RX62T Group

Stepping Motor Employing Two-Phase Excitation

R01AN0894EJ0100
Rev.1.00
Mar 28, 2012

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 RX62T Group.

Target Device

RX62T 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 RX62T 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 measured 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 $\overline{B}$-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, MTIOC3A, MTIOC0C, and MTIOC3C.

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

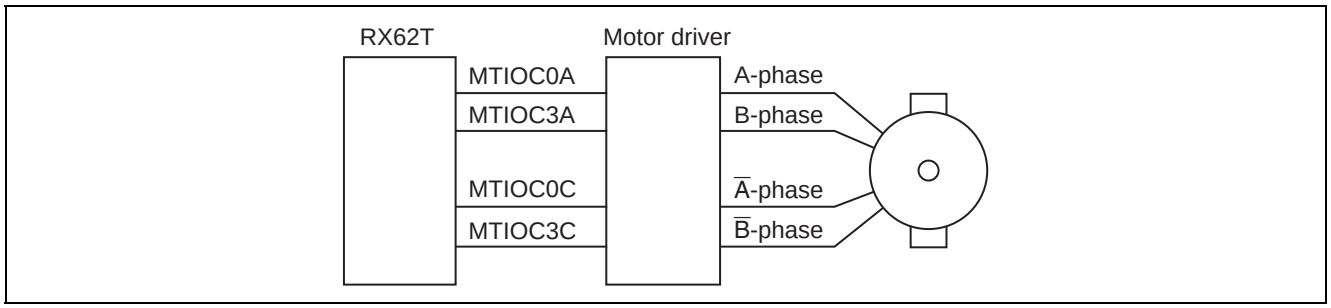

Figure 1.1 Wiring Diagram for Two-Phase Stepping Motor Control

2. Operation Confirmation Conditions

The sample code described in this application note has been confirmed to run normally under the operating conditions given below.

Table 2.1 Operating Conditions

Item	Description
MCU	RX62T Group (R5F562TAADFP)
Operating frequencies	EXTAL: 12.5 MHz ICLK: 100 MHz PCLK: 50 MHz
Operating voltage	5.0 V
Integrated development environment	Renesas Electronics High-performance embedded Workshop, Version 4.09.00.007
C compiler	Renesas Electronics RX Standard Toolchain (V.1.0.1.0)
	Compiler options: -cpu=rx600 -output=obj= "$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo (The default settings for the integrated development environment are used.)
Operating mode	Single-chip mode
Version of the sample code	1.00
Board used	Renesas Starter Kit (Under development as of February 21, 2012)

3. Hardware

3.1 List of Pins

Table 3.1 lists the pins used by the sample application.

Table 3.1 Pins and Their Functions

Pin Name	I/O	Description
P31/MTIOC0A-B	Output	Stepping motor operation control output (A-phase)
PB1/MTIOC0C	Output	Stepping motor operation control output ($\overline{A}$-phase)
P33/MTIOC3A	Output	Stepping motor operation control output (B-phase)
P32/MTIOC3C	Output	Stepping motor operation control output ($\overline{B}$-phase)

4. Software

4.1 Operation

4.1.1 Operating Principle of Stepping Motor

(1) Stepping Motor Operation Using Two-Phase Excitation

Figure 4.1 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 4.1, when a pulse is high the corresponding phase is excited.
- First, the $\overline{B}$-phase and A-phase are excited. This positions the rotor between the $\overline{B}$-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 ($\overline{B}$-phase, A-phase → A-phase, B-phase → B-phase, $\overline{A}$-phase → $\overline{A}$-phase, $\overline{B}$-phase), causing the rotor to turn.
- In reverse operation, the excitation sequence is $\overline{A}$-phase, $\overline{B}$-phase → B-phase, $\overline{A}$-phase → A-phase, B-phase → $\overline{B}$-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.

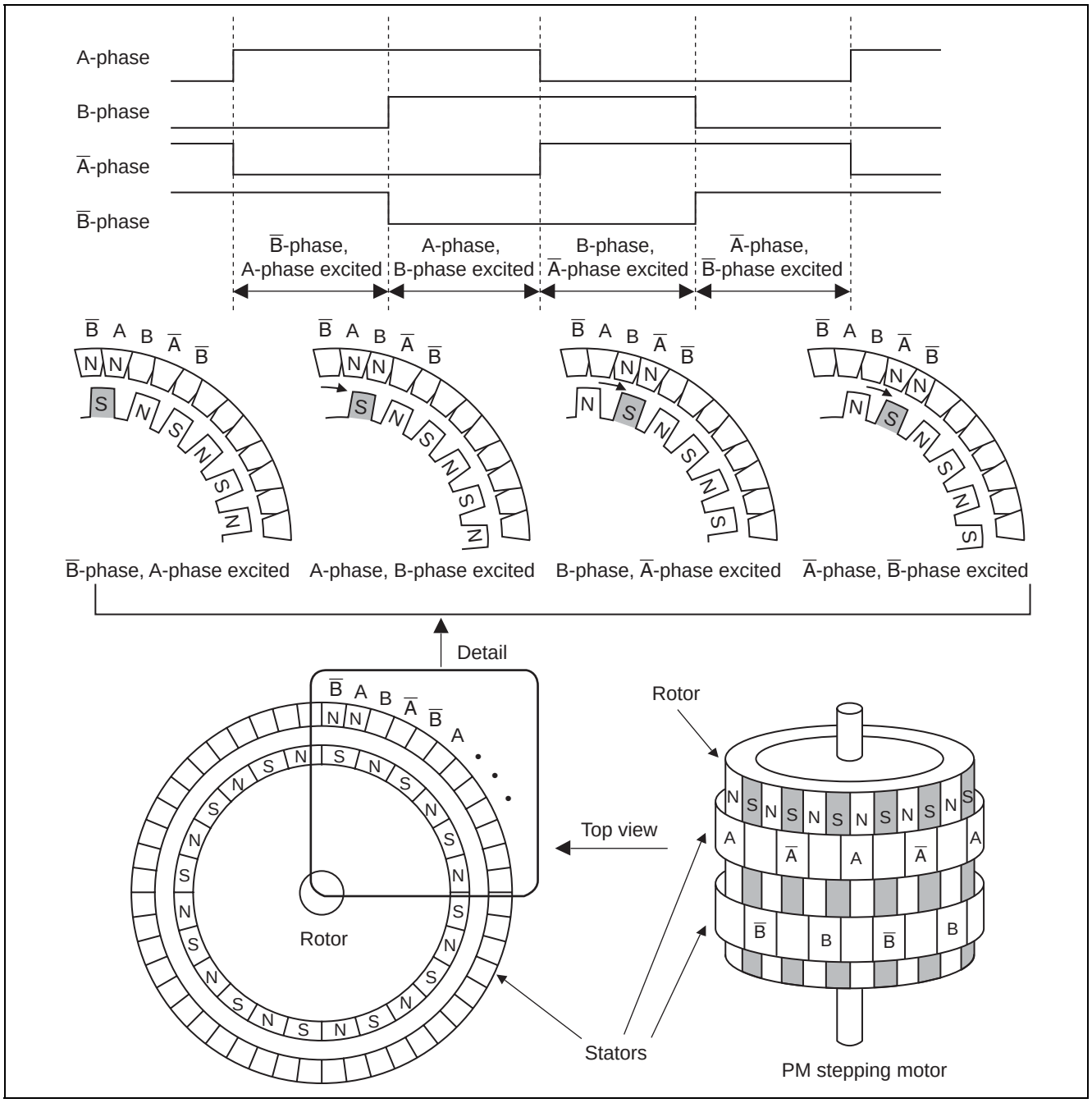

Figure 4.1 Stepping Motor Operation Example

(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 4.2 shows an example with no-overlap duration output.

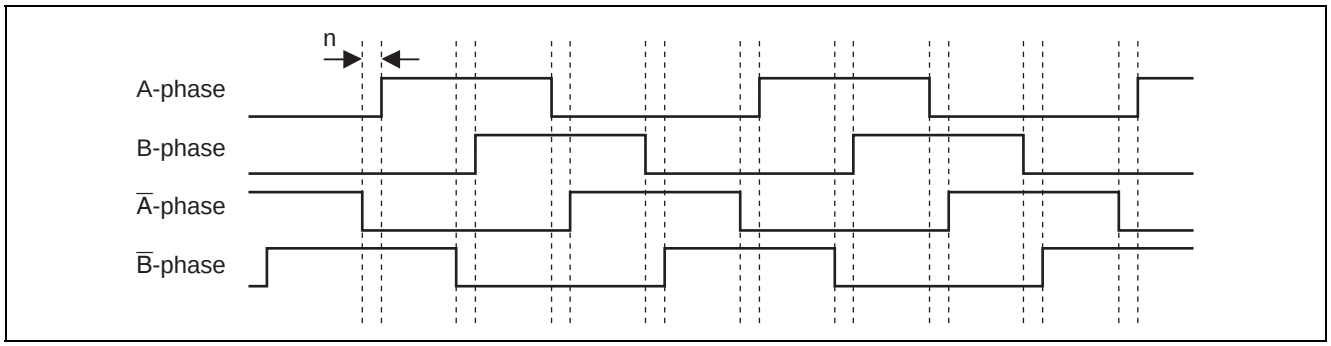

Figure 4.2 No-Overlap Duration Output Example

(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).

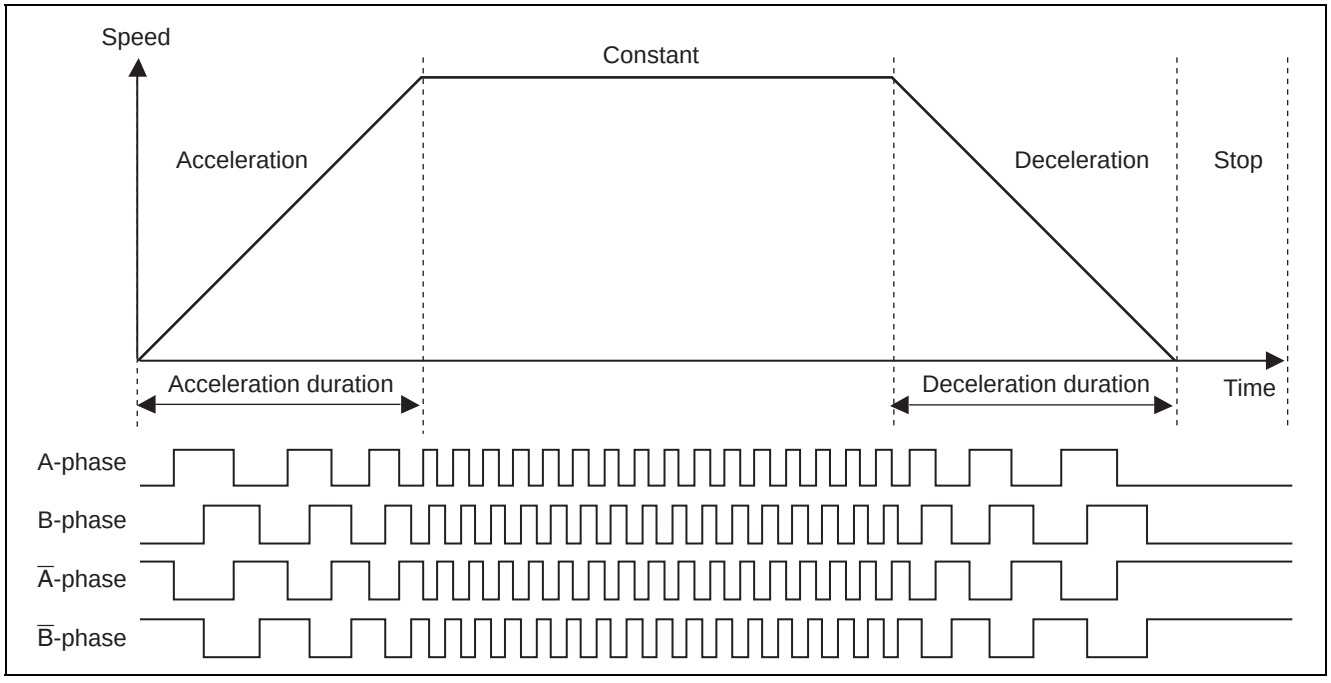

Figure 4.3 Stepping Motor Speed Control

4.1.2 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 3 (ch3) 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 ch3 of the MTU3, and the counters are cleared in the period register of ch0 (TGRA0). This causes ch0 and ch3 to start simultaneously and to operate synchronously.

In addition, ch0 and ch3 of the MTU3 are set to PWM mode 1 and output stepping motor control pulses as shown in table 4.1.

Table 4.1 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	
ch3	MTIOC3A pin	TGRA3	1 output	Output of B-phase stepping motor control pulse (PWM)
		TGRB3	0 output	
	MTIOC3C pin	TGRC3	1 output	Output of $\overline{B}$-phase stepping motor control pulse (PWM)
		TGRD3	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 TGRD3), which are listed in figure 4.4.

Table 4.2 lists the DTC transfer period data tables corresponding to each of the DTC transfer destinations. For details of the period data tables, see 4.6, Description of Data Tables.

Table 4.2 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
MTIOC3A output pin	TGRA3	cyctbl_A3	cyctbl_D3
	TGRB3	cyctbl_B3	cyctbl_C3
MTIOC3C output pin	TGRC3	cyctbl_C3	cyctbl_B3
	TGRD3	cyctbl_D3	cyctbl_A3

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

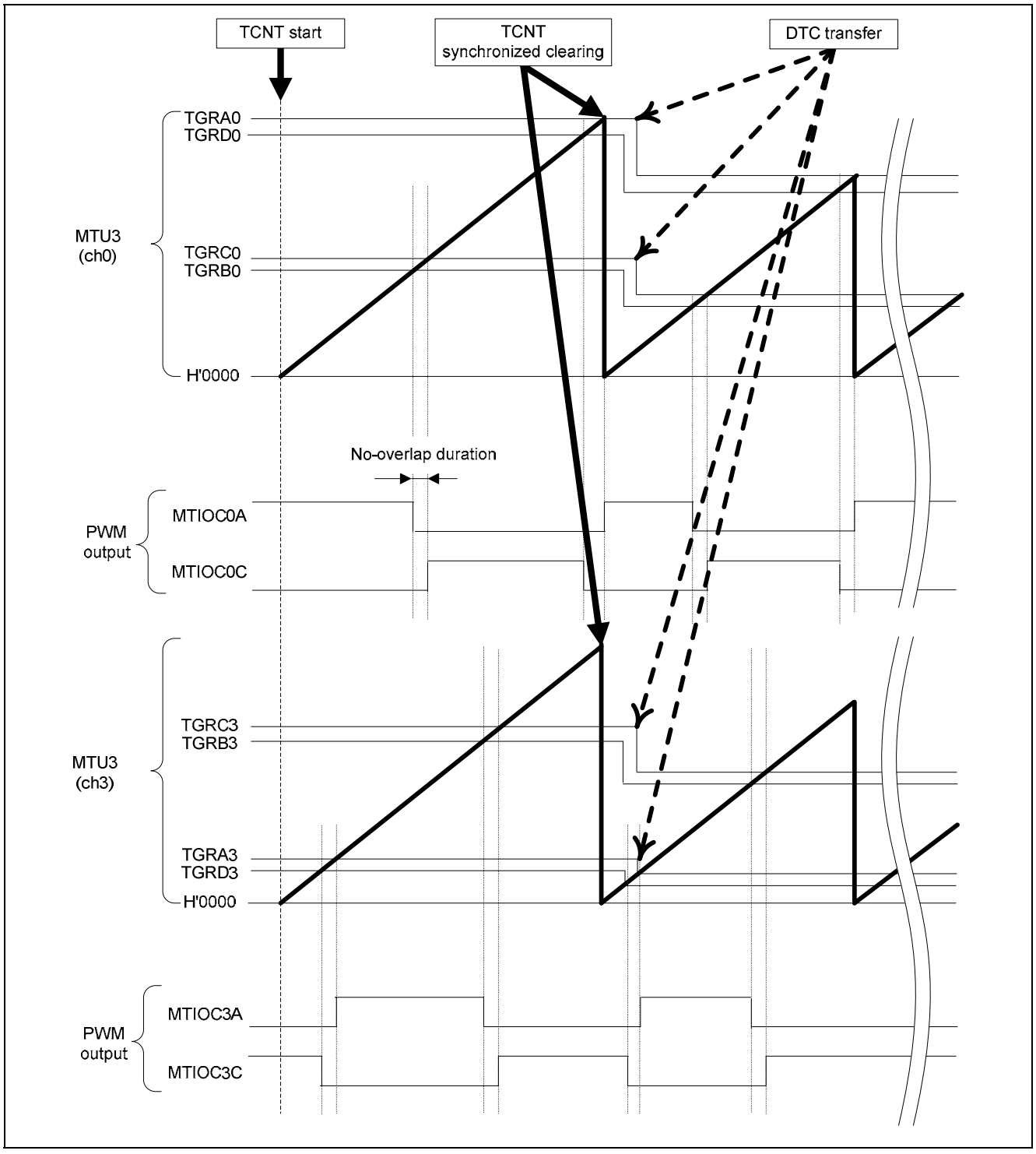

Figure 4.4 Stepping Motor Acceleration Control

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_A3 and cyctbl_D3, cyctbl_C3 and cyctbl_B3), thereby inverting the B-phase and $\overline{B}$-phase output waveforms.

Figure 4.5 shows the details of DTC chain transfers during forward and reverse operation.

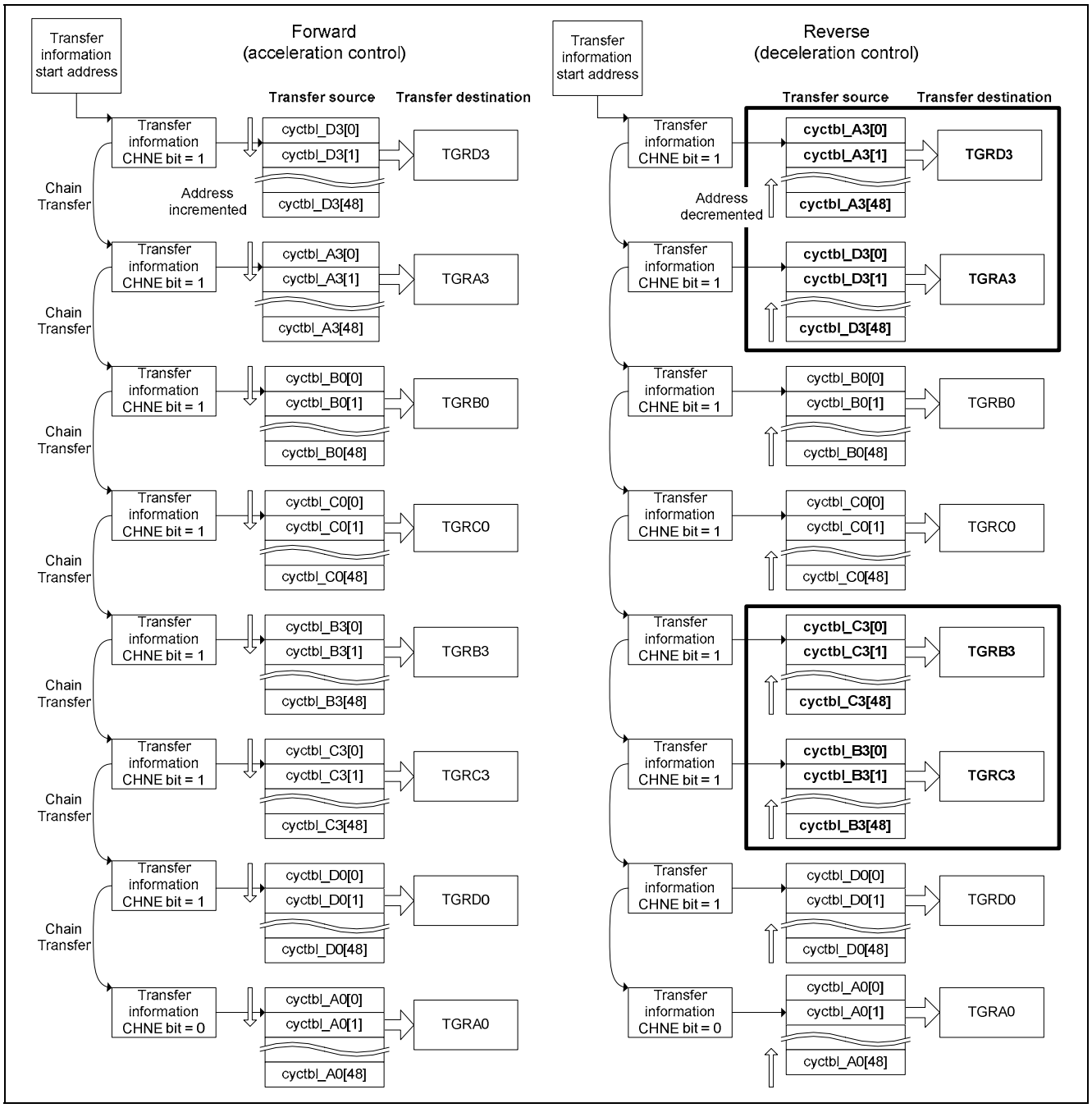

Figure 4.5 Chain Transfer

In stepping motor reverse operation, the DTC transfer source addresses of the period data tables (cyctbl_A3 and cyctbl_D3, cyctbl_C3 and cyctbl_B3) used by TGRA3 and TGRD3, TGRB3 and TGRC3 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.

4.1.3 DTC Settings

Table 4.3 lists the DTC transfer conditions of the sample application.

Table 4.3 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, ch3) of MTU3	Timer general registers (TGR) TGRA to TGRD (ch0, ch3) 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.

4.2 Required Memory Size

The memory size required for the DTC tables is listed in table 4.4.

Table 4.4 Required Memory Size

Memory Used	Size	Remarks
DTC tables (ROM)	784 bytes (98 bytes each)	cyctbl_A0, cyctbl_A3 cyctbl_B0, cyctbl_B3 cyctbl_C0, cyctbl_C3 cyctbl_D0, cyctbl_D3

4.3 List of Constants

Table 4.5 lists the constants used in the sample code.

Table 4.5 List of Constants

Constant	Value	Description
UPTIME	49	DTC transfer count during acceleration/deceleration control
CNSTTIME	48602	Measurement duration during constant control
NON_OVR	100	Shoot-through current prevention period (no-overlap duration)
STOPTIME	48602	Stop measurement duration

4.4 List of Structures and Unions

Figure 4.6 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 4.6 Structures and Unions Used in Sample Code (DTC Transfer Information)

4.5 List of Variables

Table 4.6 lists the variables used in the sample code.

Table 4.6 List of Variables

Type	Variable	Description	Used by Function
unsigned char	nextmode0	Switching among motor control modes 0: Forward acceleration control 1: Forward constant control 2: Forward deceleration control 3: Stop control 4: Reverse acceleration control 5: Reverse constant control 6: Reverse deceleration control 7: Stop control	main, tgia0_int, cmi0_int
st_dtc_full	dtc_tbl	Structure variable for transferring data to TGRA0 to TGRD0 and TGRA3 to TGRD3 of the MTU	dtc_init, fslowup0, fslowdwn0, rslowup0, rslowdwn0
void*	dtc_table[256]	DTC vector table to which address of DTC transfer information dtc_tbl[0] is assigned (The start address is set to 0x00000000.)	dtc_init

4.6 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_A3 and cyctbl_B3.

The data tables for producing $\overline{B}$-phase waveform output are cyctbl_C3 and cyctbl_D3.

```
cyctbl_A0[49] = {
   0xEA90, 0xE740, 0xE420, … , 0x5AA0, 0x5780, 0x5460
};

cyctbl_B0[49] = {
    0x7530 – NON_OVR, 0x73A0 – NON_OVR, … , 0x2BC0 – NON_OVR, 0x2A30 – NON_OVR
};

cyctbl_C0[49] = {
    0x7530, 0x73A0, 0x7210, … , 0x2D50, 0x2BC0, 0x2A30
};

cyctbl_D0[49] = {
   0xEA90 – NON_OVR, 0xE740 – NON_OVR, … , 0x5780 – NON_OVR, 0x5460 – NON_OVR
};

cyctbl_A3[49] = {
    0x3A98, 0x39D0, 0x3908, … , 0x16A8, 0x15E0, 0x1518
};

cyctbl_B3[49] = {
    0xAFC8 – NON_OVR, 0xAD70 – NON_OVR, … , 0x41A0 – NON_OVR, 0x3F48 – NON_OVR
};

cyctbl_C3[49] = {
    0xAFC8, 0xAD70, 0xAB18, … , 0x4F38, 0x41A0, 0x3F48
};

cyctbl_D3[49] = {
    0x3A98 – NON_OVR, 0x39D0 – NON_OVR, … , 0x15E0 – NON_OVR, 0x1518 – NON_OVR
};
```

Figure 4.7 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 4.8 shows the waveforms when the speed is changed.

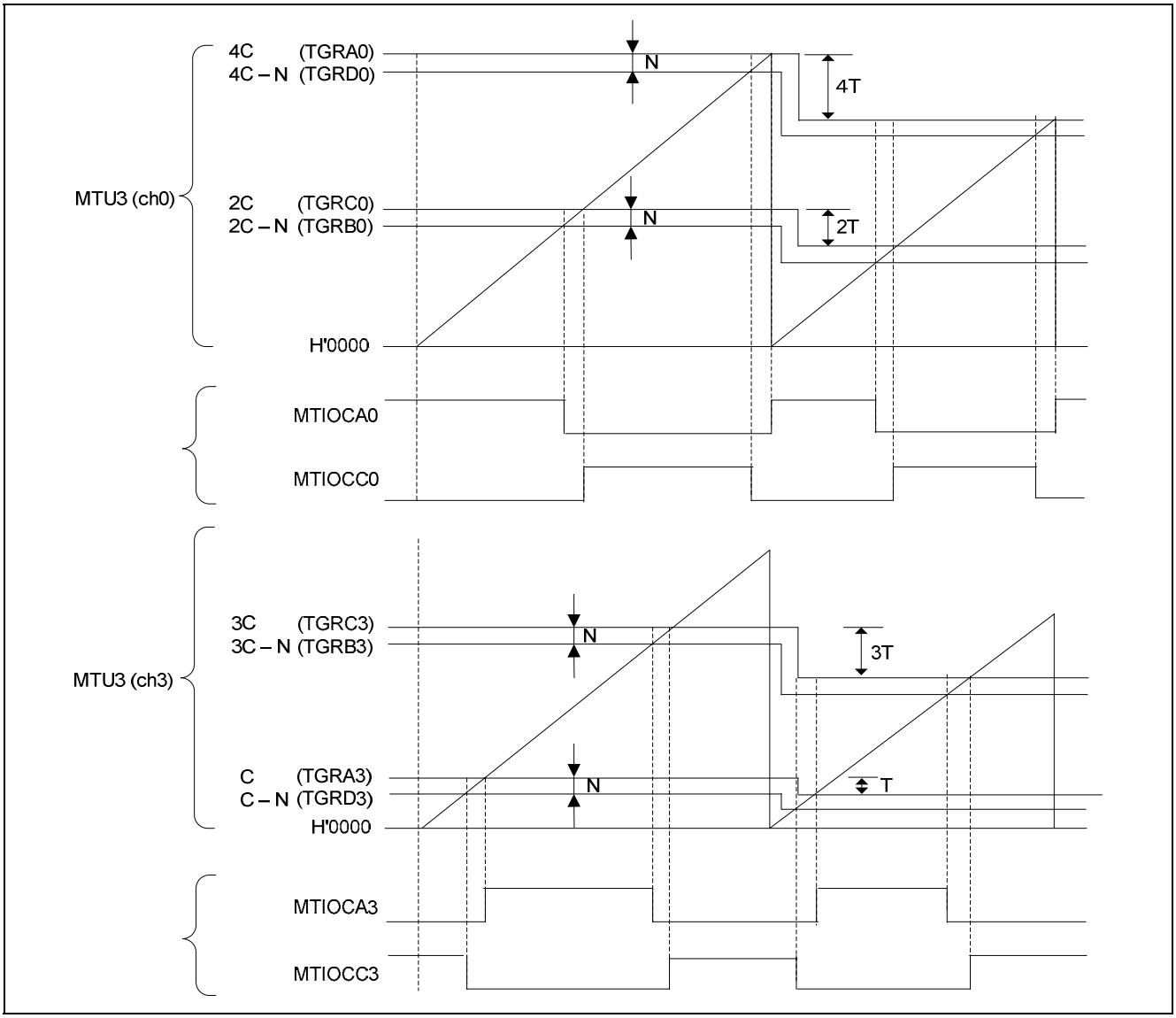

Figure 4.8 Changing the Period Data

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

Data A[n] = (C – n × T)

Data B[n] = (2C – n × 2T)

Data C[n] = (3C – n × 3T)

Data D[n] = (4C – n × 4T)

Data tables with shoot-through current prevention period inserted

Data A'[n] = (C – n × T) – N

Data B'[n] = (2C – n × 2T) – N

Data C'[n] = (3C – n × 3T) – N

Data D'[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_A3[0].

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

cyctbl_A3[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 4.9 shows example data tables created for the sample application.

cyctbl_A3[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 4.9 Acceleration and Deceleration Data Table Settings

4.7 List of Functions

Table 4.7 lists the functions.

Table 4.7 List of Functions

Function	Description
HardwareSetup	Initialization processing, clock settings, cancellation of module-stop state
main	Main process Makes initial ICU, MTU3, CMT, and DTC settings, starts the timer count.
icu_init	ICU initial settings, interrupt level settings
dtc_init	DTC initial settings, transfer information settings, DTC vector base register setting, DTC operation enable
mtu3_init	MTU3 initial settings, TCNT synchronous operation (ch0, ch3), TCNT cleared (ch0) at TGRA0 compare match, PWM mode 1 setting, TGR settings for forward acceleration control, TGIA0 interrupt enable
cmt0_init	CMI0 initial settings Clock selection, CMI0 interrupt enable
tgia0_int	MTU3 interrupt handler Settings for constant control or stop control after end of acceleration control or deceleration control
cmi0_int	CMI interrupt handler Ends constant control or stop control and makes settings for acceleration control or deceleration control.
fslowup0	Makes settings to DTC transfer information for reverse acceleration control.
frconst0	Switches to constant control after end of forward or reverse acceleration control.
fslowdwn0	Makes settings to DTC transfer information for forward deceleration control.
rslowup0	Makes settings to DTC transfer information for reverse acceleration control.
rslowdwn0	Makes settings to DTC transfer information for reverse deceleration control.
frstop0	Switches to stop control after end of forward or reverse deceleration control.
CmtStart	CMT start processing
CmtStop	CMT stop processing
Excep MTU30 TGIA0	Processing to call function tgia0_int
Excep CMT0 CMI0	Processing to call function cmi0_int

4.8 Function Specifications

The specifications of the functions of the sample code are listed below.

HardwareSetup

Overview	Initialization processing
Header	iodefine.h
Declaration	void HardwareSetup(void)
Description	This function makes initial settings. • System clock and peripheral module clock settings • Module control register settings
Arguments	None
Return values	None
Notes	

main

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

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
Notes	

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 transfer information address of DTC activation source. • Sets DTC transfer information for forward acceleration control. • Sets DTC vector base register. • After setting DTC transfer information, specifies DTC module control.
Arguments	None
Return values	None
Notes	

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 3 to synchronous operation so TCNT0 and TCNT3 are cleared simultaneously at channel 0 TGRA compare match. • Sets channels 0 and 3 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 TGRA3 to TGRD3 for channel 3. • Enables MTU3 module TGIA interrupt and clears timer status flag (TSR) of channel 0 and channel 3.
Arguments	None
Return values	None
Notes	

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 = 1 After forward constant control, transition to forward deceleration control 2. When nextmode0 = 3 After stop control, transition to reverse acceleration control 3. When nextmode0 = 5 After reverse constant control, transition to reverse deceleration control 4. When nextmode0 = 7 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
Notes	

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 = 0 After forward acceleration control, transition to forward constant control 2. When nextmode0 = 2 After forward deceleration control, transition to stop control 3. When nextmode0 = 4 After reverse acceleration control, transition to reverse constant control 4. When nextmode0 = 6 After reverse deceleration control, transition to stop control • Specifies DTC module operation. • The TGFA flag of MTU3 channel 0 is cleared. • Enables MTU3 TGIA0 interrupt request enable register.
Arguments	None
Return values	None
Notes	

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
Notes	

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
Notes	

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
Notes	

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
Notes	

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
Notes	

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
Notes	

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
Notes	

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
Notes	

4.9 Flowcharts

4.9.1 Initialization Processing

Figure 4.10 is a flowchart of the initialization processing.

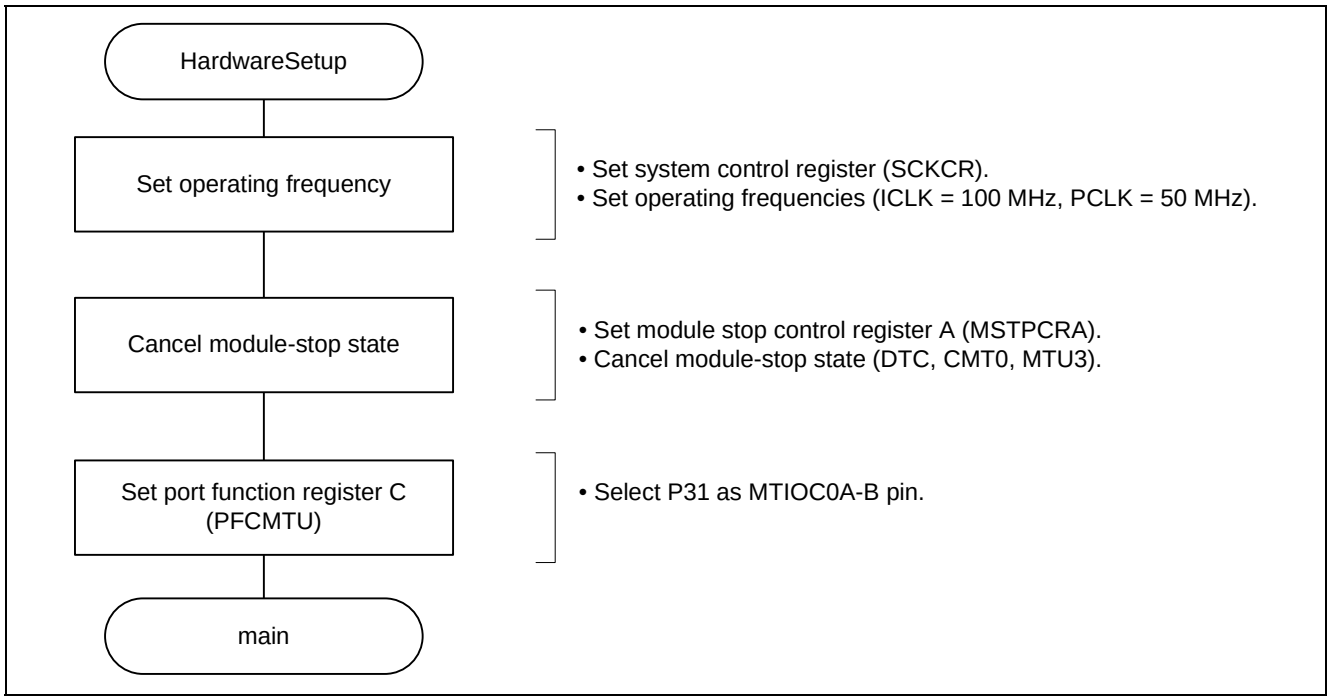

Figure 4.10 Initialization Processing

4.9.2 Main Process

Figure 4.11 is a flowchart of the main process.

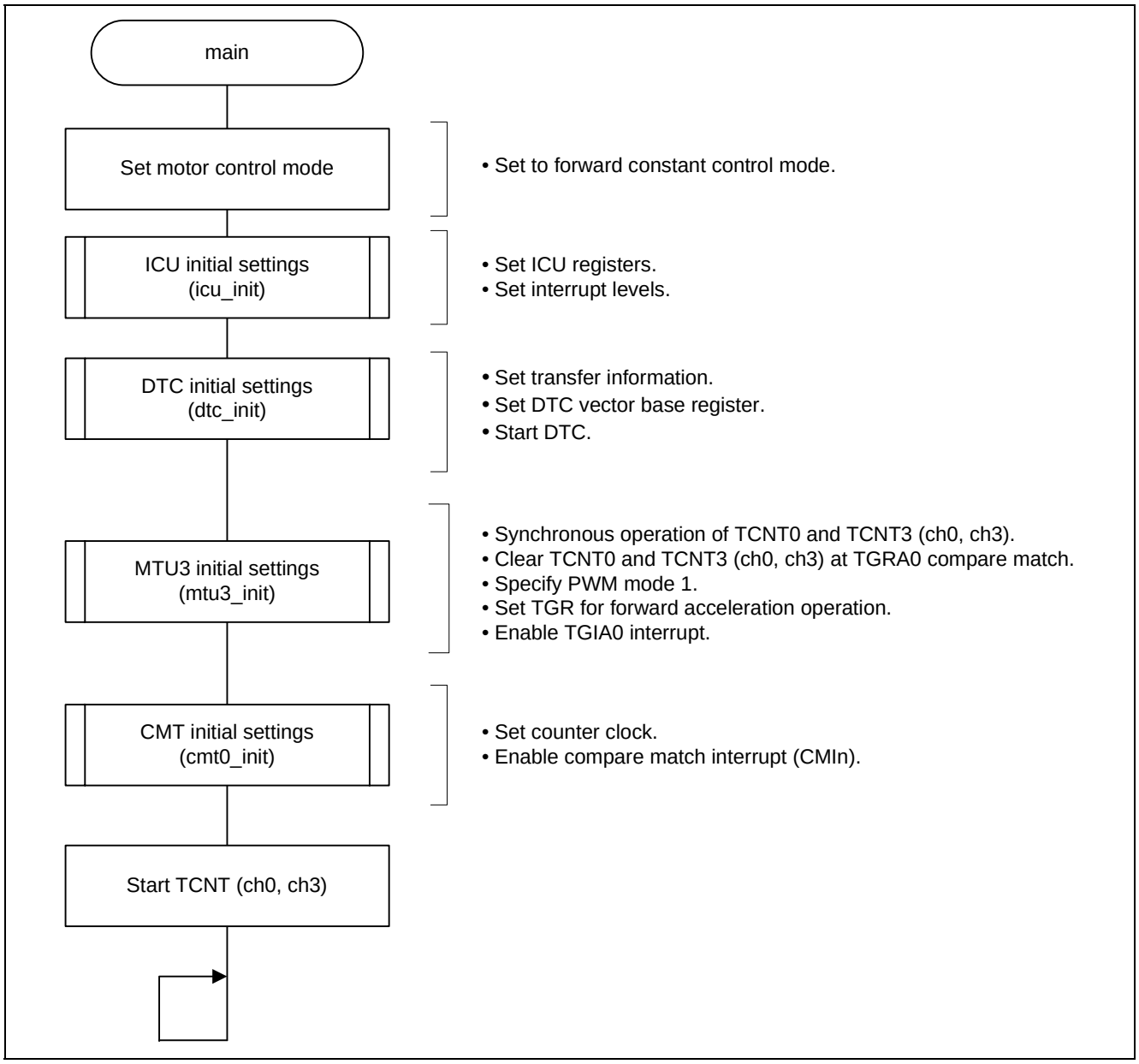

Figure 4.11 Main Process

4.9.3 ICU Initial Settings

Figure 4.12 is a flowchart of the ICU initial settings.

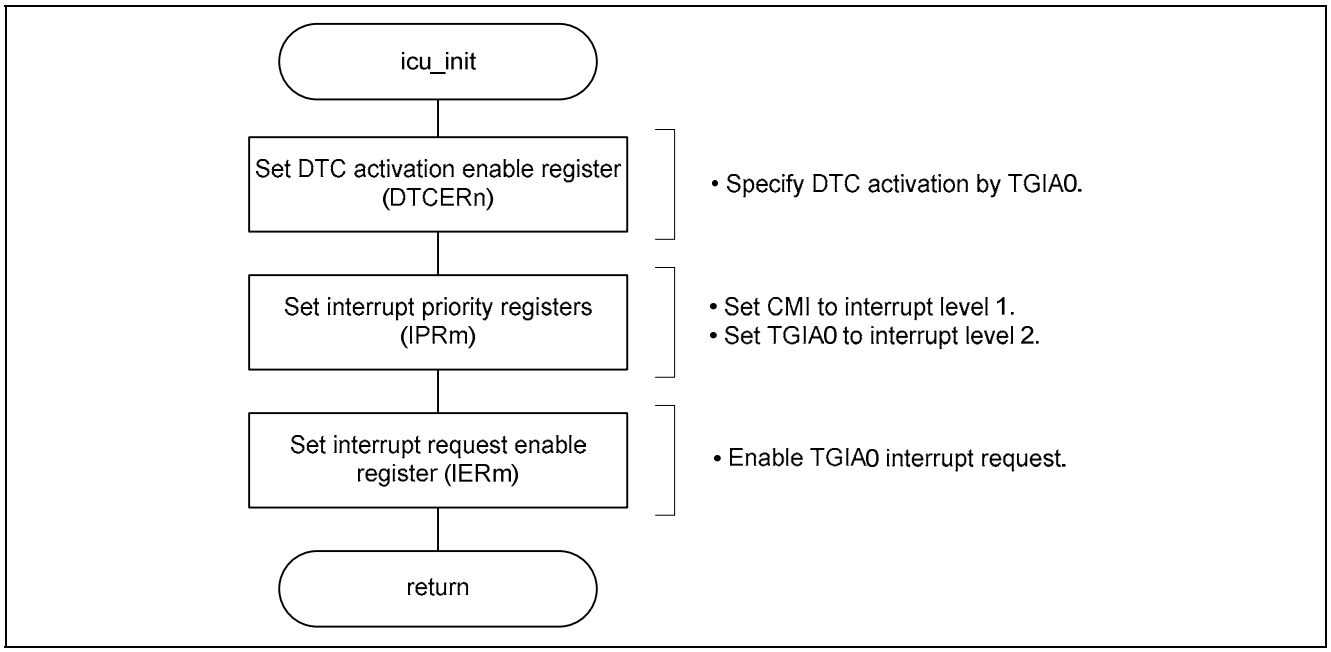

Figure 4.12 ICU Initial Settings

4.9.4 DTC Initial Settings

Figure 4.13 is a flowchart of the DTC initial settings.

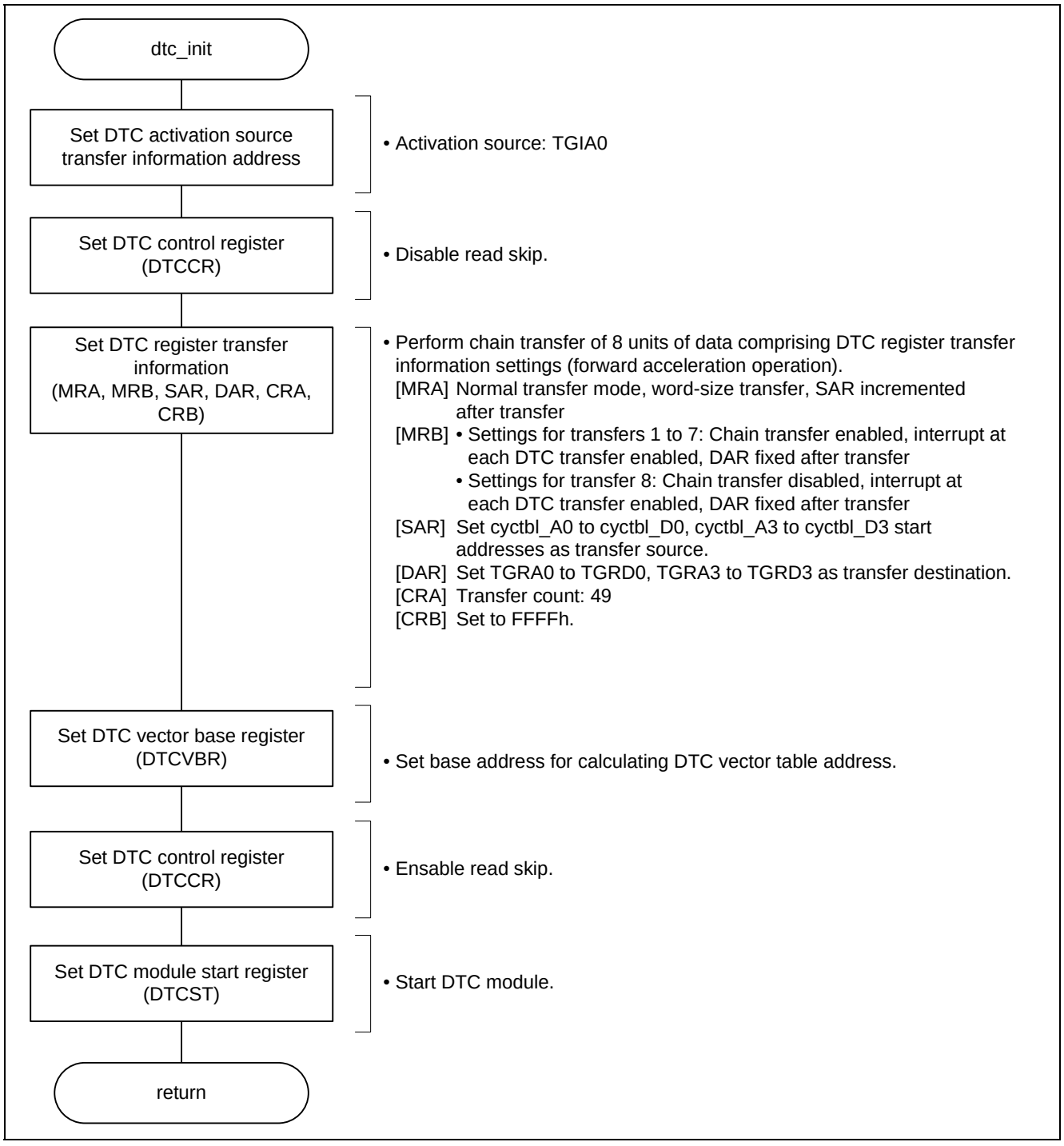

Figure 4.13 DTC Initial Settings

4.9.5 MTU3 Initial Settings

Figure 4.14 is a flowchart of the MTU3 initial settings.

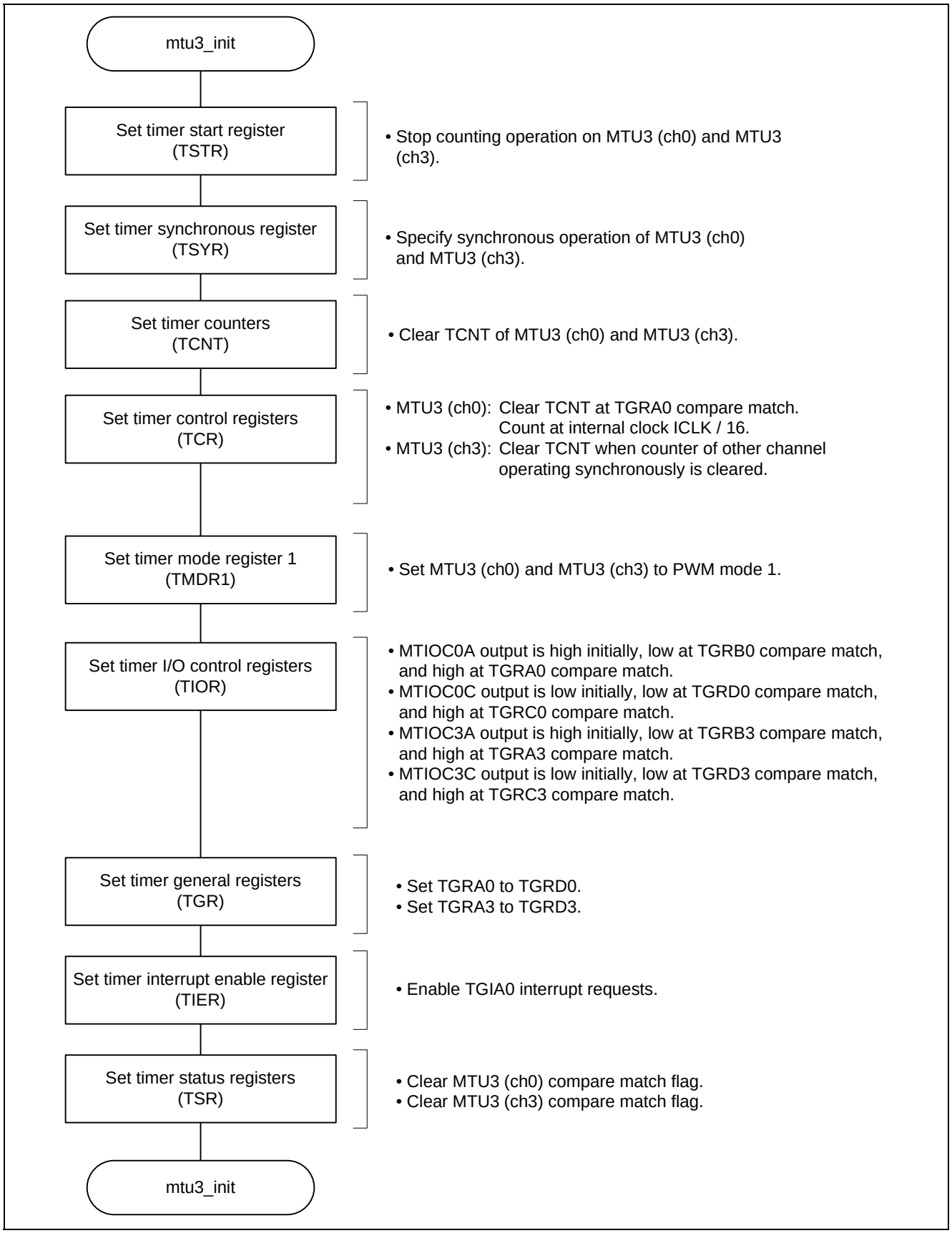

Figure 4.14 MTU3 Initial Settings

4.9.6 CMT0 Initial Settings

Figure 4.15 is a flowchart of the CMT0 initial settings.

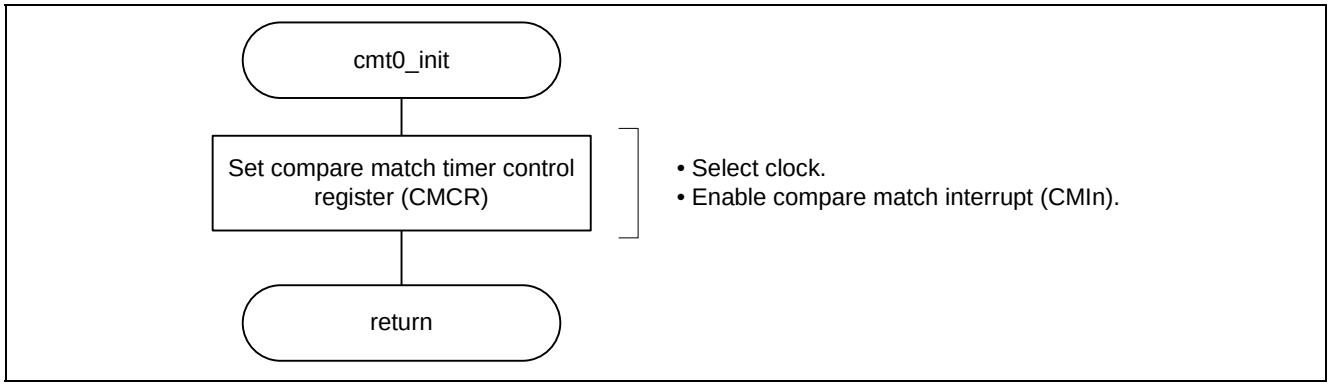

Figure 4.15 CMT0 Initial Settings

4.9.7 TGIA0 Interrupt Handler

Figure 4.16 is a flowchart of the TGIA0 interrupt handler.

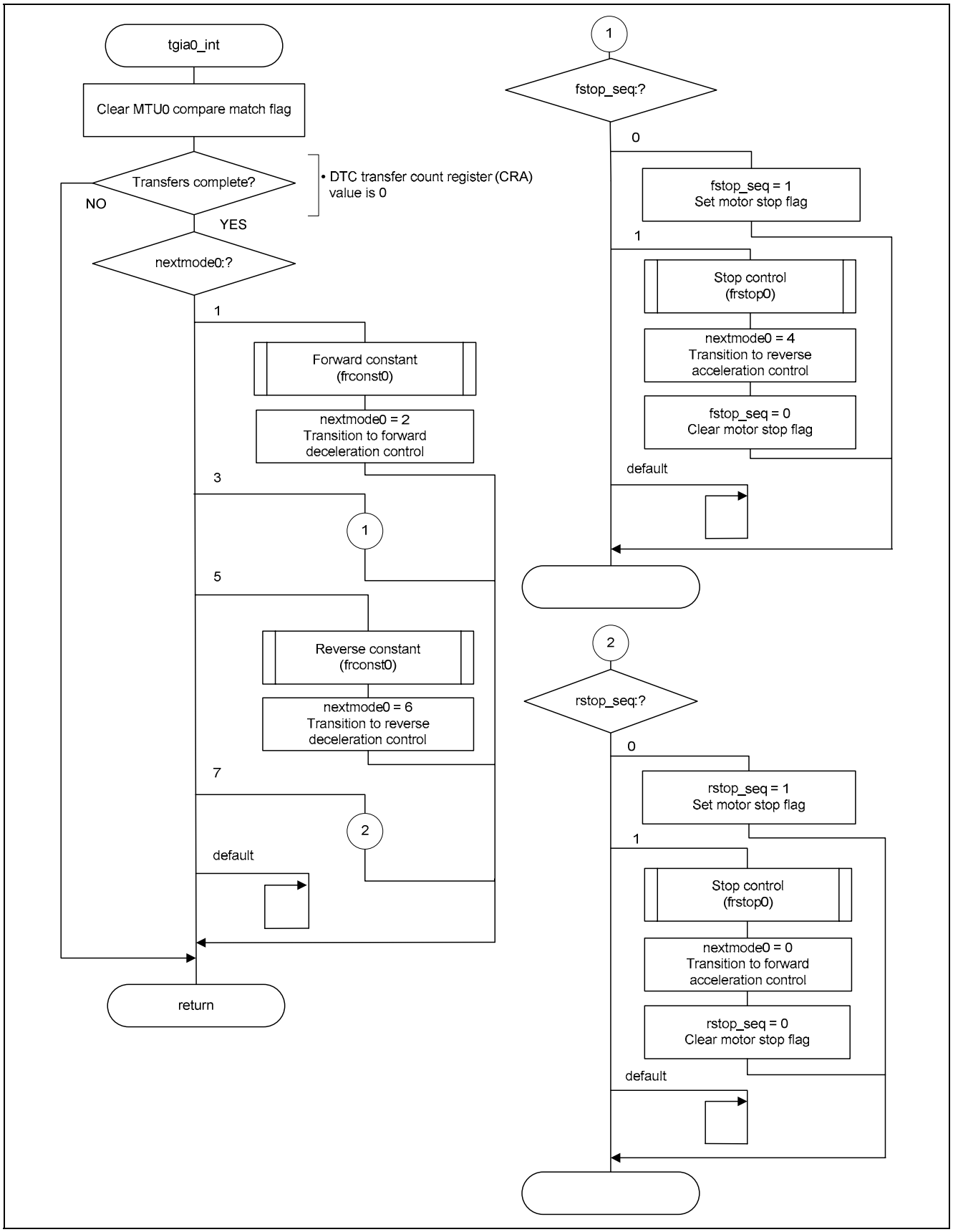

Figure 4.16 TGIA0 Interrupt

4.9.8 CMI0 Interrupt Handler

Figure 4.17 is a flowchart of the CMI0 interrupt handler.

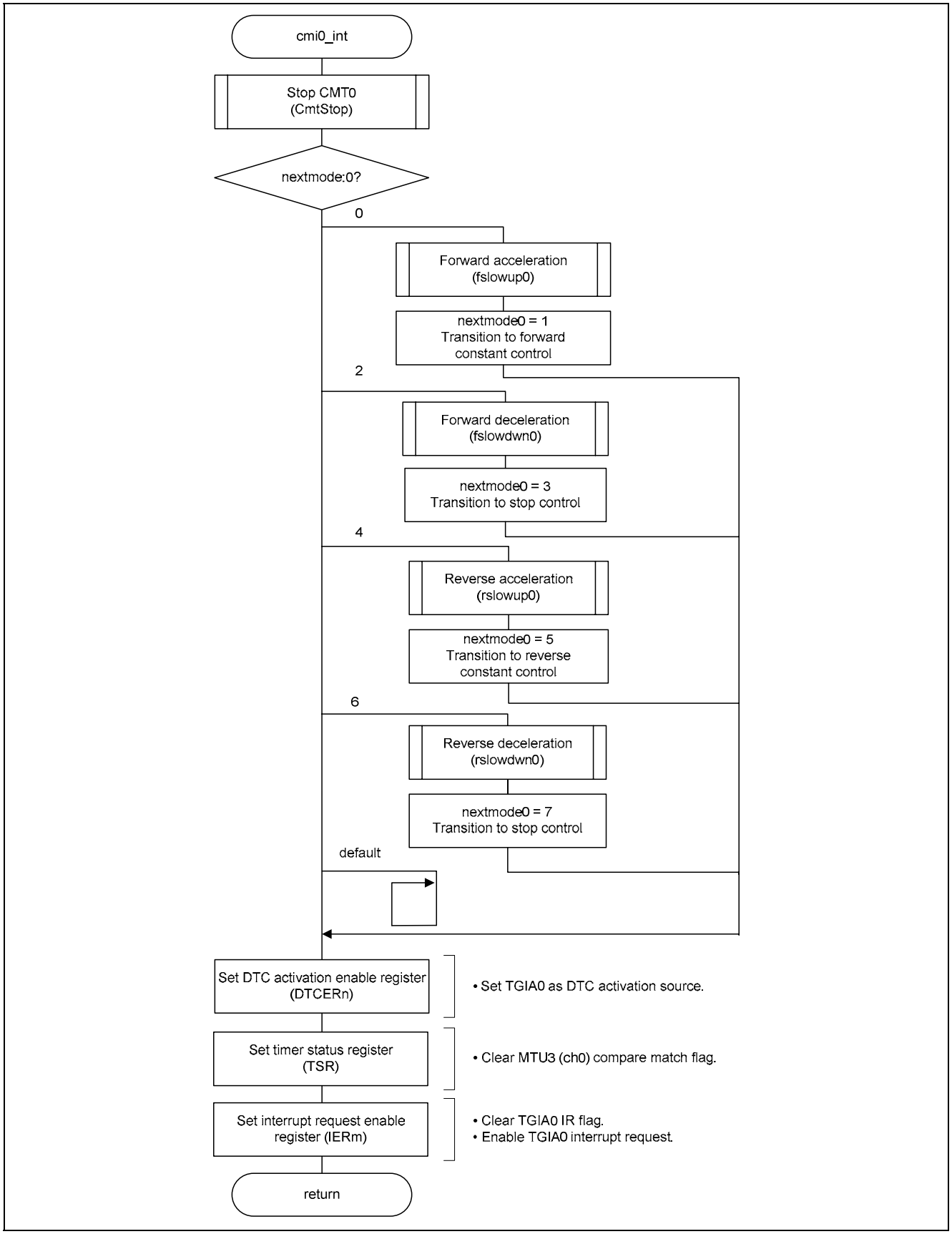

Figure 4.17 CMI0 Interrupt

4.9.9 Forward Acceleration Processing

Figure 4.18 is a flowchart of the forward acceleration processing.

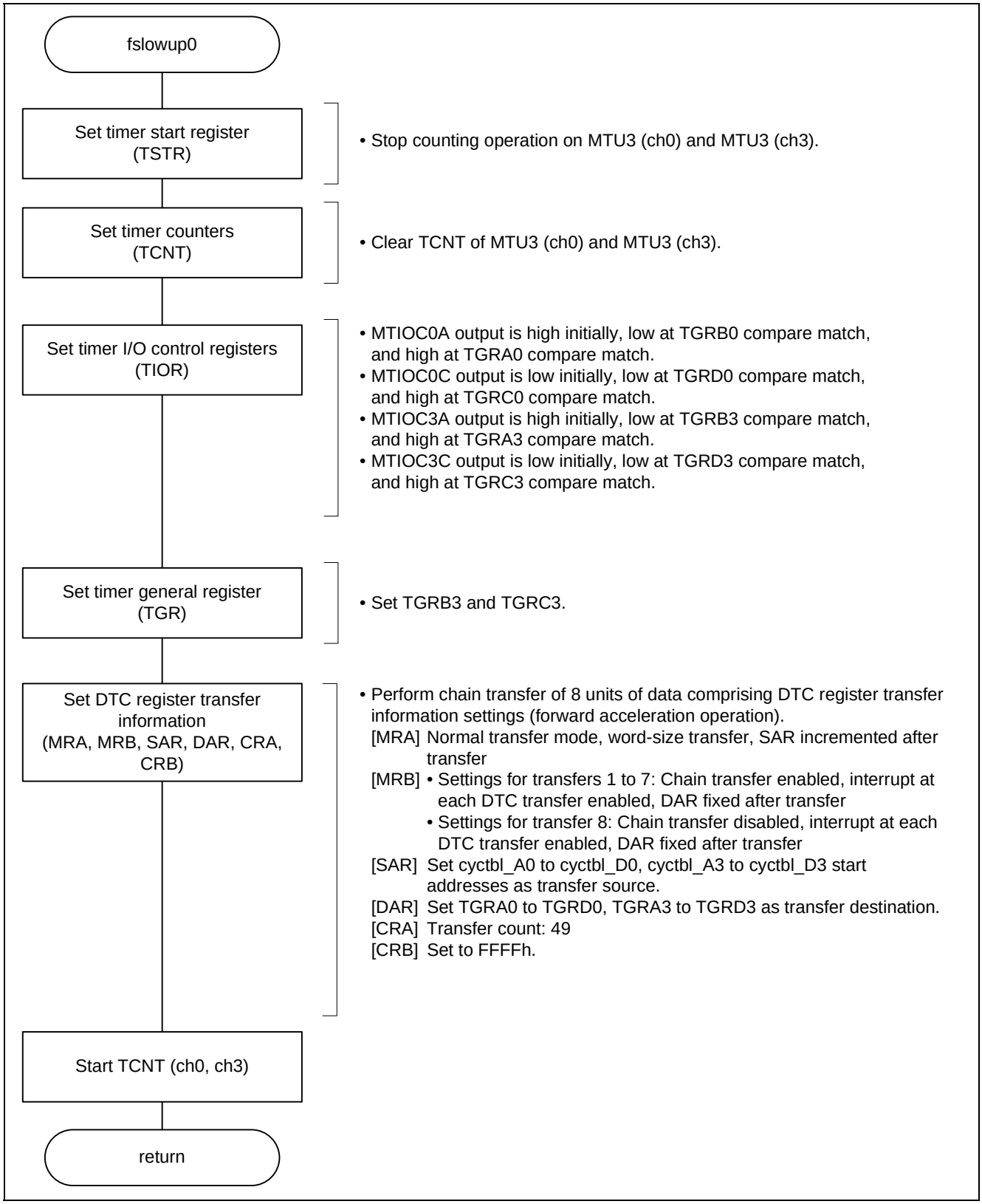

Figure 4.18 Forward Acceleration Processing

4.9.10 Forward/Reverse Constant Processing

Figure 4.19 is a flowchart of the forward/reverse constant processing.

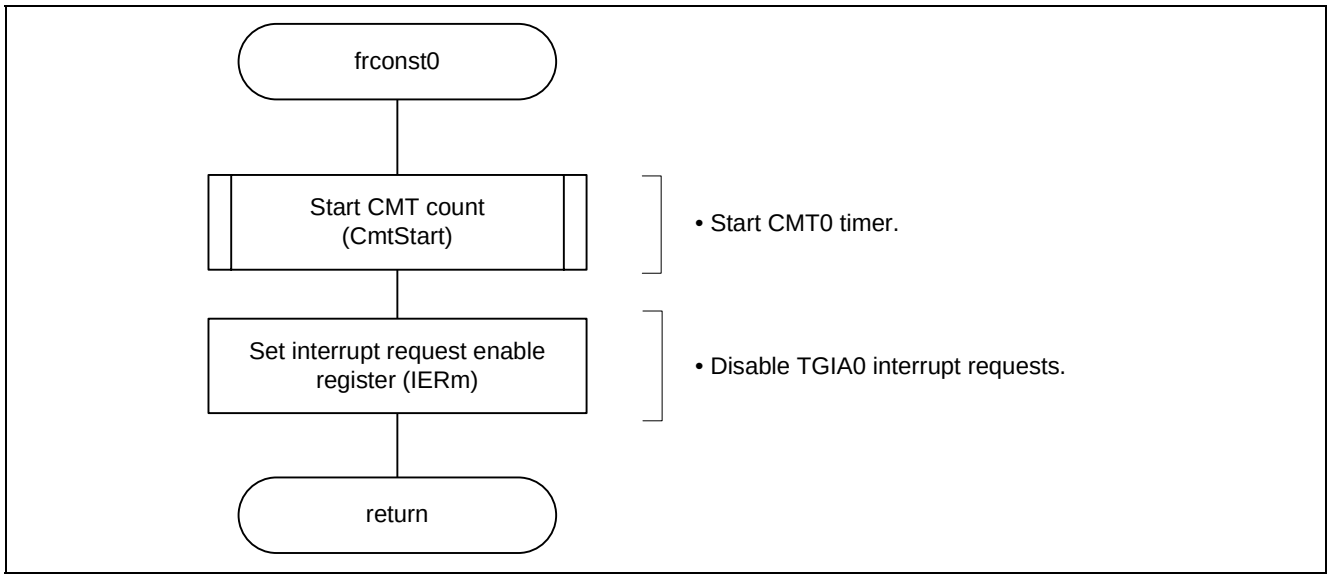

Figure 4.19 Forward/Reverse Constant Processing

4.9.11 Forward Deceleration Processing

Figure 4.20 is a flowchart of the forward deceleration processing.

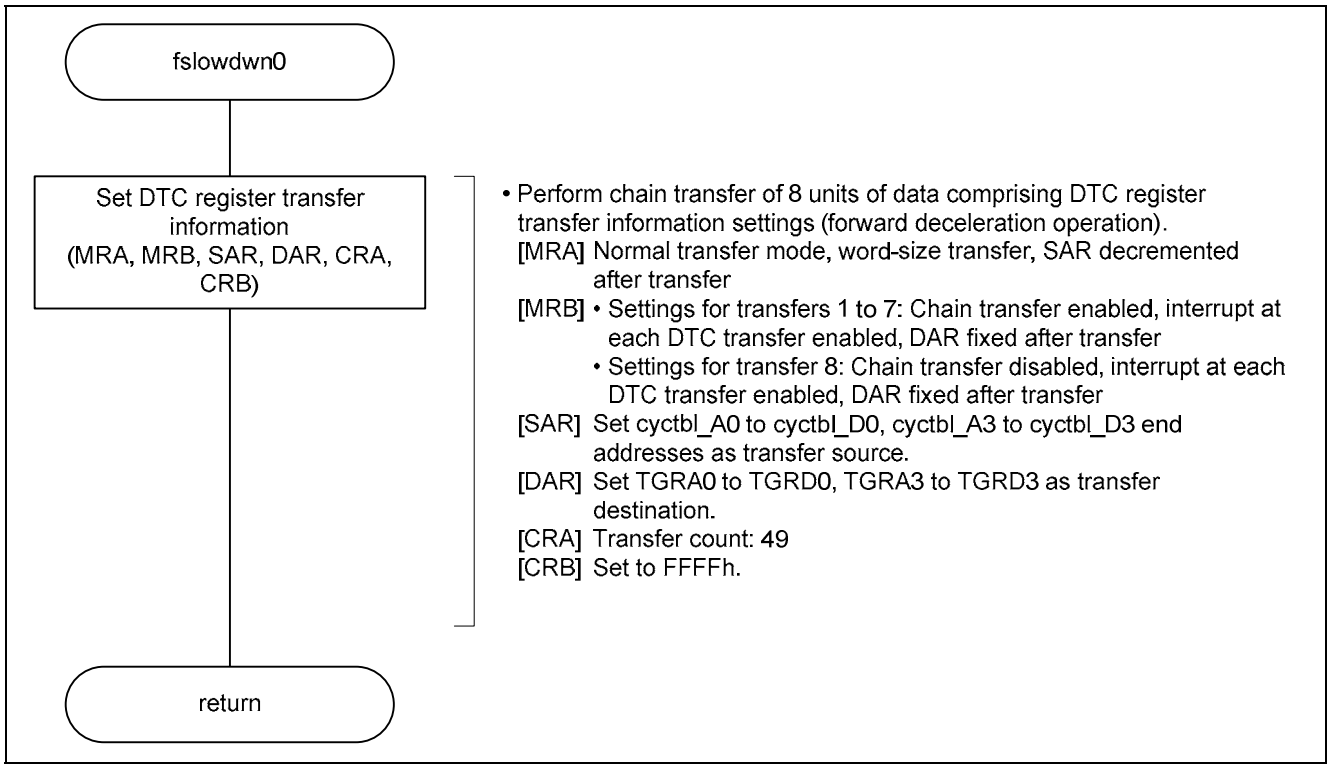

Figure 4.20 Forward Deceleration Processing

4.9.12 Reverse Acceleration Processing

Figure 4.21 is a flowchart of the reverse acceleration processing.

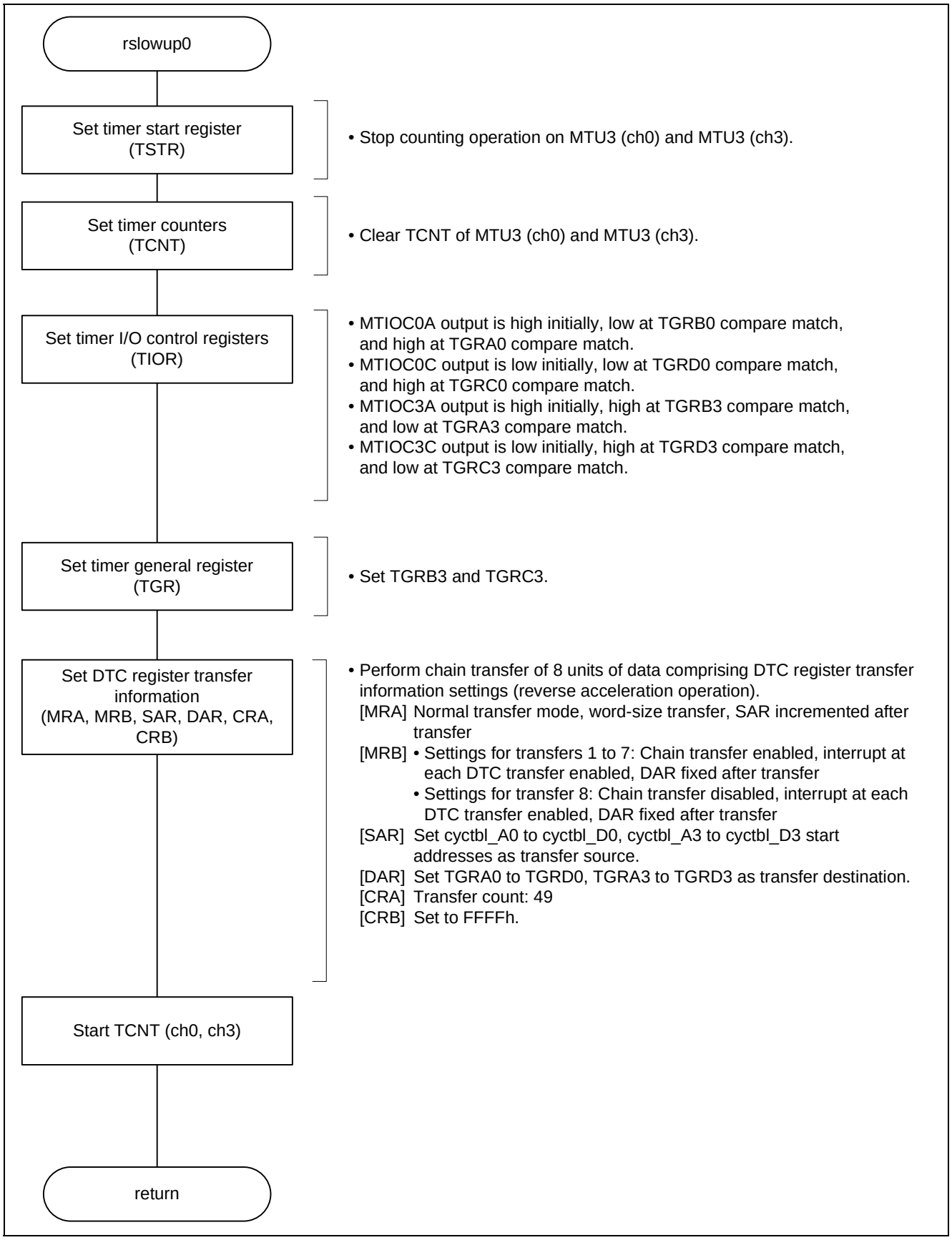

Figure 4.21 Reverse Acceleration Processing

4.9.13 Reverse Deceleration Processing

Figure 4.22 is a flowchart of the reverse deceleration processing.

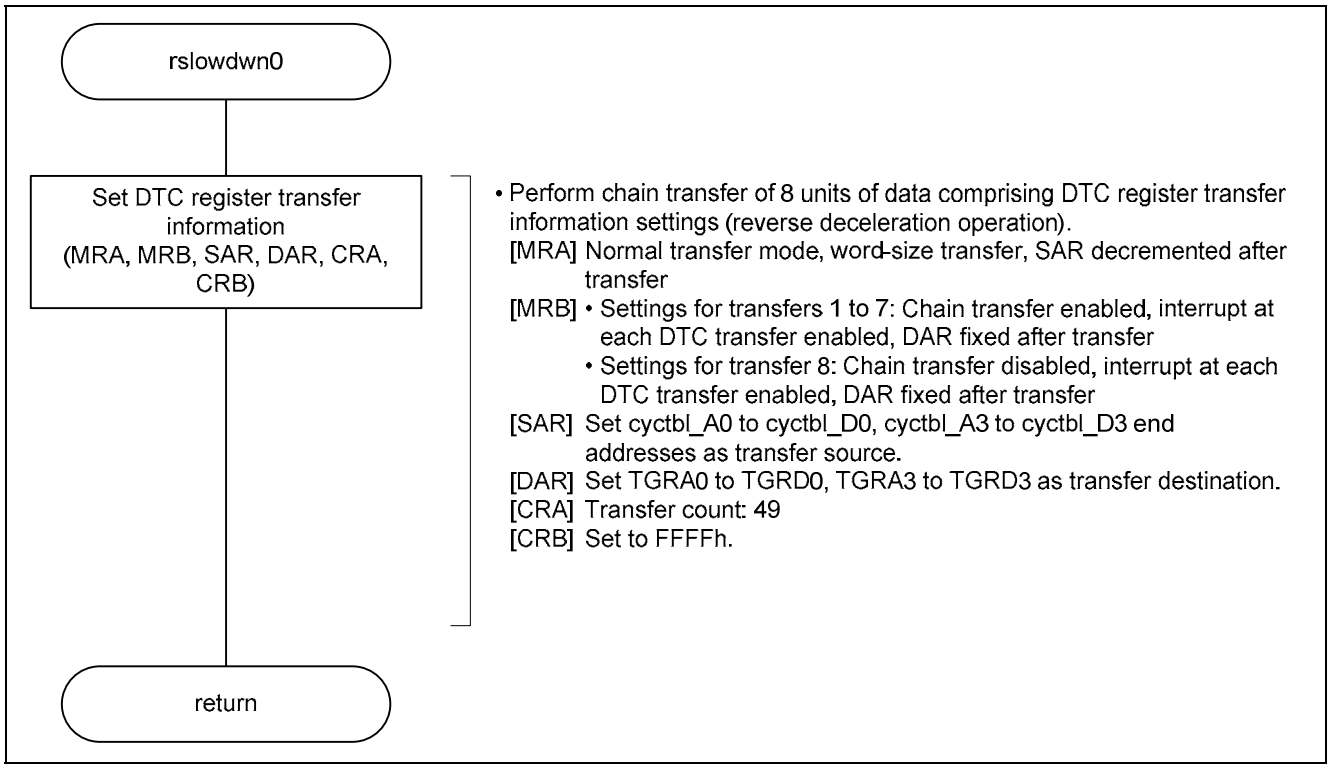

Figure 4.22 Reverse Deceleration Processing

4.9.14 Stop Processing

Figure 4.23 is a flowchart of the stop processing.

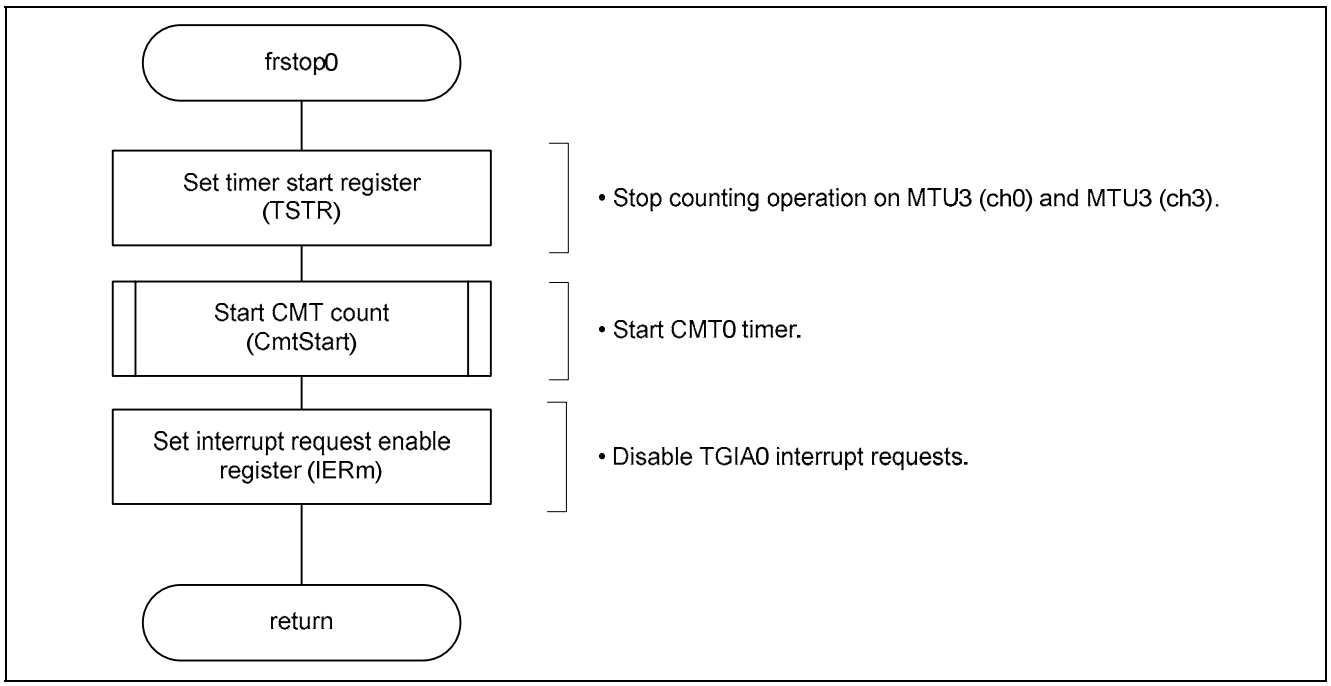

Figure 4.23 Stop Processing

4.9.15 CMT Start Processing

Figure 4.24 is a flowchart of the CMT start processing.

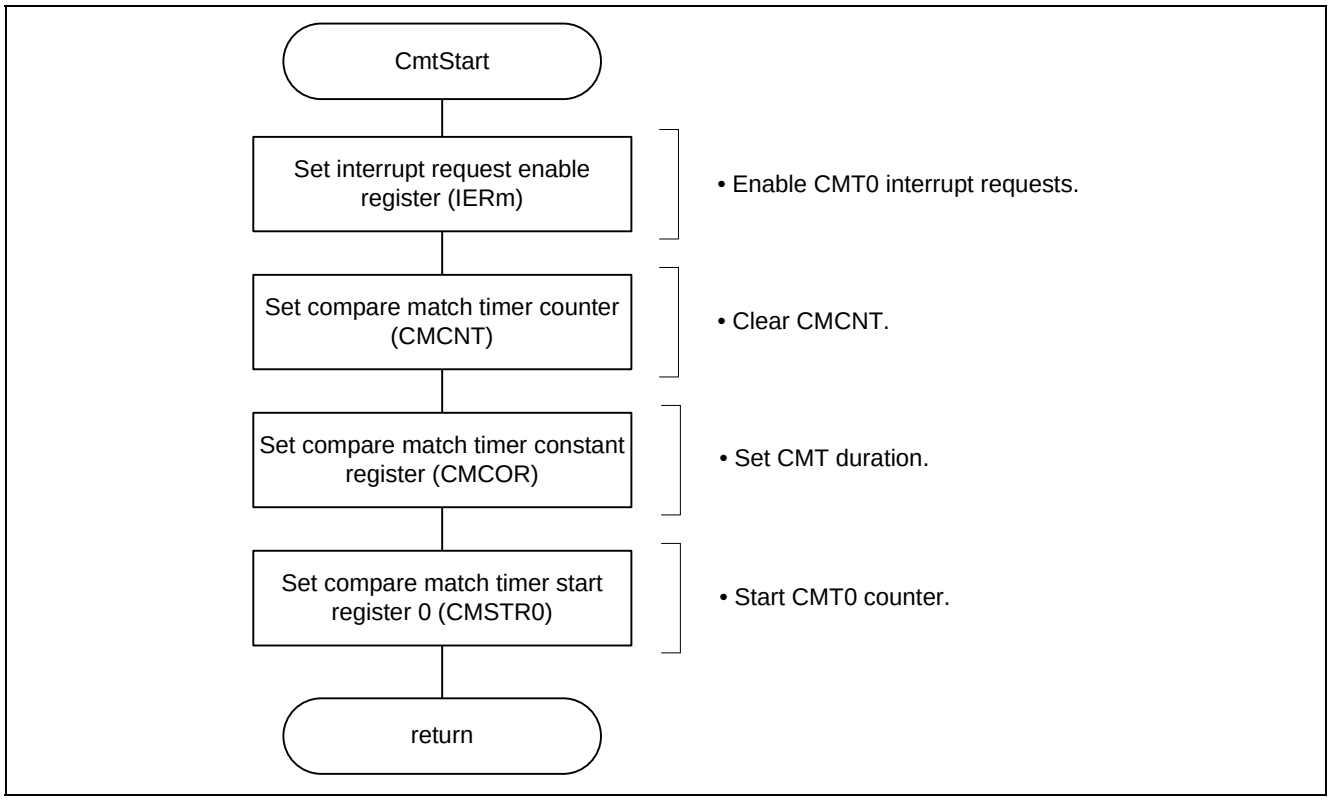

Figure 4.24 CMT Operation Start

4.9.16 CMT Stop Processing

Figure 4.25 is a flowchart of the CMT stop processing.

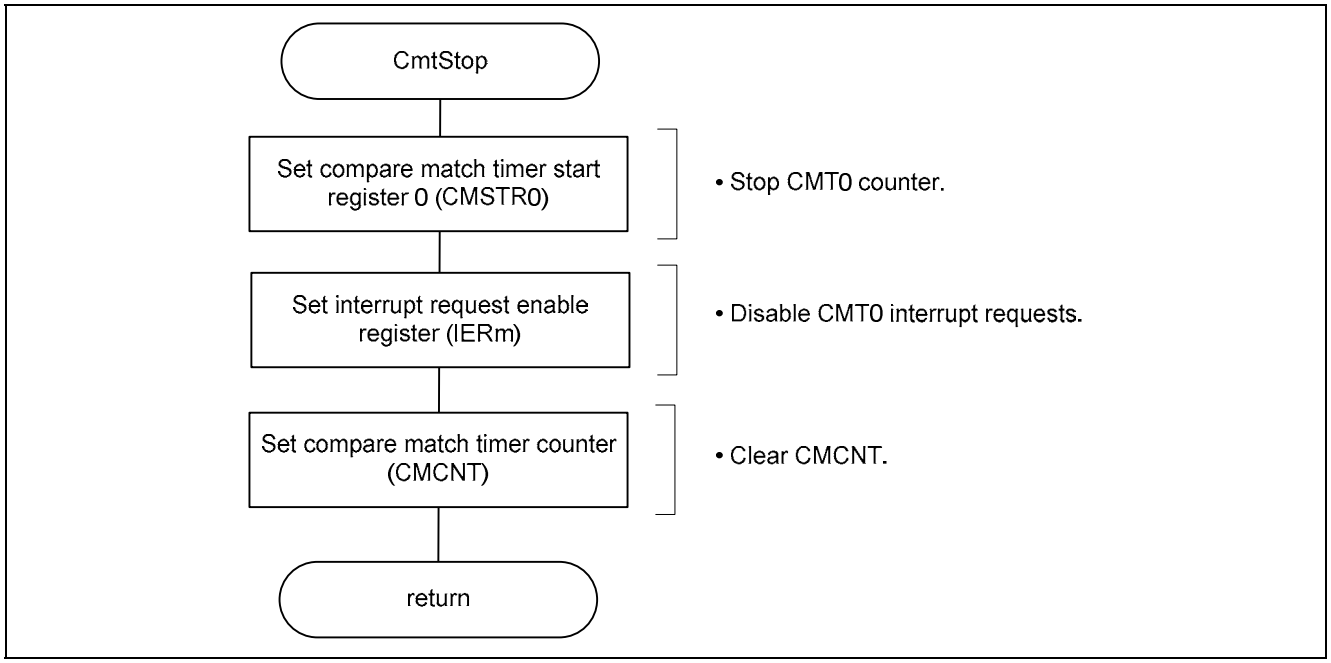

Figure 4.25 CMT Operation Stop

5. Important Points

5.1 Operating Mode Settings

The sample application sets mode pins MD1 and MD0 to 1 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 5.1 lists the operating mode settings of the sample application.

Table 5.1 Operating Mode Settings

Mode Pin		SYSCR0 Register		
MD1	**MD0**	**ROME**	**Operating Mode**	**On-chip ROM**
1	1	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.

5.2 Endian Mode Setting

The sample application supports both the big-endian and little-endian modes. Table 5.2 shows how to specify the endian mode in hardware.

Table 5.2 Endian Mode Settings (Hardware)

MDE Pin	Endian Mode
0	Little endian
1	Big endian

Table 5.3 shows the correspondence between the compiler MCU options and the endian modes.

Table 5.3 Endian Mode Settings (Compiler Options)

MCU Option	Endian Mode
endian = little	Little endian
endian = big	Big endian

Note: Set the MDE pin to match the endian mode selected as the compiler option when compiling the program.

5.3 Bit Order Setting

The sample application supports both right and left bit order. Table 5.4 shows the correspondence between the compiler MCU options and the bit order.

Table 5.4 Bit Order Settings

MCU Option	Endian Mode
bit_order = right	Bit field members assigned in order starting from the lowest bit (default if no option specified)
bit_order = left	Bit field members assigned in order starting from the highest bit

Note: The sample application uses the I/O register definition file (iodefine.h) to specify the bit order in bit fields. The setting left is specified with the #pragma bit_order extension, so bit field members are assigned in order starting from the highest bit.

Note: When both the bit_order compiler option and #pragma bit_order extension are used, the #pragma bit_order extension takes precedence. Therefore, the I/O register definition file ensures that the bit field members are assigned in order starting from the highest bit, regardless of the bit_order compiler option.

6. Sample Code

The sample code is available for download from the Renesas Electronics Web site.

7. Reference Documents

- RX62T Group User's Manual: Hardware, Rev.1.10
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- RX Family User's Manual: Hardware, Rev.1.00
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- Technical Updates/Technical News
  (The latest information can be downloaded from the Renesas Electronics Web site.)
- C Compiler Manual
  RX Family C/C++ Compiler Package V.1.01 Release 00
  RX Family C/C++ Compiler Package User's Manual V.1.0.1.0
  (The latest version can be downloaded from the Renesas Electronics Web site.)

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Mar.28.12	—	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 part 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.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

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: +44-1628-585-100, Fax: +44-1628-585-900

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 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

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.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1