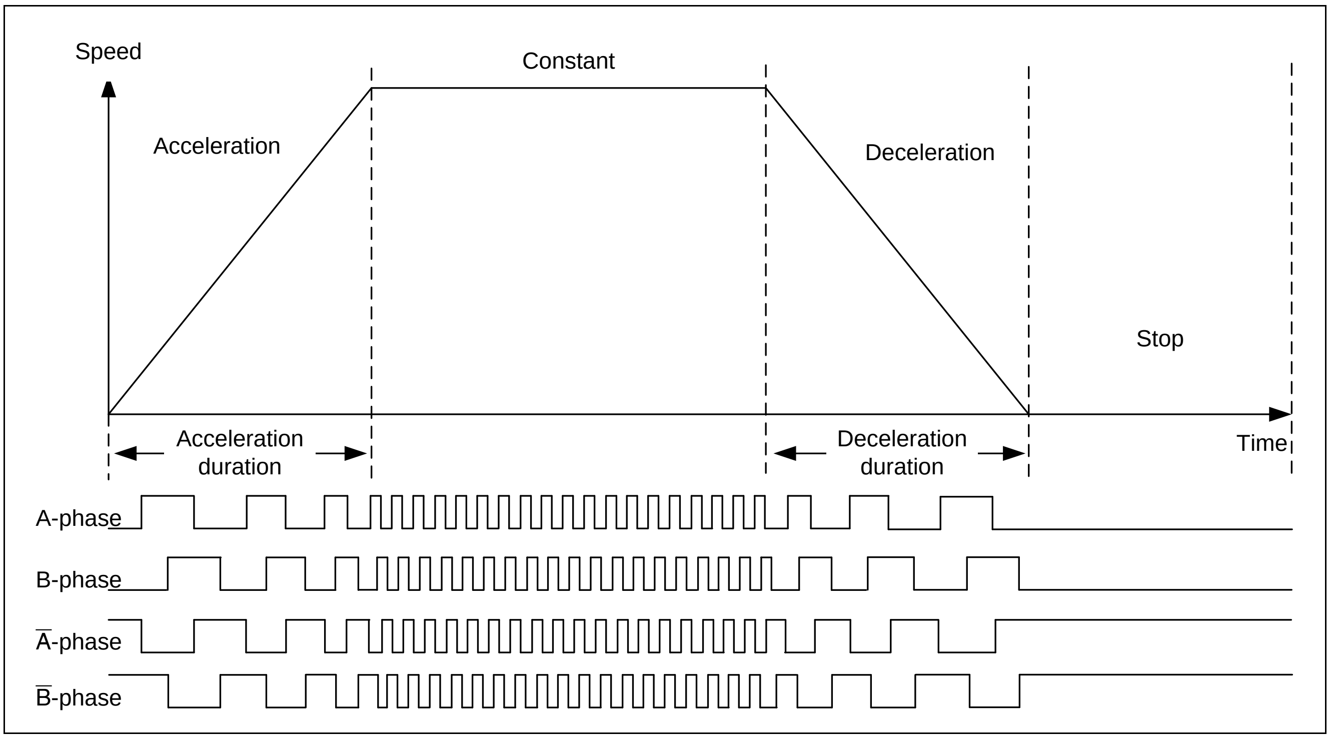


Figure 5.3 Stepping Motor Speed Control

5.1.2 Stepping Motor Control Pulse Output Control

The sample application uses the PWM output of the MTU3 and implements acceleration and deceleration by controlling the PWM output waveforms by means of DTC transfers. It implements constant and stop control of the stepping motor by continuing the PWM output waveform or maintaining the output state for the duration set by the CMT.

(1) Stepping Motor Control Pulse Output Settings

The sample application uses channel 0 (ch0) and channel 4 (ch4) of the MTU3 to output stepping motor control pulses corresponding to the A-phase, $\overline{A}$-phase, B-phase, and $\overline{B}$-phase required by the motor driver.

Synchronous operation is specified for ch0 and ch4 of the MTU3, and the counters are cleared in the period register of ch0 (TGRA0). This causes ch0 and ch4 to start simultaneously and to operate synchronously.

In addition, ch0 and ch4 of the MTU3 are set to PWM mode 1 and output stepping motor control pulses as shown in table 5.1.

Table 5.1 Assignment of MTU3 PWM Outputs and Stepping Motor Control Pulses

Channel	Output Pin	TGR Output Settings		Description
ch0	MTIOC0A pin	TGRA0	1 output	Output of A-phase stepping motor control pulse (PWM)
		TGRB0	0 output	
	MTIOC0C pin	TGRC0	1 output	Output of $\overline{A}$-phase stepping motor control pulse (PWM)
		TGRD0	0 output	
ch4	MTIOC4A pin	TGRA4	1 output	Output of B-phase stepping motor control pulse (PWM)
		TGRB4	0 output	
	MTIOC4C pin	TGRC4	1 output	Output of $\overline{B}$-phase stepping motor control pulse (PWM)
		TGRD4	0 output	

(2) Stepping Motor Acceleration/Deceleration Control and Forward/Reverse Control

Stepping motor acceleration and deceleration control involves gradually changing the period of the PWM waveforms that serve as stepping motor control pulses by modifying the MTU3 TGR values by means of DTC transfers.

A compare match with the period register of MTU3 ch0 (TGRA0) is used as the activation source for DTC transfers, and at each transfer request the corresponding period data table values are sent by sequential DTC chain transfers to the eight TGR registers of MTU3 (TGRA0 to TGRD0, TGRA4 to TGRD4), which are listed in figure 5.4.

Table 5.2 lists the DTC transfer period data tables corresponding to each of the DTC transfer destinations. For details of the period data tables, see 5.8, Description of Data Tables.

Table 5.2 Period Data Tables for DTC Transfers

		Period Data Table for DTC Transfer	
MTU3 Output Pin	**DTC Transfer Destination**	**Forward**	**Reverse**
MTIOC0A output pin	TGRA0	cyctbl_A0	cyctbl_A0
	TGRB0	cyctbl_B0	cyctbl_B0
MTIOC0C output pin	TGRC0	cyctbl_C0	cyctbl_C0
	TGRD0	cyctbl_D0	cyctbl_D0
MTIOC4A output pin	TGRA4	cyctbl_A4	cyctbl_D4
	TGRB4	cyctbl_B4	cyctbl_C4
MTIOC4C output pin	TGRC4	cyctbl_C4	cyctbl_B4
	TGRD4	cyctbl_D4	cyctbl_A4

In acceleration control, transfers are performed sequentially, starting from the bottom address in each of the period data tables, which are the transfer sources for the DTC transfers, and incrementing the address with each transfer request. In deceleration control, transfers are performed sequentially, starting from the top address in each of the period data tables, which are the transfer sources, and decrementing the address with each transfer request. In this way the period of the PWM waveform is changed little by little.

Figure 5.4 shows an example of pulse output during stepping motor acceleration control.

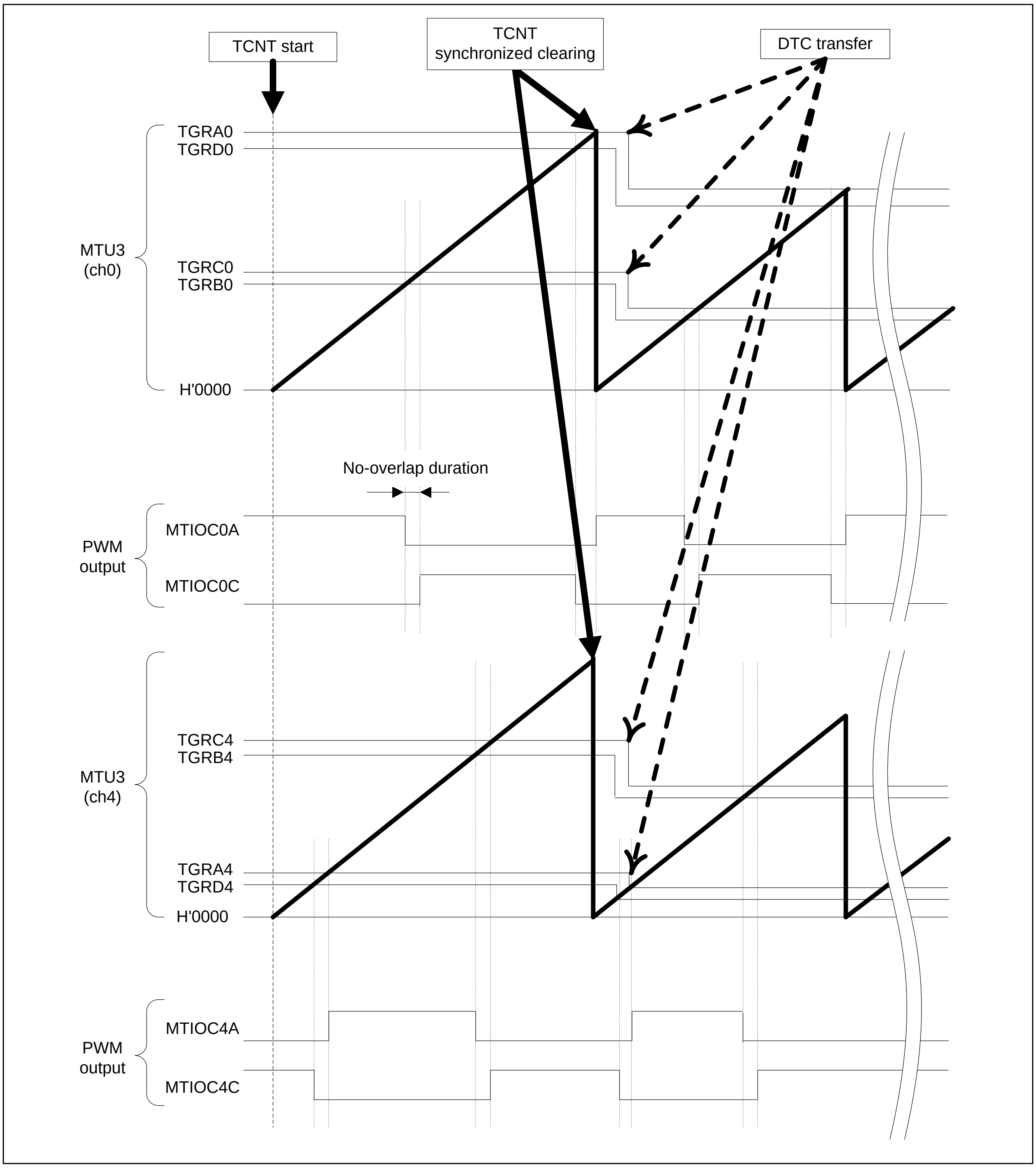


Figure 5.4 Stepping Motor Acceleration Control

Stepping motor reverse operation is implemented by exchanging the DTC transfer source addresses of the B-phase and $\overline{B}$-phase stepping motor control pulse period data tables (cyctbl_A4 and cyctbl_D4, cyctbl_C4 and cyctbl_B4), thereby inverting the B-phase and $\overline{B}$-phase output waveforms.

Figure 5.5 shows the details of DTC chain transfers during forward and reverse operation.

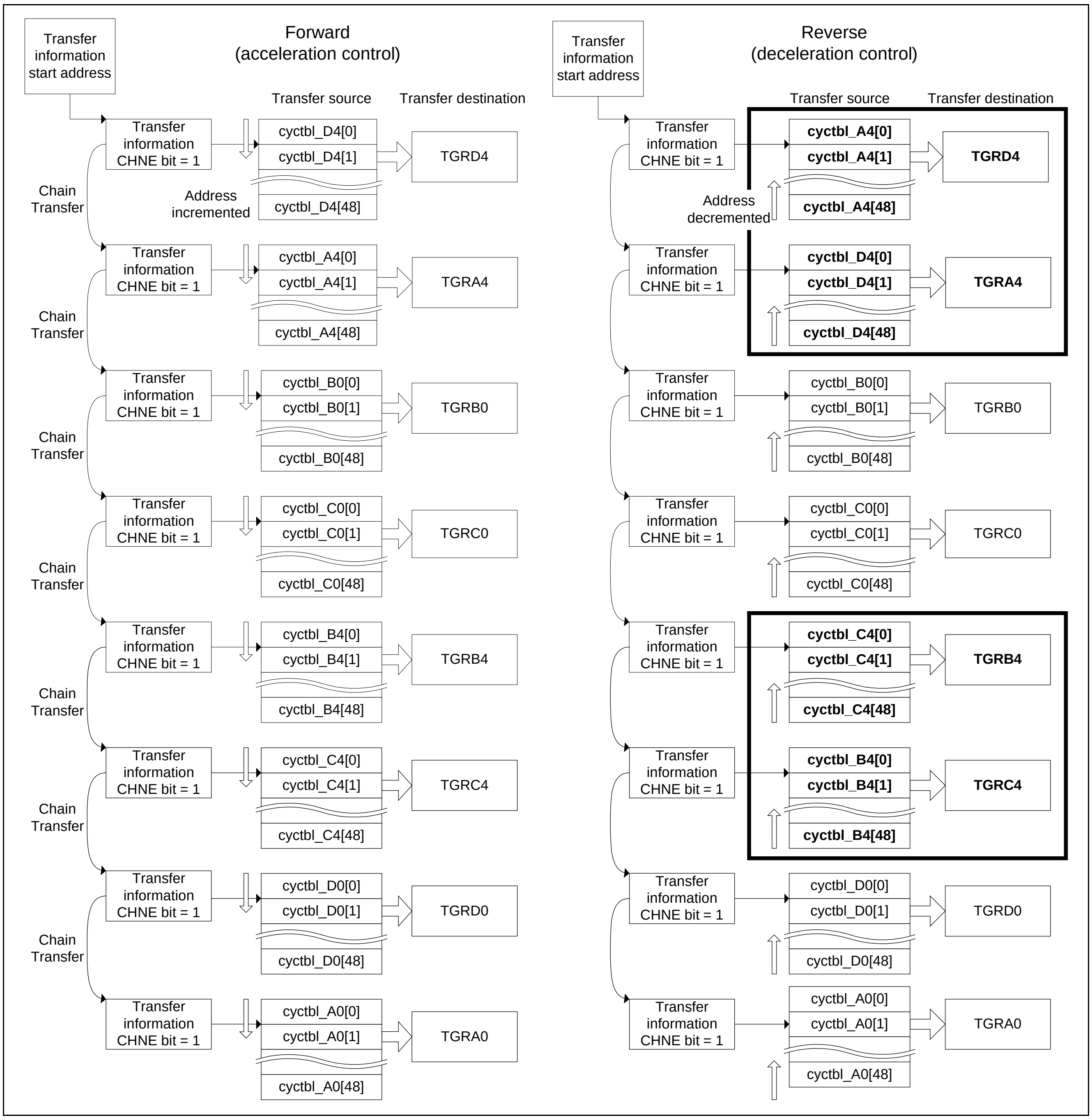


Figure 5.5 Chain Transfer

In stepping motor reverse operation, the DTC transfer source addresses of the period data tables (cyctbl_A4 and cyctbl_D4, cyctbl_C4 and cyctbl_B4) used by TGRA4 and TGRD4, TGRB4 and TGRC4 of the MTU3 are exchanged, thereby inverting the B-phase and $\overline{B}$-phase waveforms output during forward operation.

(3) Stepping Motor Constant Control and Stop Control

Stepping motor constant control is implemented by continuing the PWM waveforms that were output at the endpoint of acceleration control.

The MTU3 continues to produce the PWM output waveforms that were output at the end of acceleration control (the TGR values from the final DTC transfer). The sample application makes use of this by starting the CMT after DTC transfer ends, allowing the PWM output waveforms (constant) to continue until a CMT interrupt occurs.

When a CMT interrupt occurs, the CMT interrupt handler stops CMT operation and redoes the DTC settings required for stepping motor deceleration. A transition to deceleration control then occurs.

Stepping motor stop control is implemented by halting the operation of the MTU3 after deceleration control ends (when DTC transfer ends).

When MTU3 operation stops, the PWM output state is maintained as it was at the point when the MTU3 stopped.

In like manner to constant control, the sample application makes use of this by starting the CMT after the MTU3 stops and allowing the stepping motor to remain in the stopped state until a CMT interrupt occurs.

When a CMT interrupt occurs, the CMT interrupt handler stops CMT operation and redoes the DTC settings required for stepping motor acceleration. A transition to acceleration control then occurs.

5.1.3 DTC Settings

Table 5.3 lists the DTC transfer conditions of the sample application.

Table 5.3 DTC Transfer Conditions

Condition	Forward (Reverse) Acceleration DTC Transfers	Forward (Reverse) Deceleration DTC Transfers
Transfer information	Full address mode	Full address mode
Transfer mode	Normal mode	Normal mode
Transfer count	49	49
Transfer data	Size: Word (2 bytes)	Size: Word (2 bytes)
	Data contents: Changes values of timer general registers (TGR).	Data contents: Changes values of timer general registers (TGR).
Transfer source	On-chip ROM	On-chip ROM
Transfer destination	Timer general registers (TGR) TGRA to TGRD (ch0, ch4) of MTU3	Timer general registers (TGR) TGRA to TGRD (ch0, ch4) of MTU3
Transfer source address	Transfer source address is incremented after transfer.	Transfer source address is decremented after transfer.
Transfer destination address	Transfer destination is fixed.	Transfer destination is fixed.
Activation source	MTU3 TGIA0 compare match interrupt	MTU3 TGIA0 compare match interrupt
Interrupt	Interrupt to CPU is generated at each DTC data transfer.	Interrupt to CPU is generated at each DTC data transfer.

5.2 Required Memory Size

The memory size required for the DTC tables is listed in table 5.4.

Table 5.4 Required Memory Size

Memory Used	Size	Remarks
ROM	4,271 bytes	Of which 784 bytes are used by the data table.
RAM	2,960 bytes	
Maximum user stack usage	52 bytes	
Maximum interrupt stack usage	52 bytes	

Note: The required memory size varies depending on the C compiler version and compile options.

5.3 File Composition

Table 5.5 lists the files used in the sample code. Files generated by the integrated development environment are not included in this table.

Table 5.5 Files Used in the Sample Code

File Name	Outline	Remarks
r_init_stop_module.c r_init_stop_module.h r_init_clock.c r_init_clock.h r_init_non_existent_port.c r_init_non_existent_port.h	RX63T Group Sample initialization program	See the application note that describes the RX63T initialization example for details.
main.c	Initial settings program for operating stepping motor employing two-phase excitation	
timer_int.c	Interrupt handler program	
motor.h	Included header file related to operating stepping motor employing two-phase excitation	
intprg.c	Vector-related definitions MTU3 interrupt function and CMT interrupt function added.	

5.4 Option-Setting Memory

Table 5.6 lists the option-setting memory configured in the sample code. When necessary, set a value suited to the user system.

Table 5.6 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	After a reset, the IWDT is stopped. After a reset, the WDT is stopped.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	After a reset, voltage monitoring reset 0 is ignored.
MDES*[1]	FFFF FF83h to FFFF FF80h		(In single-chip mode)
		FFFF FFFFh	Little endian
		FFFF FFF8h	Big endian

Note: 1. The settings in this sample code set up little endian operation. See section 6.2, Endian, for details on switching the endian mode.

5.5 List of Constants

Table 5.7 lists the constants used in the sample code.

Table 5.7 List of Constants

Constant	Value	Description
UPTIME	49	DTC transfer count during acceleration/deceleration control
CNSTTIME	48605	Setting time for constant control
NON_OVR	100	Shoot-through current prevention period (no-overlap duration)
STOPTIME	48605	Stop setting time
CYCTIME	15000	Data table period setting time
CHGTIME	200	Data table period change time

5.6 List of Structures and Unions

Figure 5.6 lists the structures and unions used in the sample code.

```
#pragma bit_order left
#pragma unpack

struct st_dtc_full{
  union{
    unsigned long LONG;
    struct{
      unsigned long MRA_MD      :2;     /* MRA.MD bit */
      unsigned long MRA_SZ      :2;     /* MRA.SZ bit */
      unsigned long MRA_SM      :2;     /* MRA.SM bit */
      unsigned long             :2;
      unsigned long MRB_CHNE    :1;     /* MRB.CHNE bit */
      unsigned long MRB_CHNS    :1;     /* MRB.CHNS bit */
      unsigned long MRB_DISEL   :1;     /* MRB.DISEL bit */
      unsigned long MRB_DTS     :1;     /* MRB.DTS bit */
      unsigned long MRB_DM      :2;     /* MRB.DM bit */
      unsigned long             :2;
      unsigned long             :16;
    }BIT;
  }MR;
  void * SAR;   /* SAR register */
  void * DAR;   /* DAR register */
  struct{
    unsigned long CRA:16;   /* CRA register */
    unsigned long CRB:16;   /* CRB register */
  }CR;
};

#pragma bit_order
#pragma packoption
```

Note: #pragma unpack is used to specify an alignment count of 4 for the DTC transfer information.

Figure 5.6 Structures and Unions Used in Sample Code

5.7 List of Variables

Table 5.8 lists the global variables, table 5.9 lists the const variables, and table 5.10 lists the enumerated types variables.

Table 5.8 Global of Variables

Type	Variable	Description	Used by Function
st_dtc_full	dtc_tbl	Structure variable for transferring data to TGRA0 to TGRD0 and TGRA4 to TGRD4 of the MTU	dtc_init, fslowup0, fslowdwn0, rslowup0, rslowdwn0
void*	dtc_table[256]	DTC vector table allocated to the address of DTC transfer information dtc_tbl[0] (The start address is set to 0x00000000.)	dtc_init

Table 5.9 const of Variables

Type	Variable	Description	Used by Function
unsigned short	cyctbl_A0	Data tables transferred to TGRs	dtc_init, mtu3_init, fslowup0, fslowdwn0, rslowup0, rslowdwn0
unsigned short	cyctbl_A4		
unsigned short	cyctbl_B0		
unsigned short	cyctbl_B4		
unsigned short	cyctbl_C0		
unsigned short	cyctbl_C4		
unsigned short	cyctbl_D0		
unsigned short	cyctbl_D4		

Table 5.10 Enumerated Types of Variables

Type	Variable	Description	Used by Function
motor_mode_t	nextmode0	Switching among motor control modes F_SLOWUP_MODE : Forward acceleration control F_CONSTANT_MODE : Forward constant control F_SLOWDWN_MODE : Forward deceleration control F_STOP_MODE : Stop control R_SLOWUP_MODE : Reverse acceleration control R_CONSTANT_MODE : Reverse constant control R_SLOWDWN_MODE : Reverse deceleration control R_STOP_MODE : Stop control	main, tgia0_int, cmi0_int
motor_stop_seq_t;	fstop_seq	Forward operation stop transition flag STOP_DTCEND_SEQ : DTC transfer end STOP_MTUCMP_SEQ : Compare match occurrence end	tgia0_int
motor_stop_seq_t;	rstop_seq	Reverse operation stop transition flag STOP_DTCEND_SEQ : DTC transfer end STOP_MTUCMP_SEQ : Compare match occurrence end	tgia0_int
—	—	DTC transfer number (number in chain transfer sequence) DTCTBL_MTU4_TGRD : 1st data transfer DTCTBL_MTU4_TGRA : 2nd data transfer DTCTBL_MTU0_TGRB : 3rd data transfer DTCTBL_MTU0_TGRC : 4th data transfer DTCTBL_MTU4_TGRB : 5th data transfer DTCTBL_MTU4_TGRC : 6th data transfer DTCTBL_MTU0_TGRD : 7th data transfer DTCTBL_MTU0_TGRA : 8th data transfer	dtc_init, rslowdwn0, rslowup0, fslowdwn0, fslowup0

5.8 Description of Data Tables

The period data tables are used to produce the stepping motor excitation output.

The sample application changes the output waveforms when a compare match occurs with the timer general registers (TGR).

The data tables for producing A-phase waveform output are cyctbl_A0 and cyctbl_B0.

The data tables for producing $\overline{A}$-phase waveform output are cyctbl_C0 and cyctbl_D0.

The data tables for producing B-phase waveform output are cyctbl_A4 and cyctbl_B4.

The data tables for producing $\overline{B}$-phase waveform output are cyctbl_C4 and cyctbl_D4.

Note: The figure below shows data tables for the case in which UPTIME = 49. If the UPTIME value is changed, modify the data tables as needed by adding or removing calculation formulas.

```
cyctbl_A0[49] = {(4 * CYCTIME),
                 (4 * CYCTIME) – 1 * (4 * CHGTIME),
                 ...,
                 (4 * CYCTIME) – 48 * (4 * CHGTIME)};

cyctbl_B0[49] = {(2 * CYCTIME) – NON_OVR,
                 (2 * CYCTIME) – 1 * (2 * CHGTIME) – NON_OVR,
                 ...,
                 (2 * CYCTIME) – 48 * (2 * CHGTIME) – NON_OVR};

cyctbl_C0[49] = {(2 * CYCTIME),
                 (2 * CYCTIME) – 1 * (2 * CHGTIME),
                 ...,
                 (2 * CYCTIME) – 48 * (2 * CHGTIME)};

cyctbl_D0[49] = {(4 * CYCTIME) – NON_OVR,
                 (4 * CYCTIME) – 1 * (4 * CHGTIME) – NON_OVR,
                 ...,
                 (4 * CYCTIME) – 48 * (4 * CHGTIME) – NON_OVR};

cyctbl_A4[49] = {CYCTIME,
                 CYCTIME – 1 * CHGTIME,
                 ...,
                 CYCTIME – 48 * CHGTIME};

cyctbl_B4[49] = {(3 * CYCTIME) – NON_OVR,
                 (3 * CYCTIME) – 1 * (3 * CHGTIME) – NON_OVR,
                 ...,
                 (3 * CYCTIME) – 48 * (3 * CHGTIME) – NON_OVR};

cyctbl_C4[49] = {(3 * CYCTIME),
                 (3 * CYCTIME) – 1 * (3 * CHGTIME),
                 ...,
                 (3 * CYCTIME) – 48 * (3 * CHGTIME)};

cyctbl_D4[49] = {CYCTIME – NON_OVR,
                 CYCTIME – 1 * CHGTIME – NON_OVR,
                 ...,
                 CYCTIME – 48 * CHGTIME – NON_OVR};
```

Figure 5.7 Period Data Tables

Creating Four-Phase Output Data Tables for Two-Phase Excitation

The method of creating data tables for changing the speed is described below.

First, determine the value of the timer general register (TGR). This value is represented below as C.

Next, determine the value of the change in the period. This value is represented below as T, and the shoot-through current prevention period value as N.

Figure 5.8 shows the waveforms when the speed is changed.

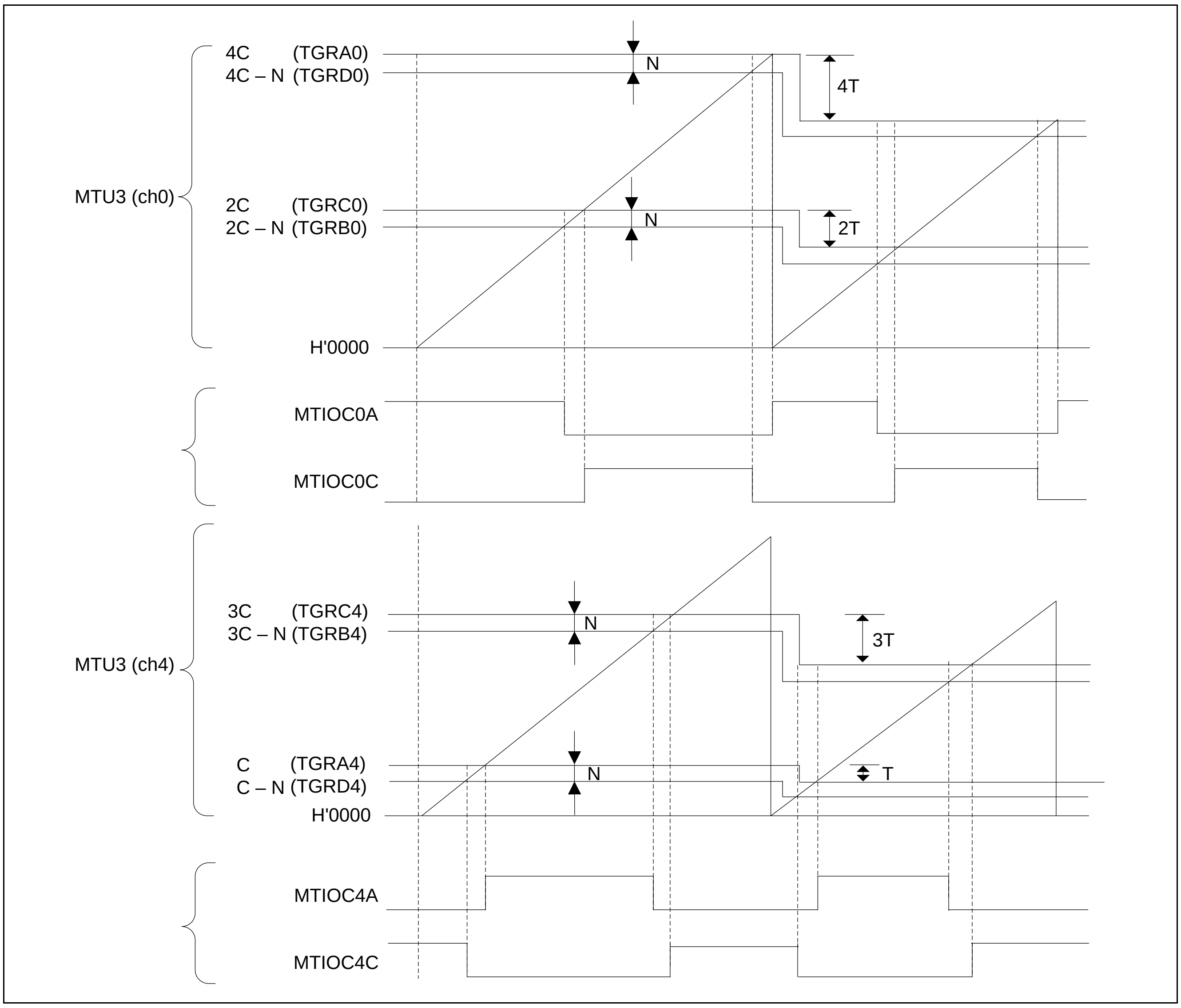


Figure 5.8 Changing the Period Data

Note: The values C, T, N, and n used here are defined in the sample code as CYCTIME, CHGTIME, NON_OVR, and UPTIME, respectively.

The following formulas can be used to create the data tables. The number of array elements is represented as n (n = 0 to 48).

cyctbl_A4[n] = (C – n × T)

cyctbl_C0[n] = (2C – n × 2T)

cyctbl_C4[n] = (3C – n × 3T)

cyctbl_A0[n] = (4C – n × 4T)

Data tables with shoot-through current prevention period inserted

cyctbl_D4[n] = (C – n × T) – N

cyctbl_B0[n] = (2C – n × 2T) – N

cyctbl_B4[n] = (3C – n × 3T) – N

cyctbl_D0[n] = (4C – n × 4T) – N

The following is an example based on the sample application:

First, the compare match value is set to 0x3A98 (D'15000) in cyctbl_A4[0].

If C = 15000, T = 200, and N = 100:

cyctbl_A4[2] = 15000 – 2 × 200 = 14600 (n = 2)

The formula for the data table with shoot-through current prevention period is as follows:

cyctbl_D0[47] = {(4 × 15000) – 47 × (4 × 200)} – 100 = 22300 (n = 47)

The data table is then generated by using the above calculation. Figure 5.9 shows example data tables created for the sample application.

cyctbl_A4[49]

No. of Elements (n)	Hexadecimal Value (C)	Decimal Value (C)	Change Value (T)
0	3A98	15000	—
1	39D0	14800	200
2	3908	14600	200
...	...	...	...
47	15E0	5600	200
48	1518	5400	200

cyctbl_D0[49]

No. of Elements (n)	Hexadecimal Value (C – N)	Decimal Value (C – N)	Change Value (4T)
0	E9FC	59900	—
1	E6DC	59100	800
2	E3BC	58300	800
...	...	...	...
47	571C	22300	800
48	53FC	21500	800

Figure 5.9 Acceleration and Deceleration Data Table Settings

5.9 Function Specification

The following tables list the sample code function specifications.

HardwareSetup	
Overview	Initialization processing
Header	iodefine.h
Declaration	void HardwareSetup(void)
Description	This function makes initial settings. • Initialization for ports that do not exist • System clock and peripheral module clock settings • Module control register settings
Arguments	None
Return values	None

main	
Overview	Main processing
Header	iodefine.h
Declaration	void main(void)
Description	• Specifies forward constant control. • DTC initial settings • MTU3 initial settings • CMT initial settings • MPC initial settings • PMR initial settings • ICU initial settings • Timer count start
Arguments	None
Return values	None

icu_init	
Overview	ICU initial settings
Header	iodefine.h
Declaration	void icu_init(void)
Description	This function makes initial settings to the ICU. • Sets MTU3 TGIA0 as the DTC activation source. • Sets the interrupt priority level of MTU3 TGIA0 and CMI channel 0. • Enables MTU3 TGIA0 in interrupt request enable register.
Arguments	None
Return values	None

dtc_init	
Overview	DTC initial settings
Header	motor.h, iodefine.h
Declaration	void dtc_init(void)
Description	This function makes initial settings to the DTC. • Sets the DTC activation source transfer information address. • Sets the DTC transfer information to forward acceleration control. • Sets the DTC vector base register. • After setting the DTC transfer information, makes DTC module control settings.
Arguments	None
Return values	None

mtu3_init	
Overview	MTU3 initial settings
Header	iodefine.h
Declaration	void mtu3_init(void)
Description	This function makes initial settings to the MTU3. • Sets timer channels 0 and 4 to synchronous operation so TCNT0 and TCNT4 are cleared simultaneously at channel 0 TGRA compare match. • Sets channels 0 and 4 to PWM mode 1 for waveform output. • Sets TIOR for forward acceleration control. • Sets period data table values in TGRA0 to TGRD0 for channel 0 and TGRA4 to TGRD4 for channel 4. • Enables MTU3 module TGIA interrupt and clears timer status flag (TSR) of channel 0 and channel 4.
Arguments	None
Return values	None

cmt0_init	
Overview	CMT0 initial settings
Header	iodefine.h
Declaration	void cmt0_init(void)
Description	This function makes initial settings to the CMT0. • Makes clock settings and enables compare match interrupt.
Arguments	None
Return values	None

mpc_init	
Overview	MPC initial settings
Header	iodefine.h
Declaration	void mpc_init(void)
Description	Selects the following functions in the MPC. • PB3 → MTIOC0A • PB1 → MTIOC0C • P72 → MTIOC4A • P75 → MTIOC4C
Arguments	None
Return values	None

pmr_init

Overview	PMR initial settings
Header	iodefine.h
Declaration	void pmr_init(void)
Description	This function makes initial settings to the PMR. • Sets PB1, PB3, P72, and P75 as peripheral function pins.
Arguments	None
Return values	None

tgia0_int

Overview TGIA0 interrupt

Header motor.h, iodefine.h

Declaration void tgia0_int(void)

Description The TGIA0 interrupt handler ends acceleration control or deceleration control.

- After the transition to the interrupt handler, the TGFA flag of MTU3 channel 0 is cleared.
  - A. If the specified number of DTC transfers have completed, the motor control mode changes.
    Motor control details
    1. When nextmode0 = F_CONSTANT_MODE
       After forward constant control, transition to forward deceleration control
    2. When nextmode0 = F_STOP_MODE
       After stop control, transition to reverse acceleration control
    3. When nextmode0 = R_CONSTANT_MODE
       After reverse constant control, transition to reverse deceleration control
    4. When nextmode0 = R_STOP_MODE
       After stop control, transition to forward acceleration control
  - B. If the specified number of DTC transfers have not completed, the motor control mode does not change.

Arguments None

Return values None

cmi0_int

Overview	CMI0 interrupt
Header	motor.h, iodefine.h
Declaration	void cmi0_int(void)
Description	The CMT interrupt handler ends constant control or stop control.

- Stops CMT0 count operation.
  Motor control mode changes.
  1. When nextmode0 = F_SLOWUP_MODE
     After forward acceleration control, transition to forward constant control
  2. When nextmode0 = F_SLOWDWN_MODE
     After forward deceleration control, transition to stop control
  3. When nextmode0 = R_SLOWUP_MODE
     After reverse acceleration control, transition to reverse constant control
  4. When nextmode0 = R_SLOWDWN_MODE
     After reverse deceleration control, transition to stop control
- Specifies DTC module operation.
- The IR flag of ICU is cleared.
- Enables MTU3 TGIA0 interrupt request enable register.

Arguments	None
Return values	None

fslowup0

Overview	Forward acceleration control
Header	iodefine.h
Declaration	void fslowup0(void)
Description	This function sets the DTC transfer information for forward acceleration control.

- Stops MTU3 operation.
- Makes TIOR settings.
- Updates DTC register transfer information.
- Starts MTU3 operation.

Arguments	None
Return values	None

frconst0

Overview	Forward (reverse) constant control
Header	iodefine.h
Declaration	void frconst0(void)
Description	This function performs constant control.

- Starts CMT operation.
- Disables MTU3 TGIA0 interrupt request enable register.

Arguments	None
Return values	None

fslowdwn0	
Overview	Forward deceleration control
Header	iodefine.h
Declaration	void fslowdwn0(void)
Description	This function sets the DTC transfer information for forward deceleration control. • Updates DTC register transfer information.
Arguments	None
Return values	None

rslowup0	
Overview	Reverse acceleration control
Header	iodefine.h
Declaration	void rslowup0(void)
Description	This function sets the DTC transfer information for forward acceleration control. • Stops MTU3 operation. • Makes TIOR settings. • Updates DTC register transfer information. • Starts MTU3 operation.
Arguments	None
Return values	None

rslowdwn0	
Overview	Reverse deceleration control
Header	iodefine.h
Declaration	void rslowdwn0(void)
Description	This function sets the DTC transfer information for reverse deceleration control. • Updates DTC register transfer information.
Arguments	None
Return values	None

frstop0	
Overview	Stop control
Header	iodefine.h
Declaration	void frstop0(void)
Description	This function performs stop control. • Stops MTU3 operation. • Starts CMT0 operation. • Enables MTU3 TGIA0 interrupt request enable register.
Arguments	None
Return values	None

CmtStart

Overview	CMT operation start
Header	iodefine.h
Declaration	void CmtStart(unsigned short cmt_cnt)
Description	This function starts CMT operation. • Enables CMI0 interrupt request enable register. • Sets CMCOR duration. • Starts CMT0 operation.
Arguments	1st argument: cmt_cnt Constant control duration or stop control duration
Return values	None

CmtStop

Overview	CMT operation stop
Header	iodefine.h
Declaration	void CmtStop(void)
Description	This function stops CMT operation. • Stops CMT0 operation. • Disables CMI0 interrupt request enable register.
Arguments	None
Return values	None

5.10 Flowcharts

5.10.1 Initialization Processing

Figure 5.10 shows flowchart of the initialization processing.

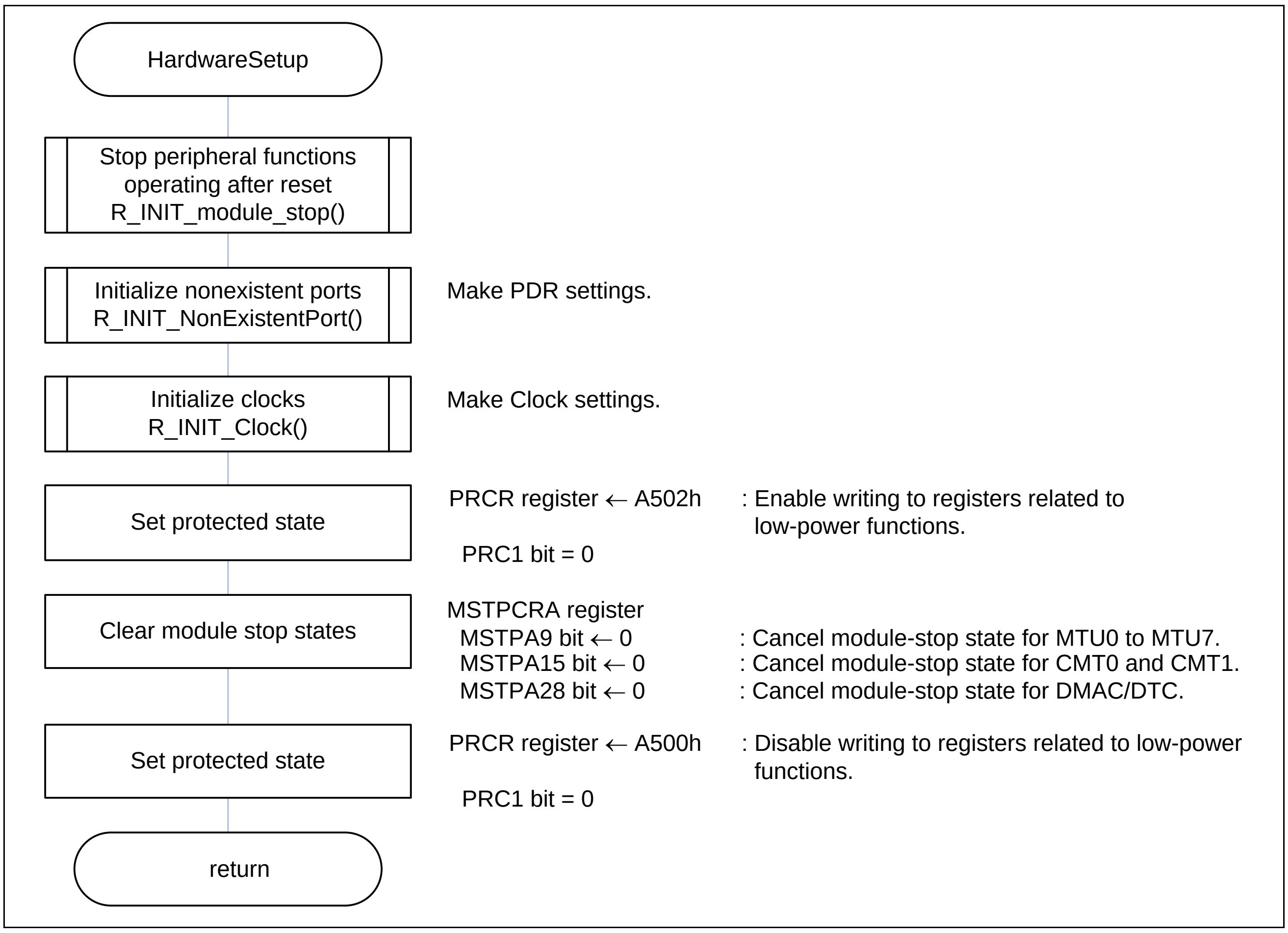

HardwareSetup
Stop peripheral functions operating after reset R_INIT_module_stop()
Initialize nonexistent ports R_INIT_NonExistentPort()
Make PDR settings.
Initialize clocks R_INIT_Clock()
Make Clock settings.
Set protected state
PRCR register ← A502h
PRC1 bit = 0
: Enable writing to registers related to low-power functions.
Clear module stop states
MSTPCRA register
MSTPA9 bit ← 0
MSTPA15 bit ← 0
MSTPA28 bit ← 0
: Cancel module-stop state for MTU0 to MTU7.
: Cancel module-stop state for CMT0 and CMT1.
: Cancel module-stop state for DMAC/DTC.
Set protected state
PRCR register ← A500h
PRC1 bit = 0
: Disable writing to registers related to low-power functions.
return

Figure 5.10 Initialization Processing

5.10.2 Main Processing

Figure 5.11 shows flowchart of the Main processing.

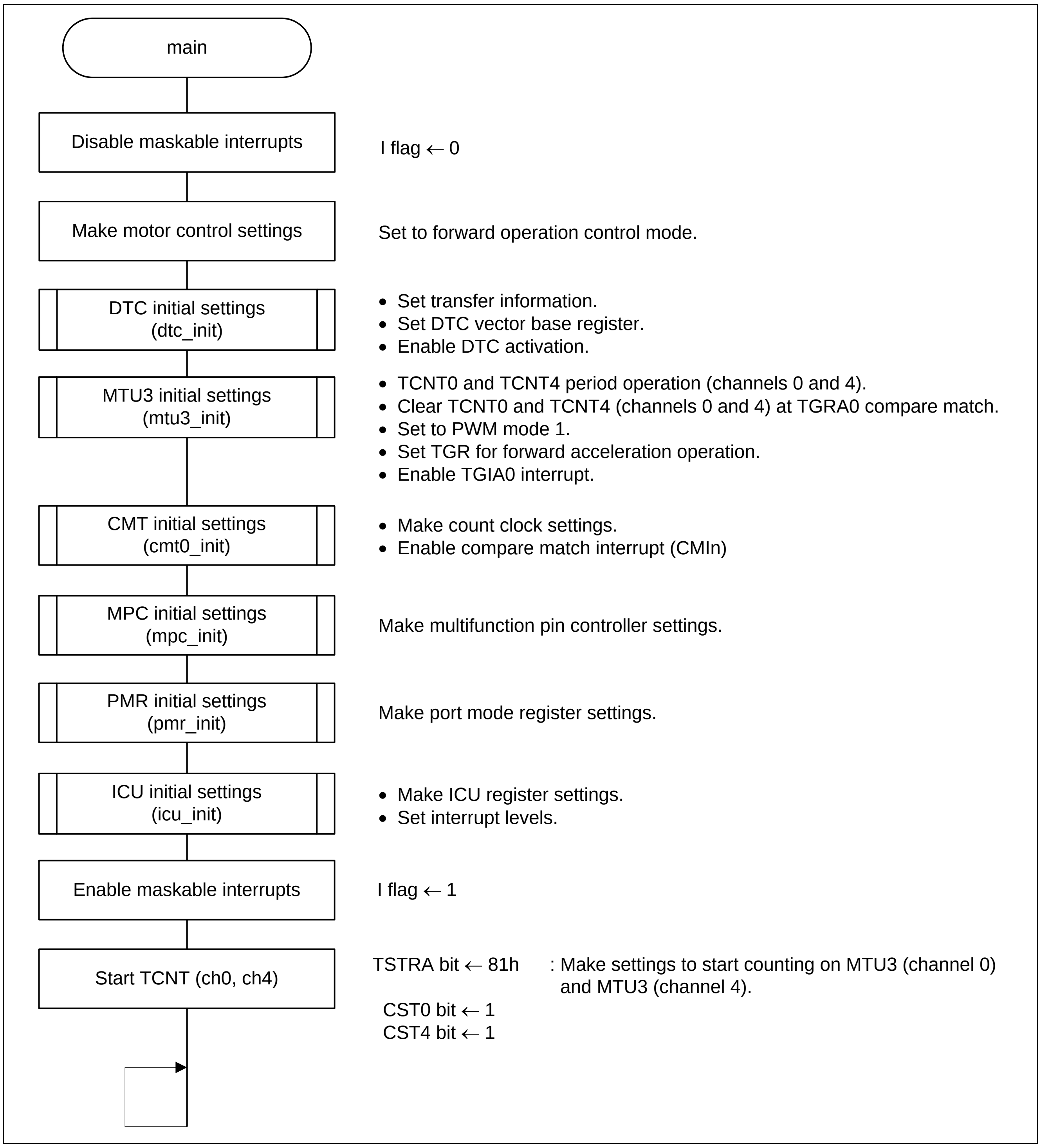

main
Disable maskable interrupts
I flag ← 0
Make motor control settings
Set to forward operation control mode.
DTC initial settings (dtc_init)
• Set transfer information.
• Set DTC vector base register.
• Enable DTC activation.
MTU3 initial settings (mtu3_init)
• TCNT0 and TCNT4 period operation (channels 0 and 4).
• Clear TCNT0 and TCNT4 (channels 0 and 4) at TGRA0 compare match.
• Set to PWM mode 1.
• Set TGR for forward acceleration operation.
• Enable TGIA0 interrupt.
CMT initial settings (cmt0_init)
• Make count clock settings.
• Enable compare match interrupt (CMIn)
MPC initial settings (mpc_init)
Make multifunction pin controller settings.
PMR initial settings (pmr_init)
Make port mode register settings.
ICU initial settings (icu_init)
• Make ICU register settings.
• Set interrupt levels.
Enable maskable interrupts
I flag ← 1
Start TCNT (ch0, ch4)
TSTRA bit ← 81h : Make settings to start counting on MTU3 (channel 0) and MTU3 (channel 4).
CST0 bit ← 1
CST4 bit ← 1

Figure 5.11 Main Processing

5.10.3 ICU Initial Settings

Figure 5.12 shows flowchart of the ICU initial settings.

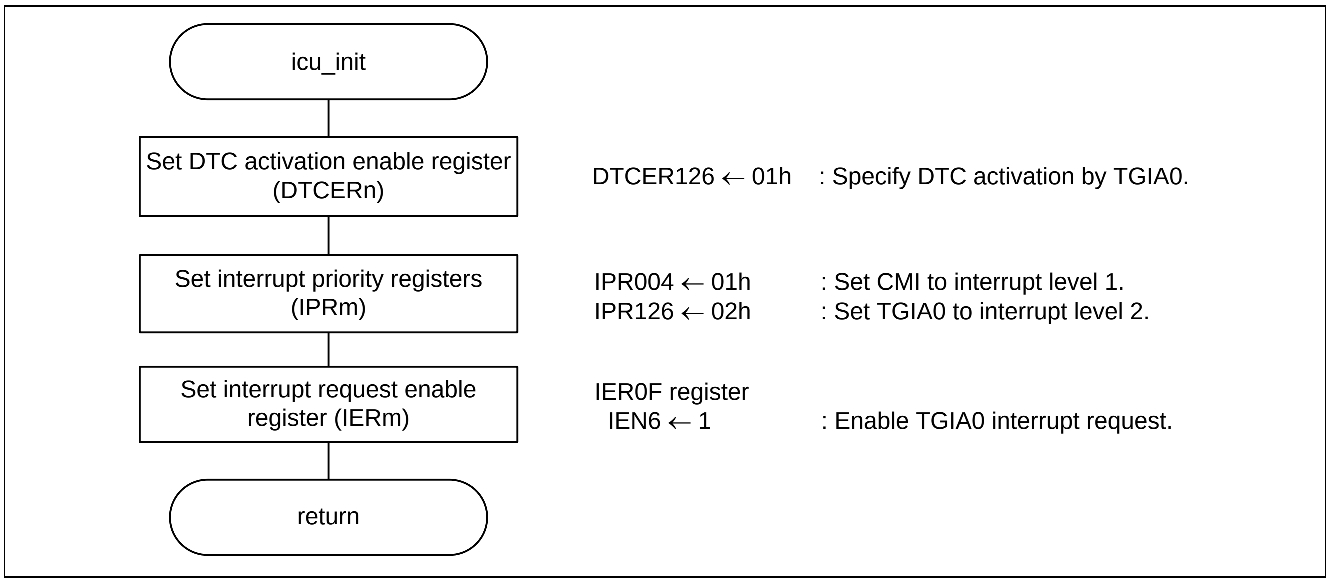

icu_init
Set DTC activation enable register (DTCERn)
DTCER126 ← 01h : Specify DTC activation by TGIA0.
Set interrupt priority registers (IPRm)
IPR004 ← 01h : Set CMI to interrupt level 1.
IPR126 ← 02h : Set TGIA0 to interrupt level 2.
Set interrupt request enable register (IERm)
IER0F register
IEN6 ← 1 : Enable TGIA0 interrupt request.
return

Figure 5.12 ICU Initial Settings

5.10.4 DTC Initial Settings

Figure 5.13 shows flowchart of the DTC initial settings.

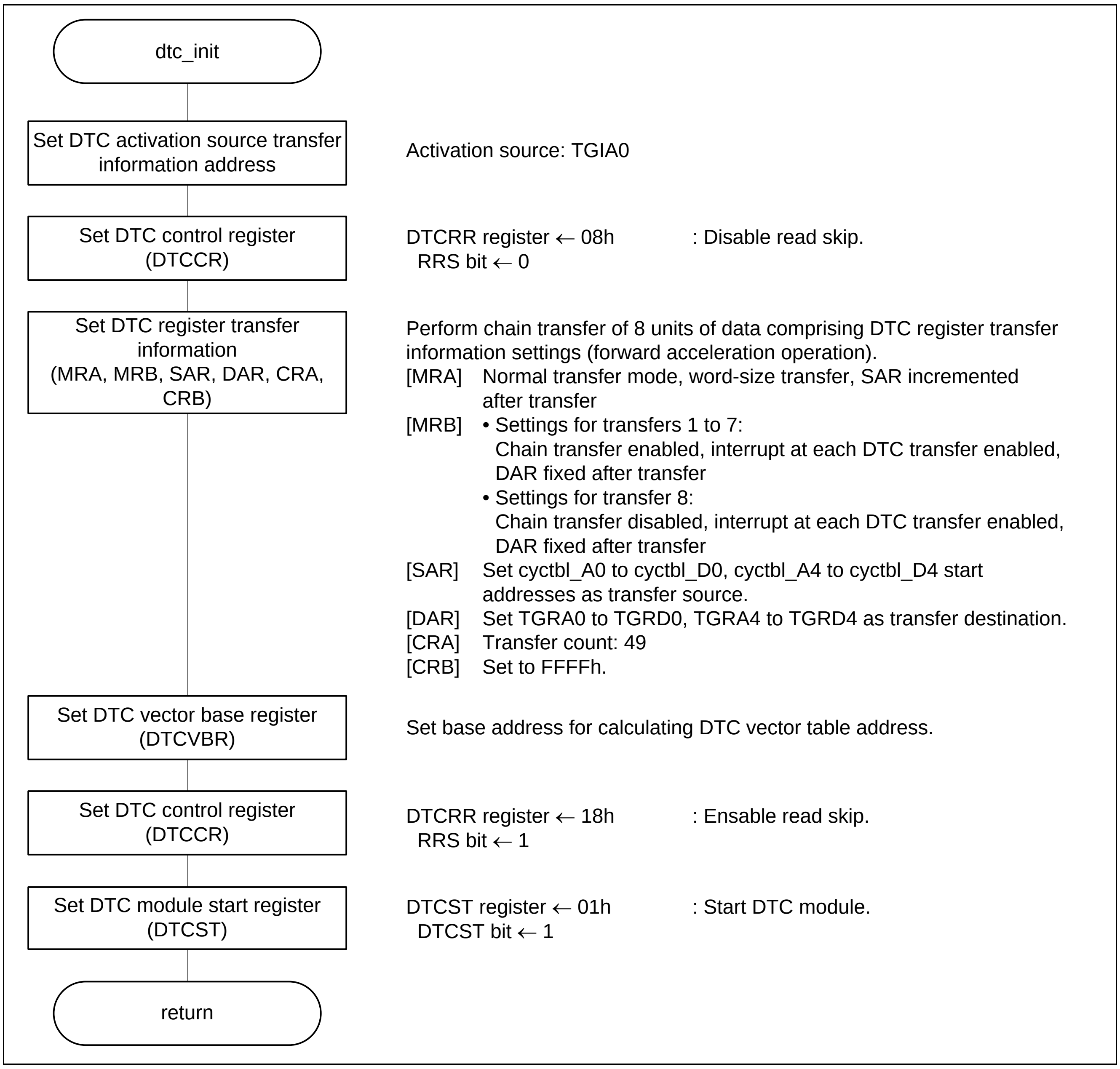
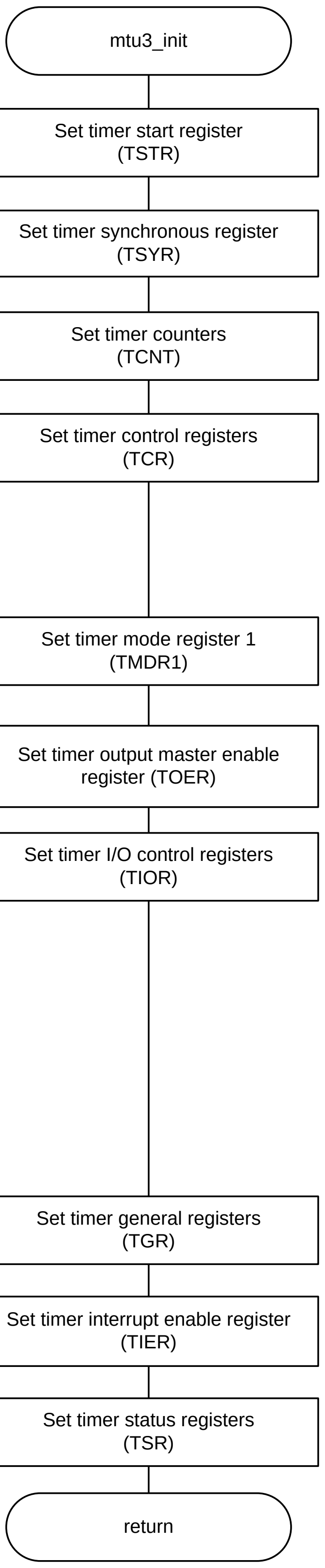

dtc_init
Set DTC activation source transfer information address
Activation source: TGIA0
Set DTC control register (DTCCR)
DTCRR register ← 08h : Disable read skip.
RRS bit ← 0
Set DTC register transfer information (MRA, MRB, SAR, DAR, CRA, CRB)
Perform chain transfer of 8 units of data comprising DTC register transfer information settings (forward acceleration operation).
[MRA] Normal transfer mode, word-size transfer, SAR incremented after transfer
[MRB] • Settings for transfers 1 to 7: Chain transfer enabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
• Settings for transfer 8: Chain transfer disabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
[SAR] Set cyctbl_A0 to cyctbl_D0, cyctbl_A4 to cyctbl_D4 start addresses as transfer source.
[DAR] Set TGRA0 to TGRD0, TGRA4 to TGRD4 as transfer destination.
[CRA] Transfer count: 49
[CRB] Set to FFFFh.
Set DTC vector base register (DTCVBR)
Set base address for calculating DTC vector table address.
Set DTC control register (DTCCR)
DTCRR register ← 18h : Ensable read skip.
RRS bit ← 1
Set DTC module start register (DTCST)
DTCST register ← 01h : Start DTC module.
DTCST bit ← 1
return

Figure 5.13 DTC Initial Settings

5.10.5 MTU3 Initial Settings

Figure 5.14 shows flowchart of the MTU3 initial settings.


mtu3_init
Set timer start register (TSTR)
Set timer synchronous register (TSYR)
Set timer counters (TCNT)
Set timer control registers (TCR)
Set timer mode register 1 (TMDR1)
Set timer output master enable register (TOER)
Set timer I/O control registers (TIOR)
Set timer general registers (TGR)
Set timer interrupt enable register (TIER)
Set timer status registers (TSR)
return

TSTRA register ← 00h	: Stop counting operation on MTU3 (ch0) and MTU3 (ch4).
TSYRA register SYNC0 bit ← 1 SYNC4 bit ← 1	: Specify synchronous operation of MTU3 (ch0) and MTU3 (ch4).

TCNT register
MTU0.TCNT ← 0000h
MTU4.TCNT ← 0000h
	: Clear TCNT of MTU3 (ch0) and MTU3 (ch4).

MTU0.TCR register ← 22h CCLR[2:0] bits ← 001b CKEG[1:0] bits ← 00b TPSC[2:0] bits ← 010b	: Clear TCNT at TGRA0 compare match. Count at internal clock PCLKA/16.
MTU4.TCR register ← E2h CCLR[2:0] bits ← 111b CKEG[1:0] bits ← 00b TPSC[2:0] bits ← 010b	: Clear TCNT when counter of other channel operating synchronously is cleared. Count at internal clock PCLKA/16.
MTU0.TMDR1 register MD[3:0] bits ← 0010b MTU4.TMDR1 register MD[3:0] bits ← 0010b	: Set MTU3 (ch0) and MTU3 (ch4) to PWM mode 1.
TOERA register ← D2h OE4A bit ← 1 OE4C bit ← 1	: Enable MTIOC4A and MTIOC4C output.
MTU0.TIORH register ← 56h IOA[3:0] bits ← 0110b IOB[3:0] bits ← 0101b	: MTIOC0A output is high initially, high at TGRA0 compare match, low at TGRB0 compare match
MTU0.TIORL register ← 12h IOC[3:0] bits ← 0010b IOD[3:0] bits ← 0001b	: MTIOC0C output is low initially, high at TGRC0 compare match, low at TGRD0 compare match.
MTU4.TIORH register ← 16h IOA[3:0] bits ← 0110b IOB[3:0] bits ← 0001b	: MTIOC0A output is high initially, high at TGRA4 compare match, low at TGRB4 compare match.
MTU4.TIORL register ← 52h IOC[3:0] bits ← 0010b IOD[3:0] bits ← 0101b	: MTIOC0C output is low initially, high at TGRC4 compare match, low at TGRD4 compare match.
• Set TGRA0 to TGRD0. • Set TGRA4 to TGRD4.	
MTU0.TIER register TGIEA ← 1	: Enable TGIA0 interrupt requests.
MTU0.TSR register MTU0.TSR ← MTU0.TSR & 0xC0	: Clear MTU3 (ch0) compare match flag.
MTU4.TSR register MTU4.TSR ← MTU4.TSR & 0xC0	: Clear MTU3 (ch4) compare match flag.

Figure 5.14 MTU3 Initial Settings

5.10.6 CMT0 Initial Settings

Figure 5.15 shows flowchart of the CMT0 initial settings.

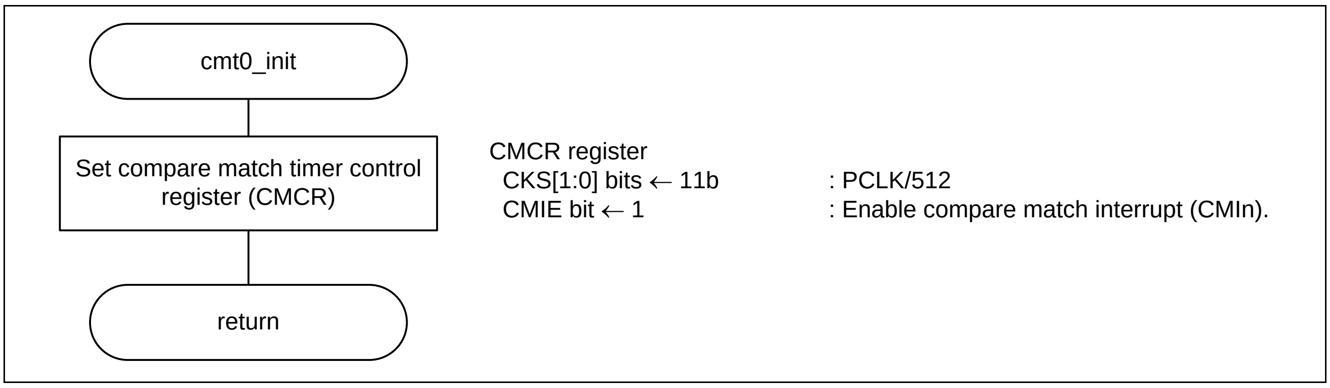

cmt0_init
Set compare match timer control register (CMCR)
return
CMCR register
CKS[1:0] bits ← 11b : PCLK/512
CMIE bit ← 1 : Enable compare match interrupt (CMIn).

Figure 5.15 CMT0 Initial Settings

5.10.7 MPC Initial Settings

Figure 5.16 shows flowchart of the MPC initial settings.

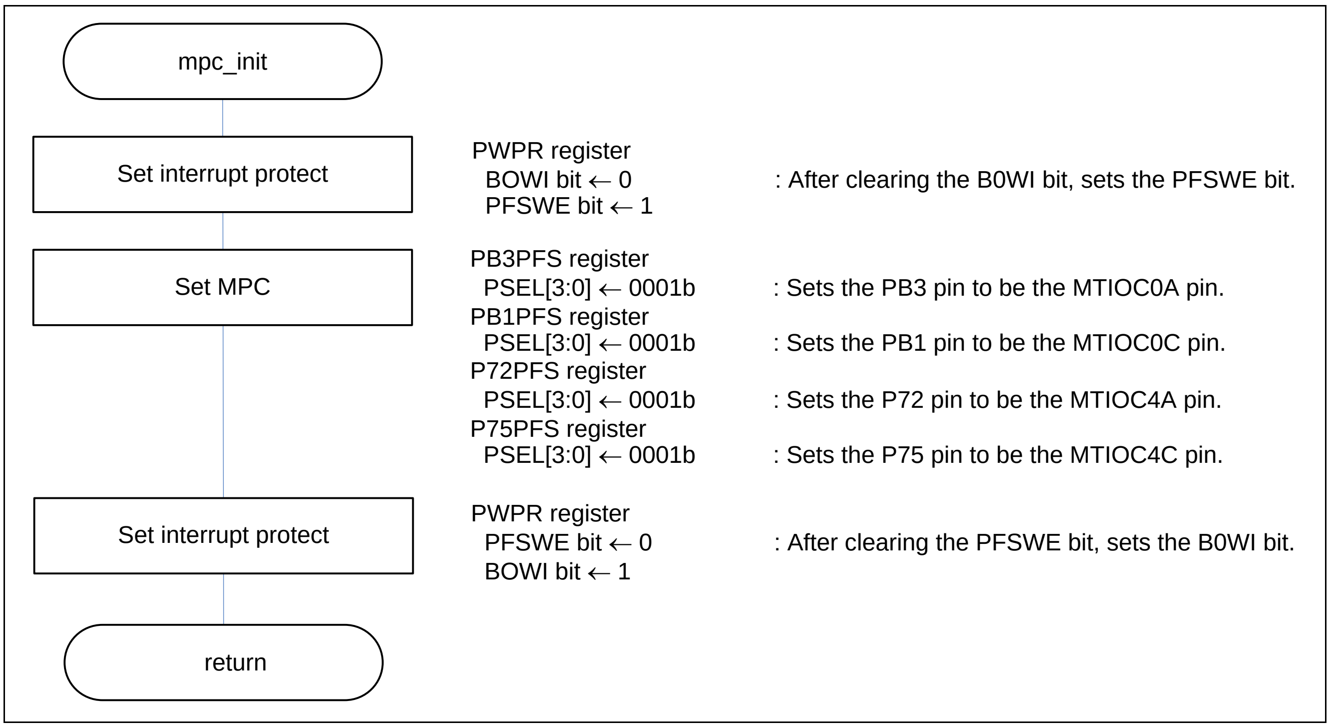

mpc_init
Set interrupt protect
PWPR register
BOWI bit ← 0 : After clearing the B0WI bit, sets the PFSWE bit.
PFSWE bit ← 1
Set MPC
PB3PFS register
PSEL[3:0] ← 0001b : Sets the PB3 pin to be the MTIOC0A pin.
PB1PFS register
PSEL[3:0] ← 0001b : Sets the PB1 pin to be the MTIOC0C pin.
P72PFS register
PSEL[3:0] ← 0001b : Sets the P72 pin to be the MTIOC4A pin.
P75PFS register
PSEL[3:0] ← 0001b : Sets the P75 pin to be the MTIOC4C pin.
Set interrupt protect
PWPR register
PFSWE bit ← 0 : After clearing the PFSWE bit, sets the B0WI bit.
BOWI bit ← 1
return

Figure 5.16 MPC Initial Settings

5.10.8 PMR Initial Settings

Figure 5.17 shows flowchart of the PMR initial settings.

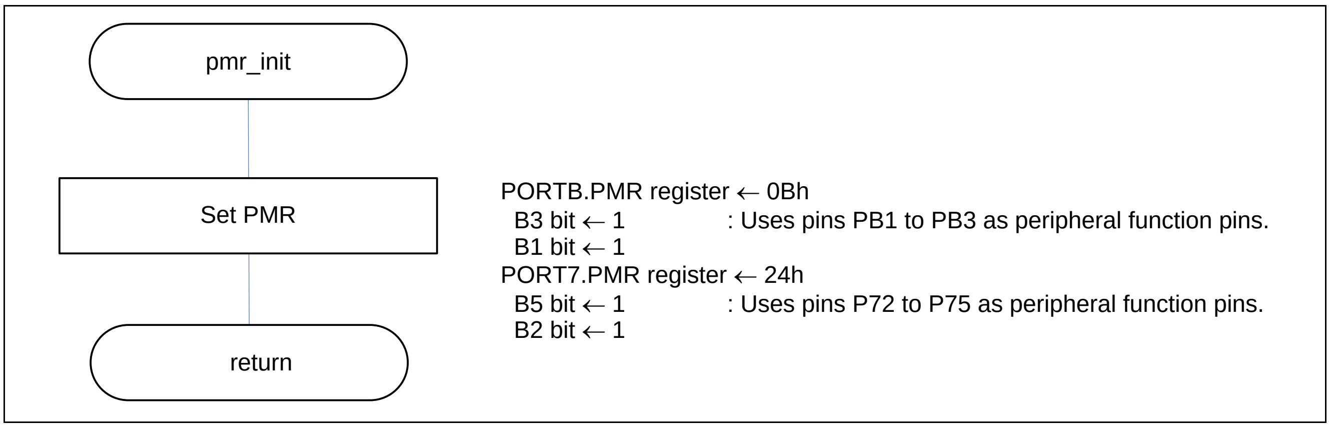

pmr_init
Set PMR
return
PORTB.PMR register ← 0Bh
B3 bit ← 1 : Uses pins PB1 to PB3 as peripheral function pins.
B1 bit ← 1
PORT7.PMR register ← 24h
B5 bit ← 1 : Uses pins P72 to P75 as peripheral function pins.
B2 bit ← 1

Figure 5.17 PMR Initial Settings

5.10.9 TGIA0 Interrupt Handler

Figure 5.18 shows flowchart of the TGIA0 interrupt handler.

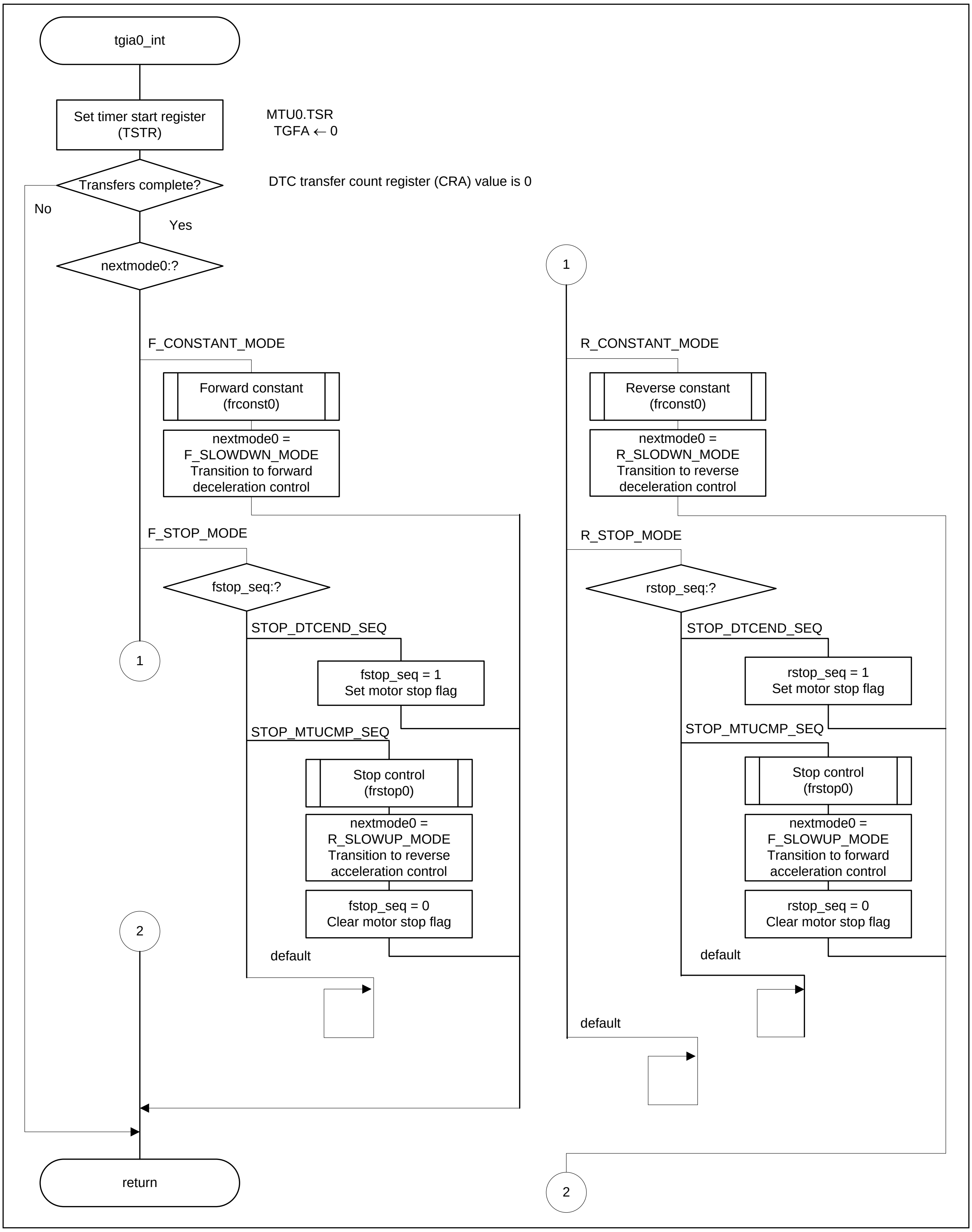

tgia0_int
Set timer start register (TSTR)
MTU0.TSR
TGFA ← 0
Transfers complete?
DTC transfer count register (CRA) value is 0
No
Yes
nextmode0:?
F_CONSTANT_MODE
Forward constant (frconst0)
nextmode0 = F_SLOWDWN_MODE Transition to forward deceleration control
F_STOP_MODE
fstop_seq:?
STOP_DTCEND_SEQ
fstop_seq = 1 Set motor stop flag
STOP_MTUCMP_SEQ
Stop control (frstop0)
nextmode0 = R_SLOWUP_MODE Transition to reverse acceleration control
fstop_seq = 0 Clear motor stop flag
default
1
2
return
1
R_CONSTANT_MODE
Reverse constant (frconst0)
nextmode0 = R_SLODWN_MODE Transition to reverse deceleration control
R_STOP_MODE
rstop_seq:?
STOP_DTCEND_SEQ
rstop_seq = 1 Set motor stop flag
STOP_MTUCMP_SEQ
Stop control (frstop0)
nextmode0 = F_SLOWUP_MODE Transition to forward acceleration control
rstop_seq = 0 Clear motor stop flag
default
default
2

Figure 5.18 TGIA0 Interrupt

5.10.10 CMI0 Interrupt

Figure 5.19 shows flowchart of the CMI0 interrupt.

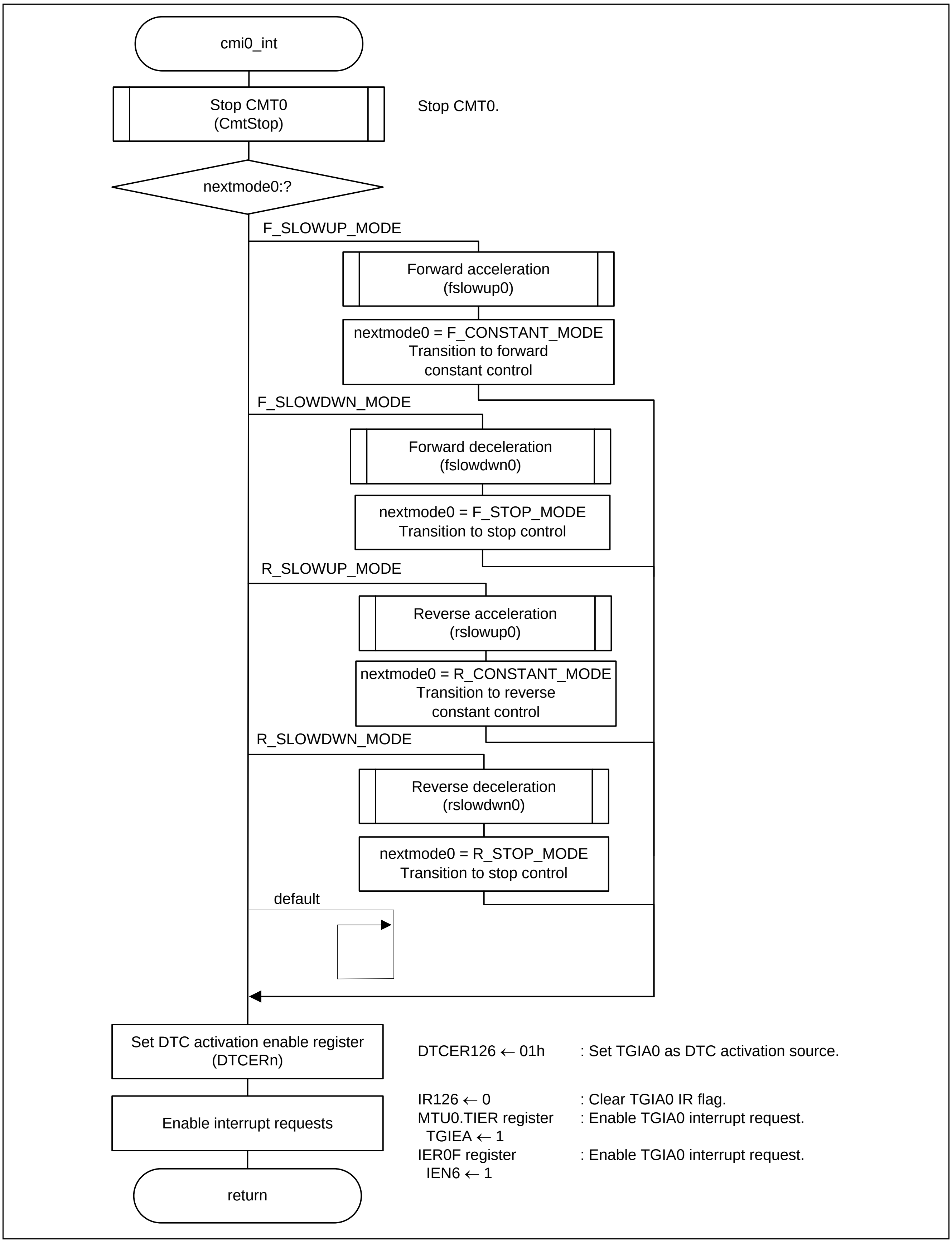
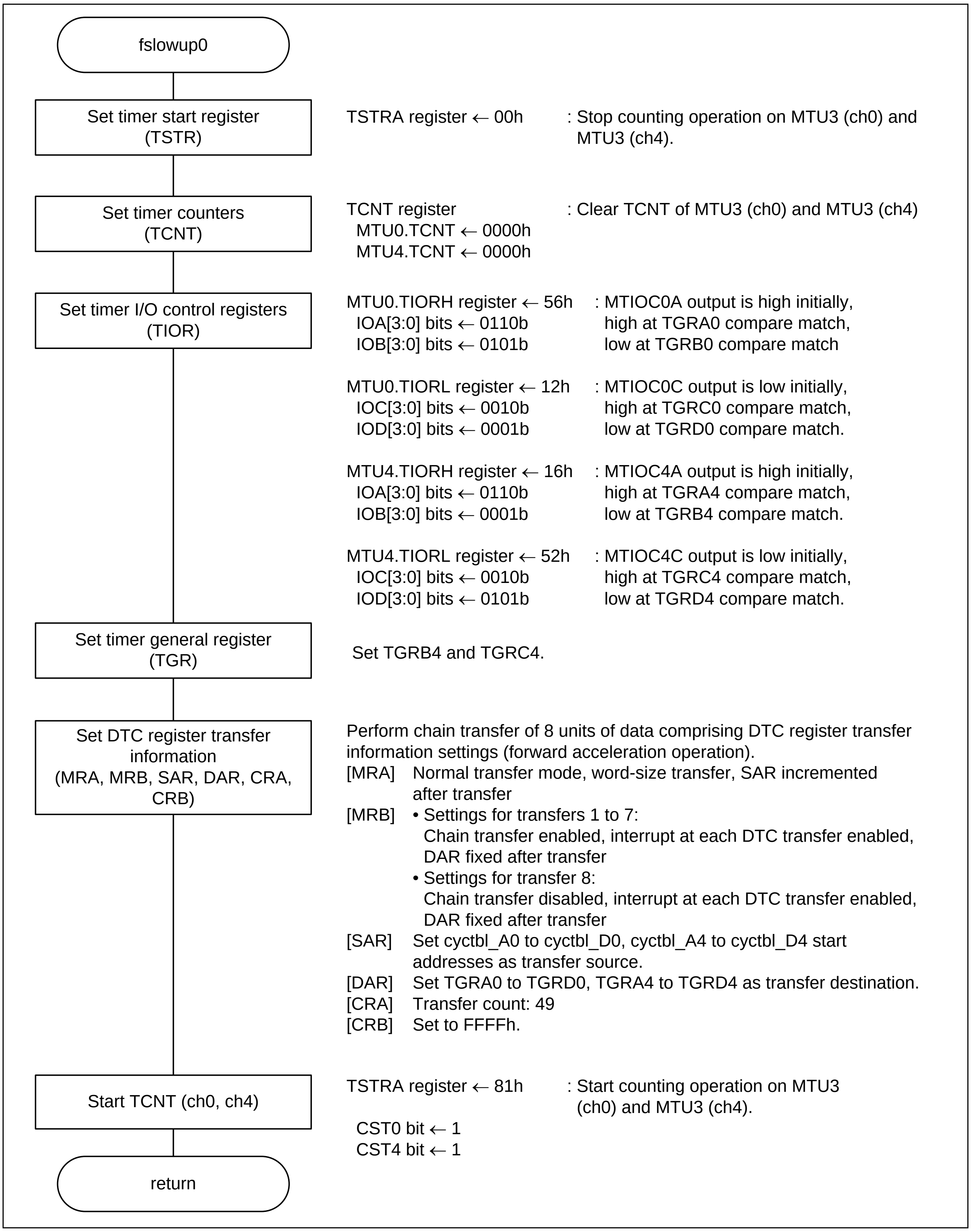

cmi0_int
Stop CMT0 (CmtStop)
Stop CMT0.
nextmode0:?
F_SLOWUP_MODE
Forward acceleration (fslowup0)
nextmode0 = F_CONSTANT_MODE Transition to forward constant control
F_SLOWDWN_MODE
Forward deceleration (fslowdwn0)
nextmode0 = F_STOP_MODE Transition to stop control
R_SLOWUP_MODE
Reverse acceleration (rslowup0)
nextmode0 = R_CONSTANT_MODE Transition to reverse constant control
R_SLOWDWN_MODE
Reverse deceleration (rslowdwn0)
nextmode0 = R_STOP_MODE Transition to stop control
default
Set DTC activation enable register (DTCERn)
DTCER126 ← 01h : Set TGIA0 as DTC activation source.
Enable interrupt requests
IR126 ← 0 : Clear TGIA0 IR flag.
MTU0.TIER register TGIEA ← 1 : Enable TGIA0 interrupt request.
IER0F register IEN6 ← 1 : Enable TGIA0 interrupt request.
return

Figure 5.19 CMI0 Interrupt

5.10.11 Forward Acceleration Processing

Figure 5.20 shows flowchart of the forward acceleration processing.



fslowup0
Set timer start register (TSTR)
TSTRA register ← 00h
: Stop counting operation on MTU3 (ch0) and MTU3 (ch4).
Set timer counters (TCNT)
TCNT register
MTU0.TCNT ← 0000h
MTU4.TCNT ← 0000h
: Clear TCNT of MTU3 (ch0) and MTU3 (ch4)
Set timer I/O control registers (TIOR)
MTU0.TIORH register ← 56h
IOA[3:0] bits ← 0110b
IOB[3:0] bits ← 0101b
: MTIOC0A output is high initially, high at TGRA0 compare match, low at TGRB0 compare match
MTU0.TIORL register ← 12h
IOC[3:0] bits ← 0010b
IOD[3:0] bits ← 0001b
: MTIOC0C output is low initially, high at TGRC0 compare match, low at TGRD0 compare match.
MTU4.TIORH register ← 16h
IOA[3:0] bits ← 0110b
IOB[3:0] bits ← 0001b
: MTIOC4A output is high initially, high at TGRA4 compare match, low at TGRB4 compare match.
MTU4.TIORL register ← 52h
IOC[3:0] bits ← 0010b
IOD[3:0] bits ← 0101b
: MTIOC4C output is low initially, high at TGRC4 compare match, low at TGRD4 compare match.
Set timer general register (TGR)
Set TGRB4 and TGRC4.
Set DTC register transfer information (MRA, MRB, SAR, DAR, CRA, CRB)
Perform chain transfer of 8 units of data comprising DTC register transfer information settings (forward acceleration operation).
[MRA] Normal transfer mode, word-size transfer, SAR incremented after transfer
[MRB] • Settings for transfers 1 to 7: Chain transfer enabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
• Settings for transfer 8: Chain transfer disabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
[SAR] Set cyctbl_A0 to cyctbl_D0, cyctbl_A4 to cyctbl_D4 start addresses as transfer source.
[DAR] Set TGRA0 to TGRD0, TGRA4 to TGRD4 as transfer destination.
[CRA] Transfer count: 49
[CRB] Set to FFFFh.
Start TCNT (ch0, ch4)
TSTRA register ← 81h
CST0 bit ← 1
CST4 bit ← 1
: Start counting operation on MTU3 (ch0) and MTU3 (ch4).
return

Figure 5.20 Forward Acceleration Processing

5.10.12 Forward/Reverse Constant Processing

Figure 5.21 shows flowchart of the forward/reverse constant processing.

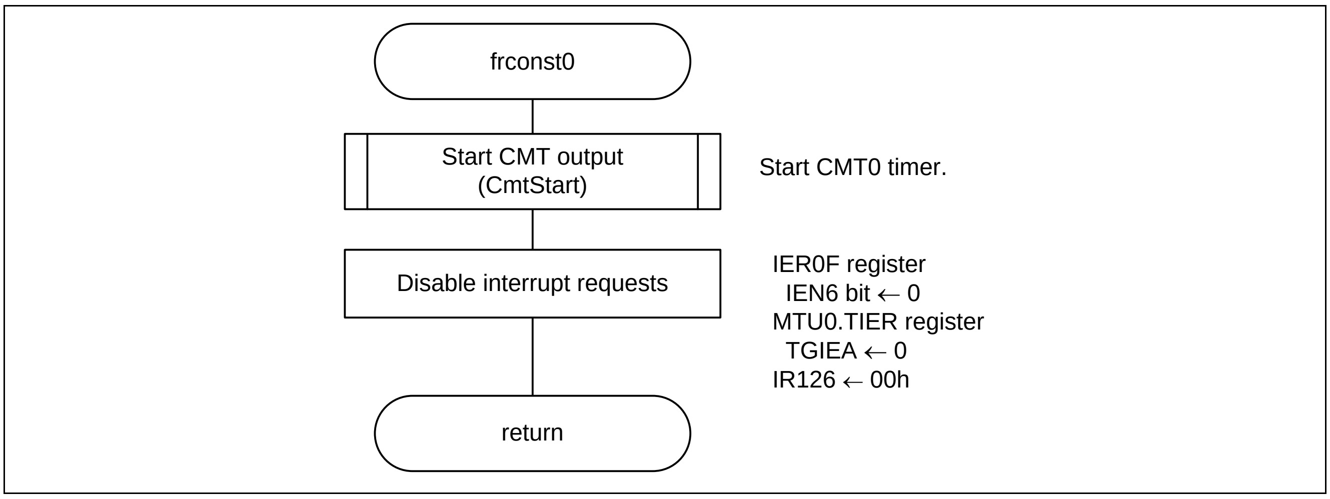
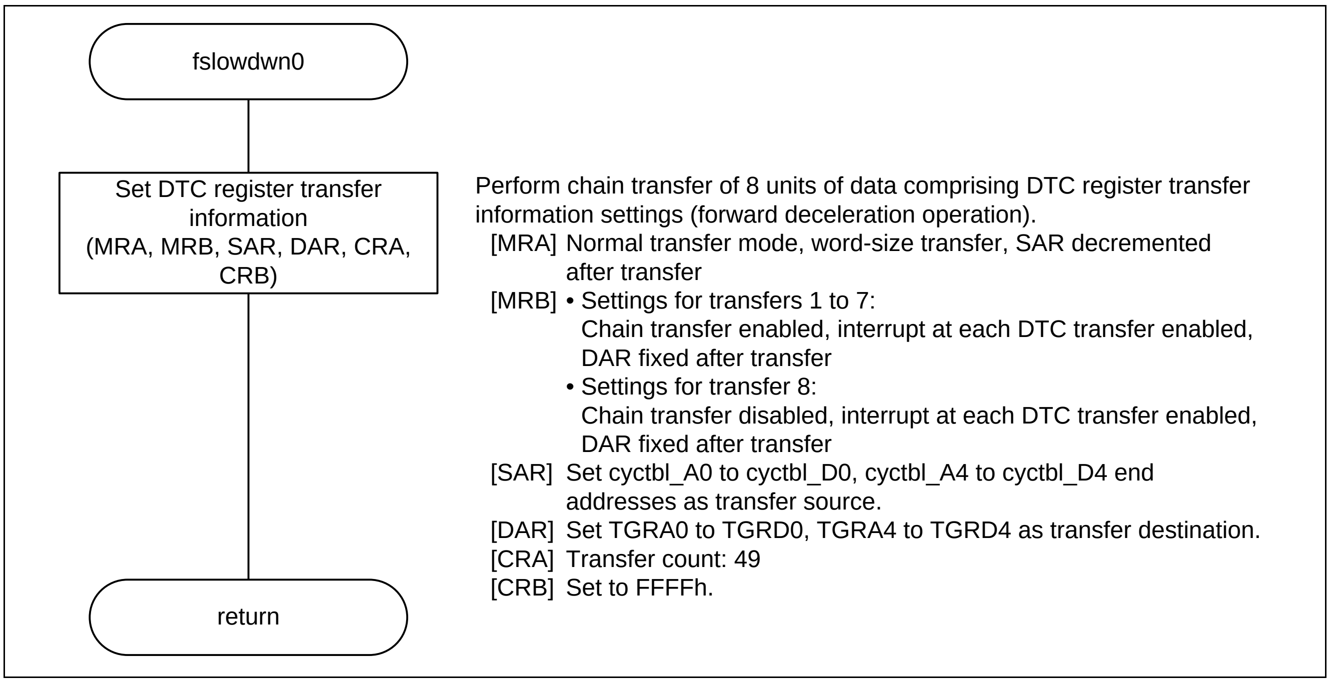


Figure 5.21 Forward/Reverse Constant Processing

5.10.13 Forward Deceleration Processing

Figure 5.22 shows flowchart of the forward deceleration processing.


fslowdwn0
Set DTC register transfer information (MRA, MRB, SAR, DAR, CRA, CRB)
return

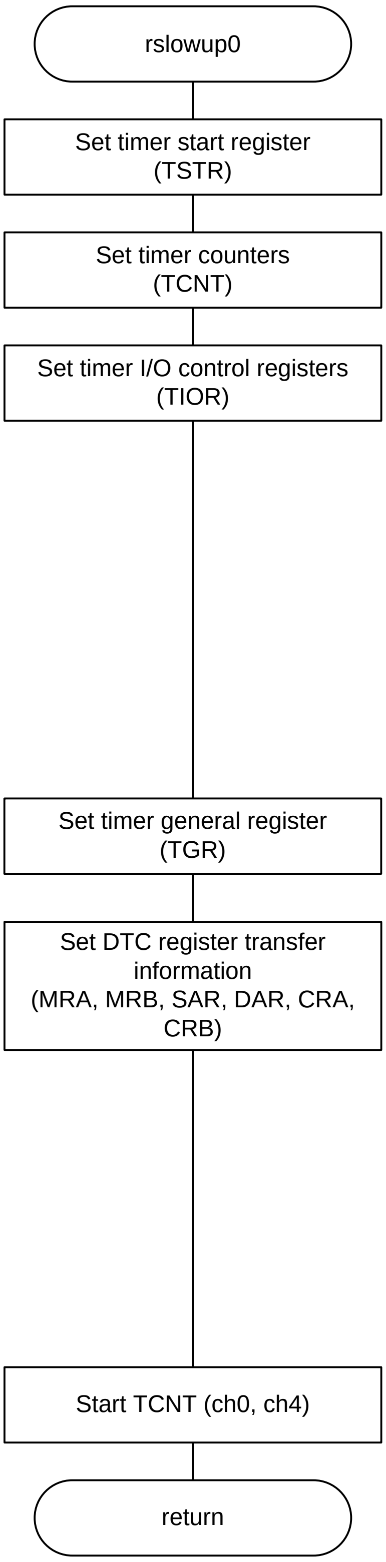
Perform chain transfer of 8 units of data comprising DTC register transfer information settings (forward deceleration operation).

[MRA]	Normal transfer mode, word-size transfer, SAR decremented after transfer
[MRB]	• Settings for transfers 1 to 7: Chain transfer enabled, interrupt at each DTC transfer enabled, DAR fixed after transfer • Settings for transfer 8: Chain transfer disabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
[SAR]	Set cyctbl_A0 to cyctbl_D0, cyctbl_A4 to cyctbl_D4 end addresses as transfer source.
[DAR]	Set TGRA0 to TGRD0, TGRA4 to TGRD4 as transfer destination.
[CRA]	Transfer count: 49
[CRB]	Set to FFFFh.

Figure 5.22 Forward Deceleration Processing

5.10.14 Reverse Acceleration Processing

Figure 5.23 shows flowchart of the reverse acceleration processing.


rslowup0
Set timer start register (TSTR)
Set timer counters (TCNT)
Set timer I/O control registers (TIOR)
Set timer general register (TGR)
Set DTC register transfer information (MRA, MRB, SAR, DAR, CRA, CRB)
Start TCNT (ch0, ch4)
return

TSTRA register ← 00h : Stop counting operation on MTU3 (ch0) and MTU3 (ch4).

TCNT register : Clear TCNT of MTU3 (ch0) and MTU3 (ch4)
MTU0.TCNT ← 0000h
MTU4.TCNT ← 0000h

MTU0.TIORH register ← 56h : MTIOC0A output is high initially, high at TGRA0 compare match, low at TGRB0 compare match
IOA[3:0] bits ← 0110b
IOB[3:0] bits ← 0101b

MTU0.TIORL register ← 12h : MTIOC0C output is low initially, high at TGRC0 compare match, low at TGRD0 compare match.
IOC[3:0] bits ← 0010b
IOD[3:0] bits ← 0001b

MTU4.TIORH register ← 61h : MTIOC0A output is low initially, low at TGRA4 compare match, high at TGRB4 compare match.
IOA[3:0] bits ← 0001b
IOB[3:0] bits ← 0110b

MTU4.TIORL register ← 25h : MTIOC4C output is high initially, low at TGRC4 compare match, high at TGRD4 compare match.
IOC[3:0] bits ← 0101b
IOD[3:0] bits ← 0010b

Set TGRB4 and TGRC4.

Perform chain transfer of 8 units of data comprising DTC register transfer information settings (forward acceleration operation).
[MRA] Normal transfer mode, word-size transfer, SAR incremented after transfer
[MRB] • Settings for transfers 1 to 7:
Chain transfer enabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
• Settings for transfer 8:
Chain transfer disabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
[SAR] Set cyctbl_A0 to cyctbl_D0, cyctbl_A4 to cyctbl_D4 start addresses as transfer source.
[DAR] Set TGRA0 to TGRD0, TGRA4 to TGRD4 as transfer destination.
[CRA] Transfer count: 49
[CRB] Set to FFFFh.

TSTRA register ← 81h : Start counting operation on MTU3 (ch0) and MTU3 (ch4).
CST0 bit ← 1
CST4 bit ← 1

Figure 5.23 Reverse Acceleration Processing

5.10.15 Reverse Deceleration Processing

Figure 5.24 shows flowchart of the reverse deceleration processing.

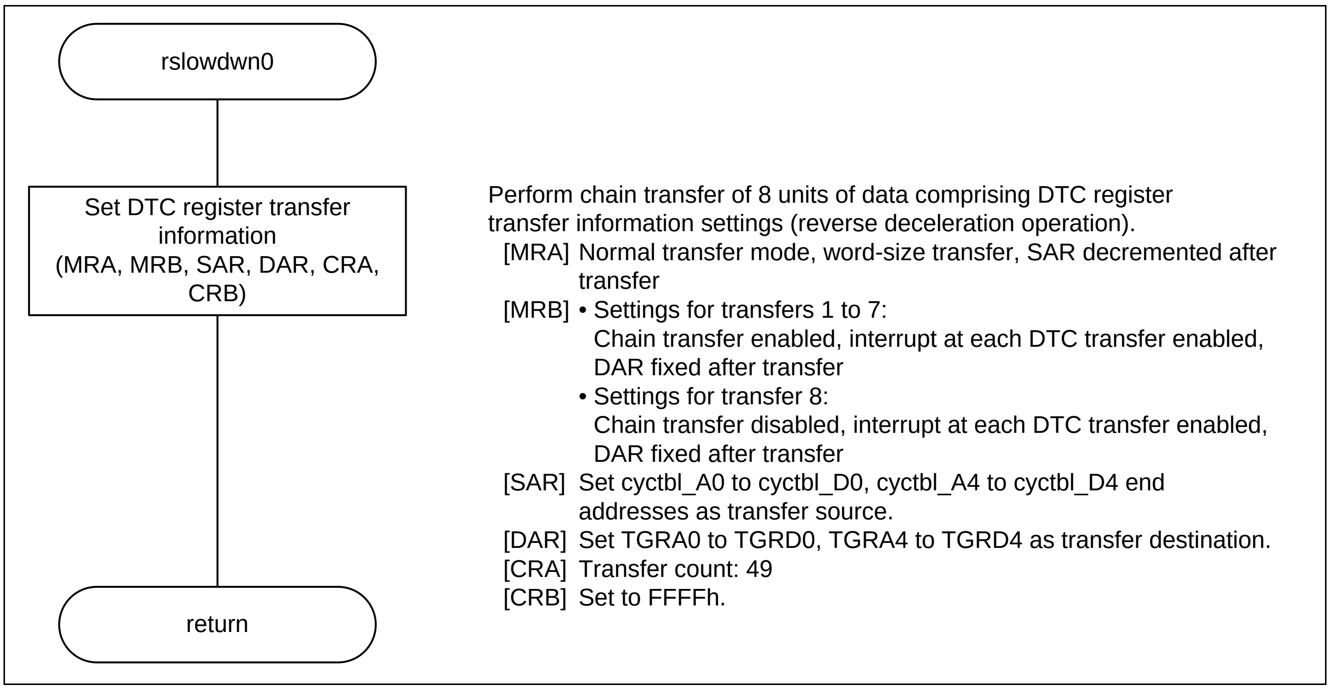

rslowdwn0
Set DTC register transfer information (MRA, MRB, SAR, DAR, CRA, CRB)
return
Perform chain transfer of 8 units of data comprising DTC register transfer information settings (reverse deceleration operation).
[MRA] Normal transfer mode, word-size transfer, SAR decremented after transfer
[MRB] • Settings for transfers 1 to 7: Chain transfer enabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
• Settings for transfer 8: Chain transfer disabled, interrupt at each DTC transfer enabled, DAR fixed after transfer
[SAR] Set cyctbl_A0 to cyctbl_D0, cyctbl_A4 to cyctbl_D4 end addresses as transfer source.
[DAR] Set TGRA0 to TGRD0, TGRA4 to TGRD4 as transfer destination.
[CRA] Transfer count: 49
[CRB] Set to FFFFh.

Figure 5.24 Reverse Deceleration Processing

5.10.16 Stop Processing

Figure 5.25 shows flowchart of the stop processing.

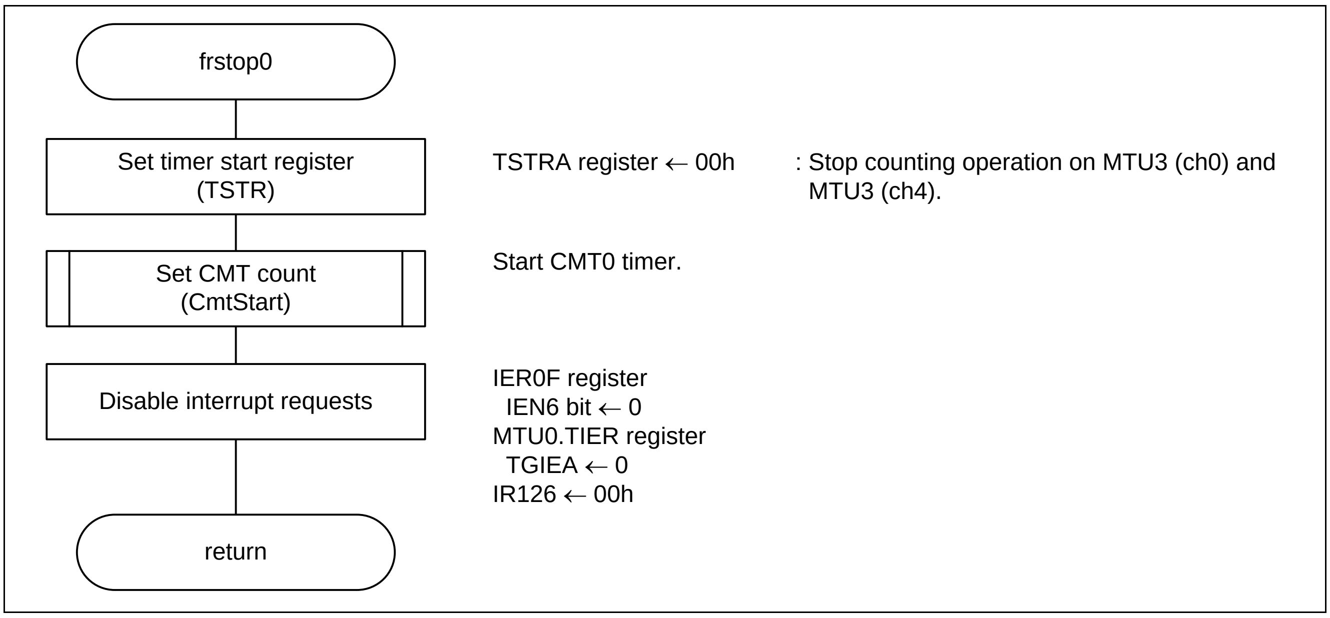

frstop0
Set timer start register (TSTR)
TSTRA register ← 00h
: Stop counting operation on MTU3 (ch0) and MTU3 (ch4).
Set CMT count (CmtStart)
Start CMT0 timer.
Disable interrupt requests
IER0F register IEN6 bit ← 0
MTU0.TIER register TGIEA ← 0
IR126 ← 00h
return

Figure 5.25 Stop Processing

5.10.17 CMT Start Processing

Figure 5.26 shows flowchart of the CMT start processing.

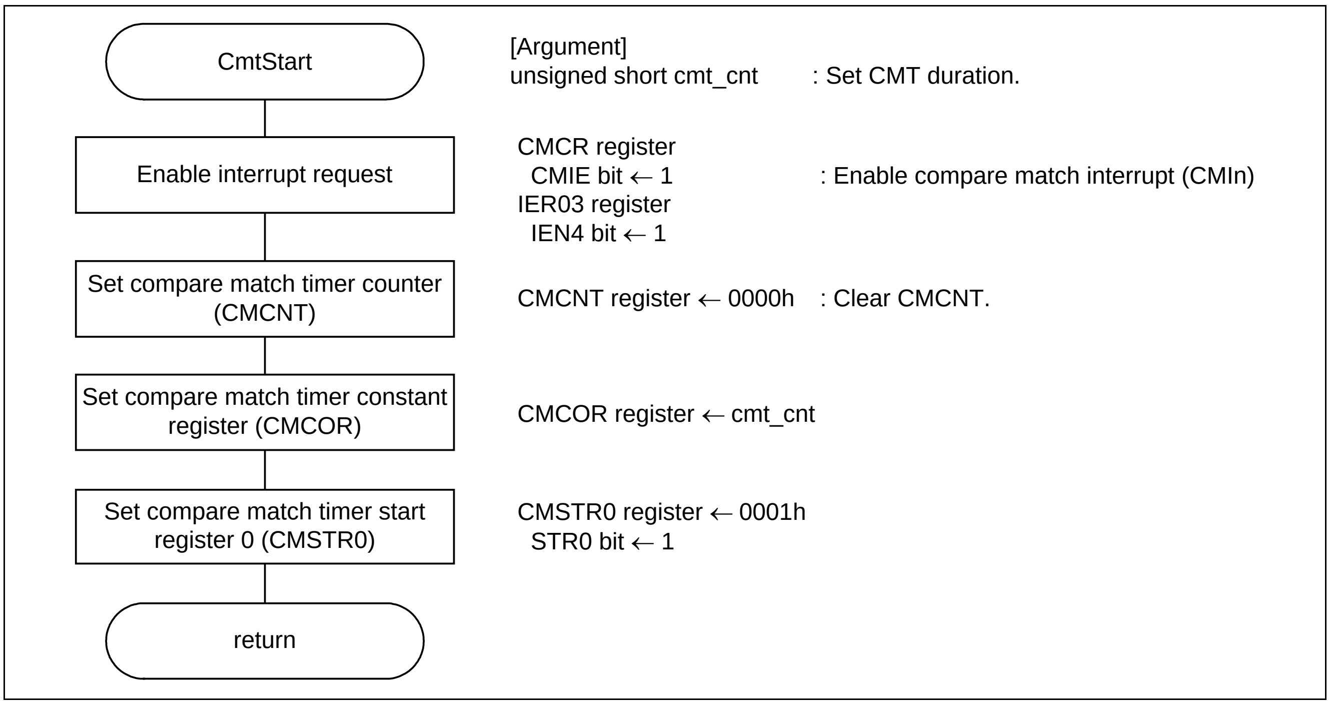

CmtStart
Enable interrupt request
Set compare match timer counter (CMCNT)
Set compare match timer constant register (CMCOR)
Set compare match timer start register 0 (CMSTR0)
return
[Argument]
unsigned short cmt_cnt : Set CMT duration.
CMCR register
CMIE bit ← 1 : Enable compare match interrupt (CMIn)
IER03 register
IEN4 bit ← 1
CMCNT register ← 0000h : Clear CMCNT.
CMCOR register ← cmt_cnt
CMSTR0 register ← 0001h
STR0 bit ← 1

Figure 5.26 CMT Start Processing

5.10.18 CMT Stop Processing

Figure 5.27 shows flowchart of the CMT stop processing.

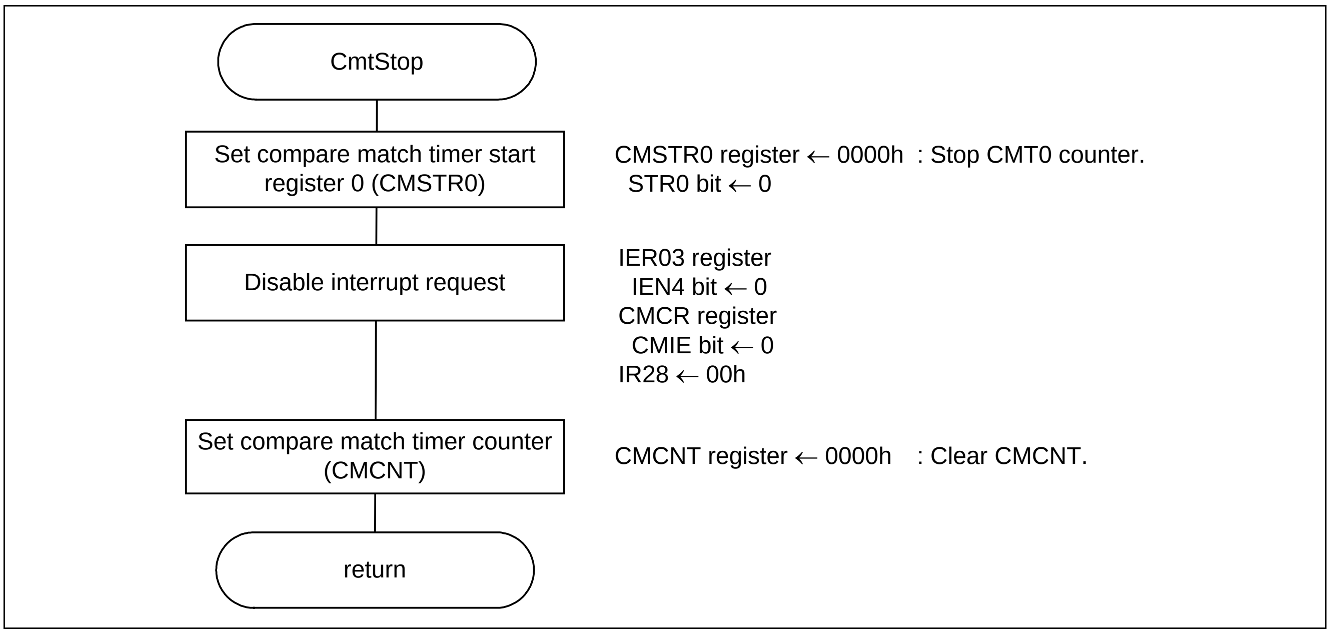

CmtStop
Set compare match timer start register 0 (CMSTR0)
Disable interrupt request
Set compare match timer counter (CMCNT)
return
CMSTR0 register ← 0000h : Stop CMT0 counter.
STR0 bit ← 0
IER03 register
IEN4 bit ← 0
CMCR register
CMIE bit ← 0
IR28 ← 00h
CMCNT register ← 0000h : Clear CMCNT.

Figure 5.27 CMT Stop Processing

6. Notes

6.1 Operating Mode Settings

The sample application sets mode pins MD = high to select single-chip mode as the operating mode and sets the ROME bit to 1 in system control register 0 (SYSCR0) to enable the on-chip ROM.

Table 6.1 lists the operating mode settings of the sample application.

Table 6.1 Operating Mode Settings

Mode Setting Pin	SYSCR0 Register		
MD	**ROME**	**Operating Mode**	**On-chip ROM**
High	1	Single-chip mode	Enabled

Note: The initial value of the ROME bit in the SYSCR0 is 1, so the program does not make settings to the SYSCR0 register.

6.2 Endian

The sample code provided with this application note supports both little endian and big endian operation.

6.2.1 When Using Little Endian

Make the following settings for operation in little-endian mode.

Set the endian setting in the compiler options to "Little endian." The MDES setting in 5.4, Option-Setting Memory, is the value for little-endian.

6.2.2 When Using Big Endian

Make the following settings for operation in big-endian mode.

Set the endian setting in the compiler options to "Big endian." The MDES setting in 5.4, Option-Setting Memory, is the value for big-endian.

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User’s Manual: Hardware

RX63T Group User’s Manual: Hardware Rev.2.00
(The latest version can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News

(The latest version can be downloaded from the Renesas Electronics website.)

User’s Manual: Development Tools

RX Family C/C++ Compiler Package V.1.01 User’s Manual Rev.1.00
(Including the documentation included with V.1.0.2)
(The latest version can be downloaded from the Renesas Electronics website.)

Application Note

RX63T Group Initialization Example Rev.1.00 (R01AN1252EJ0100)
(The latest version can be downloaded from the Renesas Electronics website.)

Website and Support

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

Inquiries
http://www.renesas.com/contact/

Revision History

		Description	
Rev.	Date	Page	Summary
1.00	May. 14, 2014	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different type number, confirm that the change will not lead to problems.
   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 LanGao Rd., Putuo District, Shanghai, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141