To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

RENESAS

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

RENESAS

Application Note

V850ES/JG2, V850ES/JJ2

32-bit Single-Chip Microcontrollers

V850ES/JG2:
μPD70F3715
μPD70F3716
μPD70F3717
μPD70F3718
μPD70F3719

V850ES/JJ2:
μPD70F3720
μPD70F3721
μPD70F3722
μPD70F3723
μPD70F3724

Document No. U18676EJ1V0AN00 (1st edition)
Date Published July 2007 N

© NEC Electronics Corporation 2007
Printed in Japan

[MEMO]

NOTES FOR CMOS DEVICES

① **VOLTAGE APPLICATION WAVEFORM AT INPUT PIN**

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

② **HANDLING OF UNUSED INPUT PINS**

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

③ **PRECAUTION AGAINST ESD**

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

④ **STATUS BEFORE INITIALIZATION**

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

⑤ **POWER ON/OFF SEQUENCE**

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

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

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

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

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

- **The information in this document is current as of March, 2007. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC 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 NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".
  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.
  "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.
  "Special": 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, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)
(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02. 11-1

INTRODUCTION

Cautions 1. Download the program used in this manual from the page of Programming Examples (http://www.necel.com/micro/en/designsupports/sampleprogram/index.html) in the NEC Electronics Website (http://www.necel.com/).

2. This sample program is provided for reference purposes only and operations are therefore not subject to guarantee by NEC Electronics Corporation. When using this sample program, customers are kindly advised to sufficiently evaluate this product based on their system before usage.

3. When using this sample program, reference the following startup file and link directive file and adjust them if as necessary.

- **Startup file: JG2_start.s**
- **Link directive file: JG2_link.dir**

Readers

This manual is intended for users who understand the V850ES/JG2 (μPD70F3715, 70F3716, 70F3717, 70F3718, 70F3719) and V850ES/JJ2 (μPD70F3720, 70F3721, 70F3722, 70F3723, 70F3724) functions and who design application systems using these products.

Purpose

This manual is intended to give users an understanding of the basic functions of the V850ES/JG2 and V850ES/JJ2, using the application programs.

How to Read This Manual

It is assumed that the readers of this manual have general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

To understand the details of hardware functions

→ Refer to the **V850ES/JJ2 Hardware User's Manual** and **V850ES/JG2 Hardware User's Manual**.

To understand the details of an instruction function

→ Refer to the **V850ES Architecture User's Manual**.

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	$\overline{\times\times\times}$ (overscore over pin or signal name)
Memory map address:	High order at high stage and low order at low stage
Note:	Footnote for item marked with **Note** in the text.
Caution:	Information requiring particular attention.
Remark:	Supplementary information
Numeric representation:	Binary… ××××× or ××××B
	Decimal… ×××××
	Hexadecimal… ××××H

Suffix representing an exponent of 2 (in address space or memory capacity)

K (Kilo): $2^{10} = 1024$

M (Mega): $2^{20} = 1024^2$

G (Giga): $2^{30} = 1024^3$

The function lists are structured as follows.

Hardware name (symbol)

[Function]	Function description
[Function name]	Name of sample function
[Argument(s)]	Type and overview of argument(s)
[Processing content]	Processing content of sample function
[Starting method]	Conditions for calling a function
[SFR(s) used]	Register name and setting content
[call function(s)]	Name and function of call function(s)
[Variable(s)]	Type, name, and overview of variable(s) used in sample function
[Interrupt(s)]	Name of function
[Interrupt source(s)]	Name
[File name]	Name of corresponding sample program file
[Caution(s)]	Caution(s) upon function usage

Interrupt function

[Function name]	Name of interrupt function
[Overview]	Processing content
[Factor(s)]	Name of interrupt and conditions for occurrence
[call function(s)]	None
[Variable(s)]	Name of variable, function
[File name]	Name of corresponding sample program file
[Caution(s)]	None

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

Documents related to V850ES/JG2, V850ES/JJ2

Document Name	Document No.
V850ES Architecture User's Manual	U15943E
V850ES/JG2 Hardware User's Manual	U17715E
V850ES/JJ2 Hardware User's Manual	U17714E
V850ES/JG2, V850ES/JJ2 Application Note	This manual

Documents related to development tools (user's manuals)

Document Name		Document No.
QB-V850ESSX2 In-Circuit Emulator		U17091E
QB-V850MINI On-Chip Debug Emulator		U17638E
QB-MINI2 On-Chip Debug Emulator with Programming Function		U18371E
CA850 Ver. 3.00 C Compiler Package	Operation	U17293E
	C Language	U17291E
	Assembly Language	U17292E
	Link Directive	U17294E
PM+ Ver. 6.30 Project Manager		U18416E
ID850QB Ver. 3.20 Integrated Debugger	Operation	U17964E
SM850 Ver. 2.50 System Simulator	Operation	U16218E
SM850 Ver. 2.00 or Later System Simulator	External Part User Open Interface Specifications	U14873E
SM+ System Simulator	Operation	U18601E
	User Open Interface	U18212E
RX850 Ver. 3.20 Real-time OS	Basic	U13430E
	Installation	U17419E
	Technical	U13431E
	Task Debugger	U17420E
RX850 Pro Ver. 3.20 Real-time OS	Basic	U13773E
	Installation	U17421E
	Technical	U13772E
	Task Debugger	U17422E
AZ850 Ver. 3.30 System Performance Analyzer		U17423E
PG-FP4 Flash Memory Programmer		U15260E

CONTENTS

CHAPTER 1 INITIALIZATION PROCESSING 10

CHAPTER 2 CLOCK GENERATION FUNCTIONS 13
- 2.1 PLL Mode 13
- 2.2 Clock Monitor Mode 16

CHAPTER 3 16-BIT TIMER/EVENT COUNTER P (TMP) 19
- 3.1 Interval Timer Mode 19
- 3.2 External Event Count Mode 24
- 3.3 External Trigger Pulse Output Mode 28
- 3.4 One-Shot Pulse Output Mode 34
- 3.5 PWM Output Mode 40
- 3.6 Free-Running Timer Mode 47
- 3.7 Pulse Width Measurement Mode 54

CHAPTER 4 16-BIT TIMER/EVENT COUNTER Q (TMQ) 60
- 4.1 Interval Timer Mode 60
- 4.2 External Event Count Mode 65
- 4.3 External Trigger Pulse Output Mode 69
- 4.4 One-Shot Pulse Output Mode 75
- 4.5 PWM Output Mode 81
- 4.6 Free-Running Timer Mode 87
- 4.7 Pulse Width Measurement Mode 94

CHAPTER 5 16-BIT INTERVAL TIMER M (TMM) 100
- 5.1 Interval Timer Mode 100

CHAPTER 6 WATCH TIMER FUNCTION 104
- 6.1 Watch Timer 104
- 6.2 Interval Timer 108

CHAPTER 7 WATCHDOG TIMER FUNCTION 112
- 7.1 Reset Mode 112
- 7.2 Non-Maskable Interrupt Request Mode 114

CHAPTER 8 A/D CONVERTER 117
- 8.1 External Trigger Mode (Continuous Scan Mode) 117
- 8.2 Software Trigger Mode (Continuous Select Mode) 123
- 8.3 Software Trigger Mode (One-Shot Scan Mode) 127
- 8.4 Timer Trigger Mode (One-Shot Select Mode) 133

CHAPTER 9 D/A CONVERTER 137
- 9.1 Normal Mode 137
- 9.2 Real-Time Output Mode 140

CHAPTER 10 ASYNCHRONOUS SERIAL INTERFACE A (UARTA) 143
- 10.1 Asynchronous Serial Interface A (UARTAn) (n = 0 to 2) 143

CHAPTER 11 3-WIRE VARIABLE-LENGTH SERIAL I/O (CSIB) 151
11.1 Continuous Transfer Mode (Master Mode, Transmission/Reception Mode) 151
11.2 Continuous Transfer Mode (Slave Mode, Transmission/Reception Mode) 158
11.3 Single Transfer Mode (Master Mode, Transmission Mode) 165
11.4 Single Transfer Mode (Slave Mode, Reception Mode) 172

CHAPTER 12 INTERRUPT/EXCEPTION PROCESSING FUNCTION 178
12.1 External Interrupt 178

CHAPTER 13 STANDBY FUNCTION 183
13.1 HALT Mode 183
13.2 IDLE Mode 186
13.3 STOP Mode 191

CHAPTER 14 LOW VOLTAGE DETECTOR (LVI) 195
14.1 Internal Reset Mode 195
14.2 Internal Interrupt Mode 197

CHAPTER 1 INITIALIZATION PROCESSING

[Function]	Executes an initialization required to be executed first by the CPU.
[Function name]	JG2_init_main
[Argument]	None
[Processing content]	Sets the VSWC, OCDM, and WDTM2 registers.
[SFR used]	None
[call function]	JG2_init_set
[Variable]	None
[Interrupt]	None
[Interrupt source]	None
[File name]	JG2_init.c
[Caution]	Follow the specific sequence when setting a value to the special register.

[Function name]	JG2_init_set
[Processing content]	Executes register settings required to be executed first by the CPU.
[SFRs used]	DCHC0.E00: 0 (Disables DMA0 transfer.) DCHC1.E11: 0 (Disables DMA1 transfer.) DCHC2.E22: 0 (Disables DMA2 transfer.) DCHC3.E33: 0 (Disables DMA3 transfer.) PRCMD: 0x01 (Writes to the command register.) OCDM: 0x01 (Enables on-chip debug mode as operation mode.) WDTM2: 0x00 (Stops watchdog timer operation.)
[call function]	None
[Variable]	None
[File name]	JG2_init.c
[Caution]	• A DMA transfer is disabled in this sample program, since a DMA transfer must be disabled before setting data to a special register.

Figure 1-1. Initialization Processing (1)

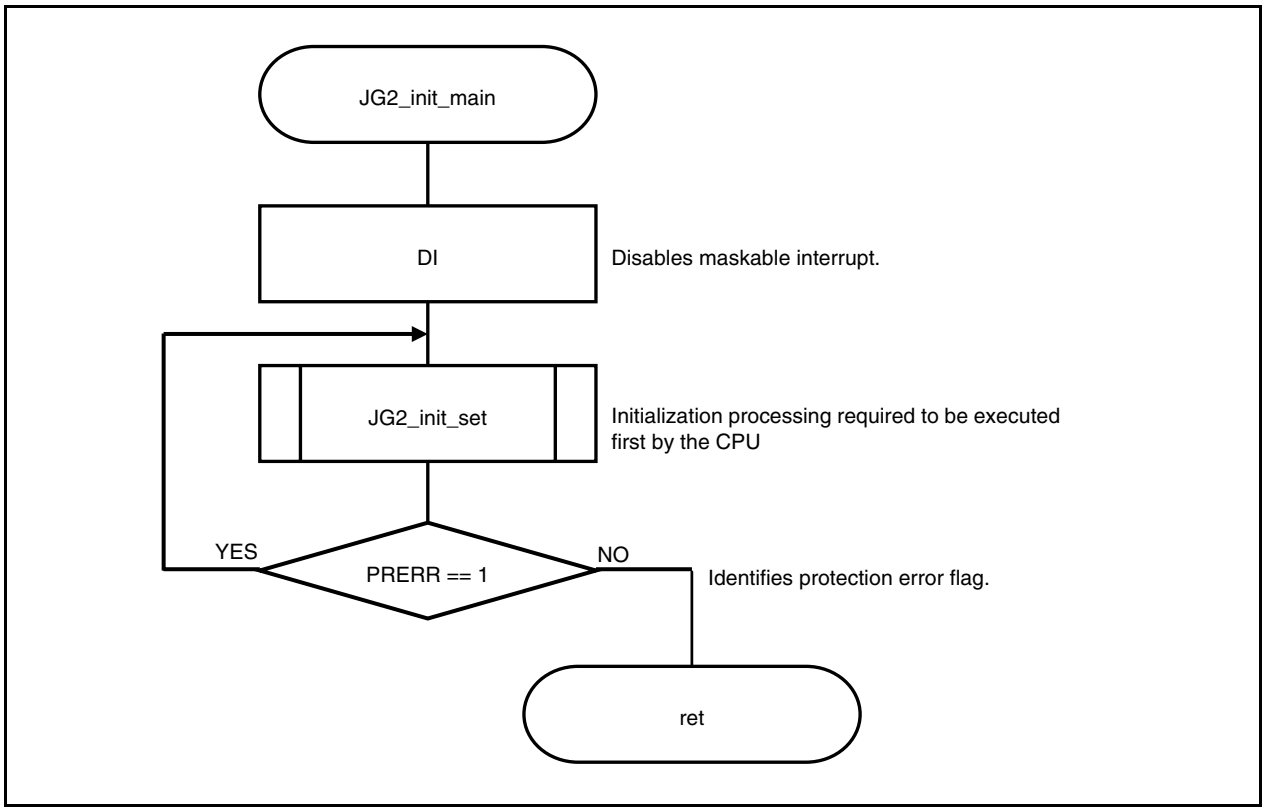
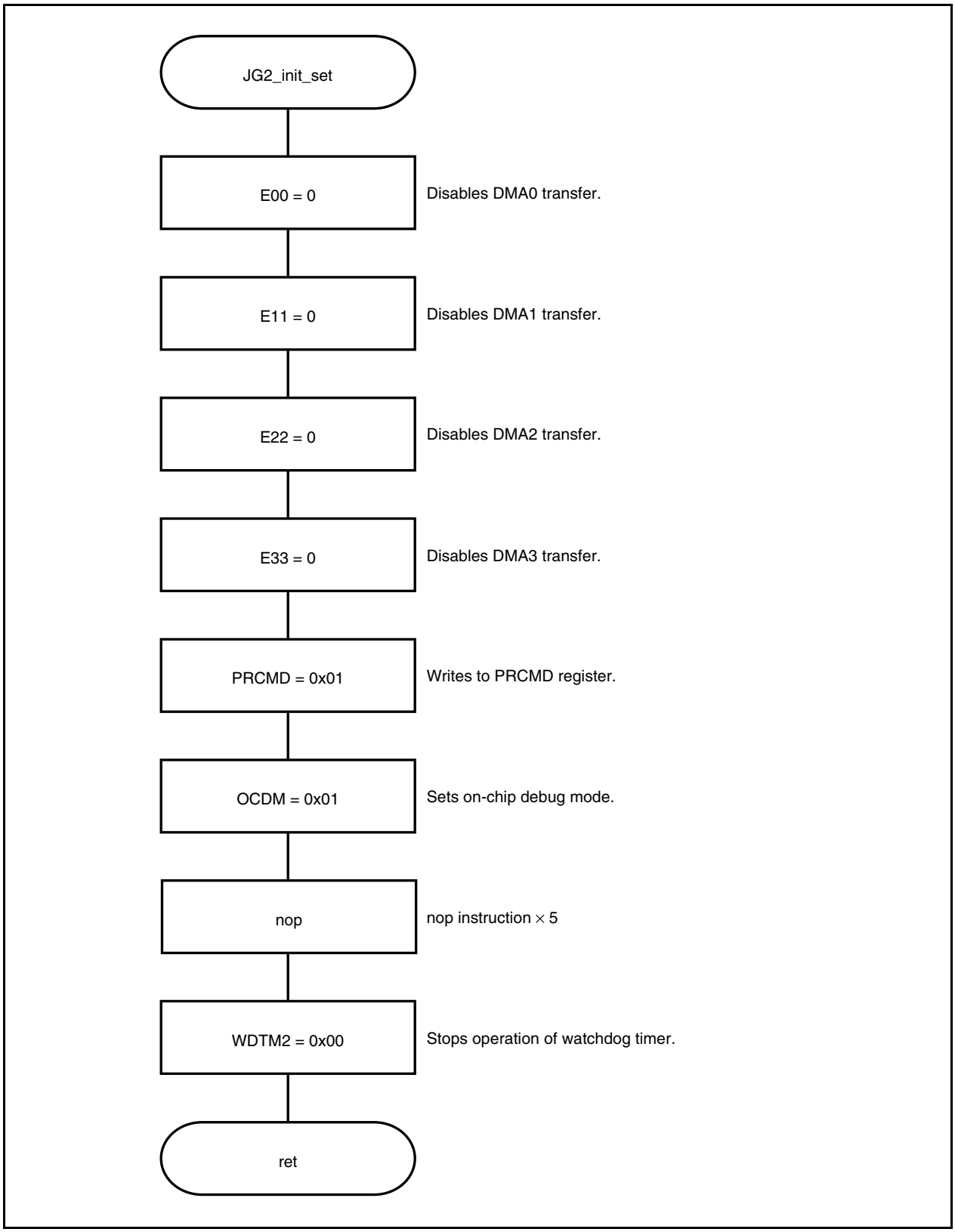


Figure 1-2. Initialization Processing (2)



CHAPTER 2 CLOCK GENERATION FUNCTIONS

2.1 PLL Mode

[Function]	Sets the clock frequency by setting the CPU operation clock (PLL mode) and the PCC register which is a special register.
[Function name]	clock_pll_main
[Argument]	None
[Processing content]	Sets PLLCTL and PCC registers and uses PLL.
[Starting method]	None
[SFR used]	None
[call function]	clock_pll_mode, clock_pcc_mode
[Variable]	None
[Interrupt]	None
[Interrupt source]	None
[File name]	clock_pll\clock_pll.c
[Caution]	Note that if the clock_pll_main function is called, the maskable interrupt request will not be enabled (EI).

[Function name]	clock_pll_mode
[Processing content]	Sets to PLL mode by the PLLCTL register.
[SFR used]	PLLCTL: 0x03 (Sets PLL mode and PLL operation.)
[call function]	None
[Variable]	None
[File name]	clock_pll\clock_pll.c
[Caution]	None

[Function name]	clock_pcc_mode
[Processing content]	Sets the PCC register by disabling DMA transfer.
[SFRs used]	DCHC0.E00: 0 (Disables DMA0 transfer.) DCHC1.E11: 0 (Disables DMA1 transfer.) DCHC2.E22: 0 (Disables DMA2 transfer.) DCHC3.E33: 0 (Disables DMA3 transfer.) PRCMD: 0x00 (Writes to command register.) PCC: 0x00 (Selects f_{XX} as clock.)
[call function]	None
[Variable]	None
[File name]	clock_pll\clock_pll.c
[Cautions]	• DMA transfer is disabled in this sample program, because all DMA transfers must be disabled before performing data setting to the special register. • Set the PCC register after switching to the PLL mode. The PCC register is a special register and can therefore only be written in a combination of specific sequences.

Figure 2-1. PLL Mode (1)

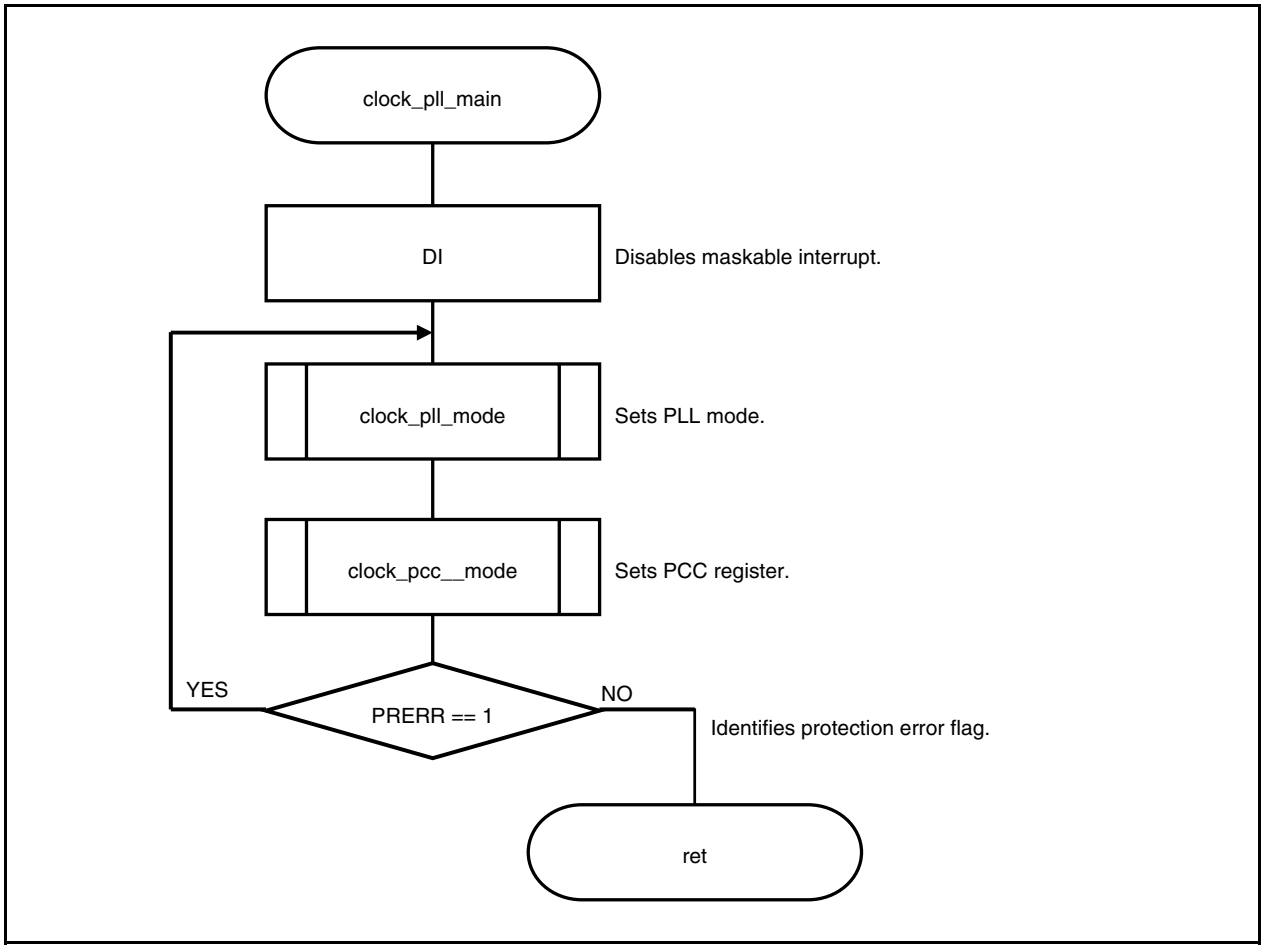
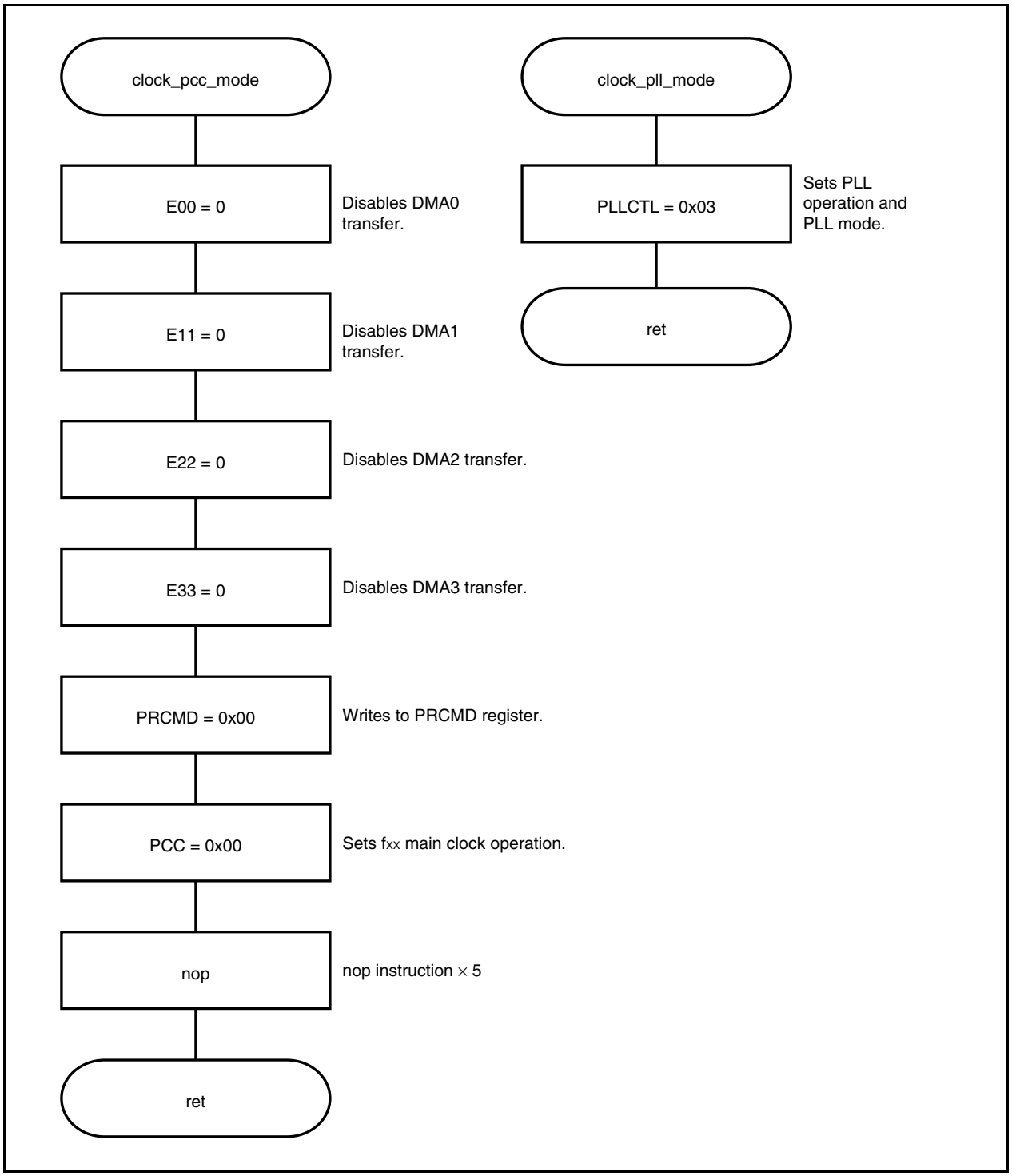


Figure 2-2. PLL Mode (2)



2.2 Clock Monitor Mode

[Function]	Clock monitor operation
[Function name]	clock_monitor_main
[Arguments]	None
[Processing content]	Enables operation of the clock monitor mode by calling the clock monitor mode function.
[Starting method]	None
[SFR used]	None
[call function]	clock_monitor_mode
[Variable]	None
[Interrupt]	None
[Interrupt source]	None
[File name]	clock_monitor\clock_monitor.c
[Caution]	Once CLM.CLME bit is set to 1, it can be cleared to 0 only by reset.

[Function name]	clock_monitor_mode
[Processing content]	Disables DMA transfer and sets the clock monitor operation mode by the CLM register.
[SFRs used]	DCHC0.E00: 0 (Disables DMA0 transfer.) DCHC1.E11: 0 (Disables DMA1 transfer.) DCHC2.E22: 0 (Disables DMA2 transfer.) DCHC3.E33: 0 (Disables DMA3 transfer.) PRCMD: 0x01 (Writes to command register.) CLM: 0x01 (Enables operation.)
[call function]	None
[Variable]	None
[File name]	clock_monitor\clock_monitor.c
[Cautions]	• DMA transfer is disabled in this sample program, because all DMA transfers must be disabled before performing data setting to the special register. • The CLM register is a special register and can therefore only be written in a combination of specific sequences.

Figure 2-3. Clock Monitor Mode (3)

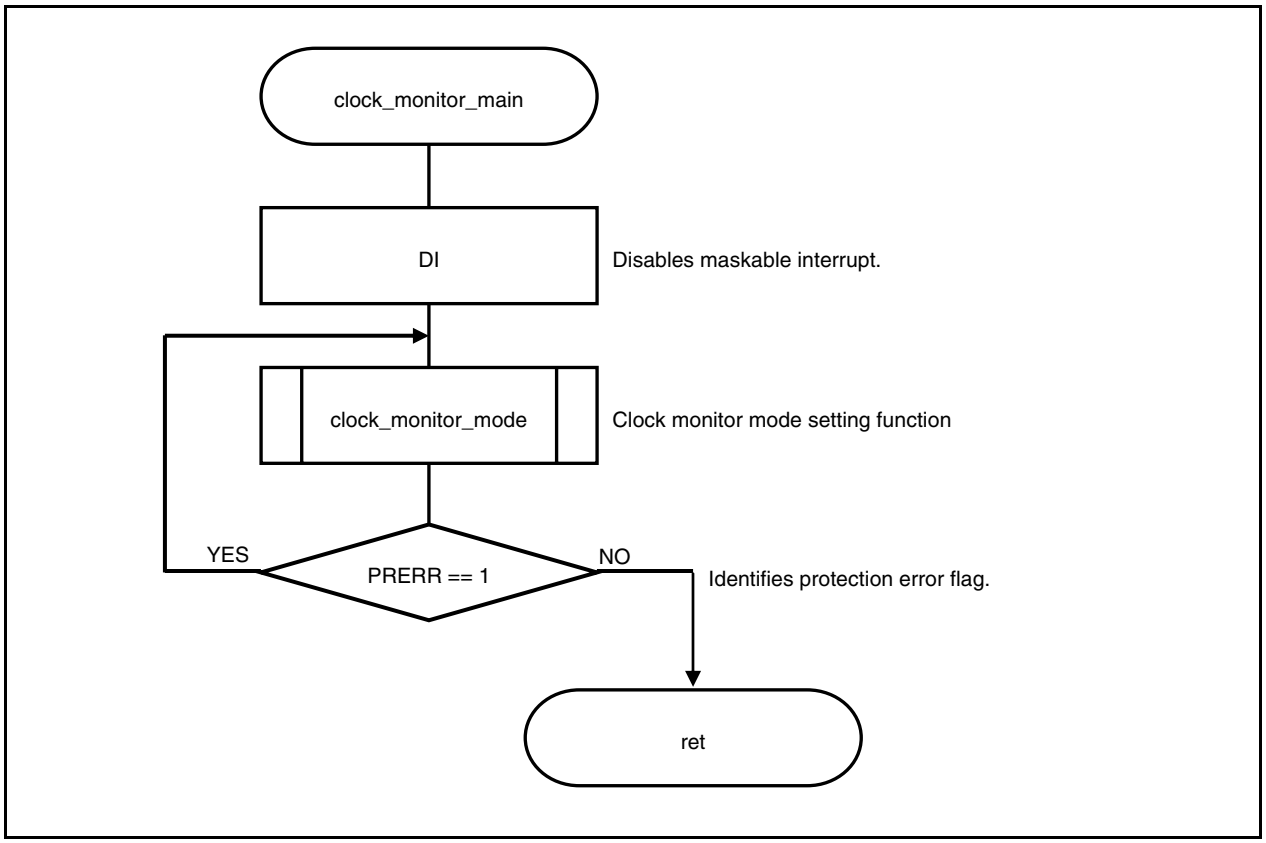
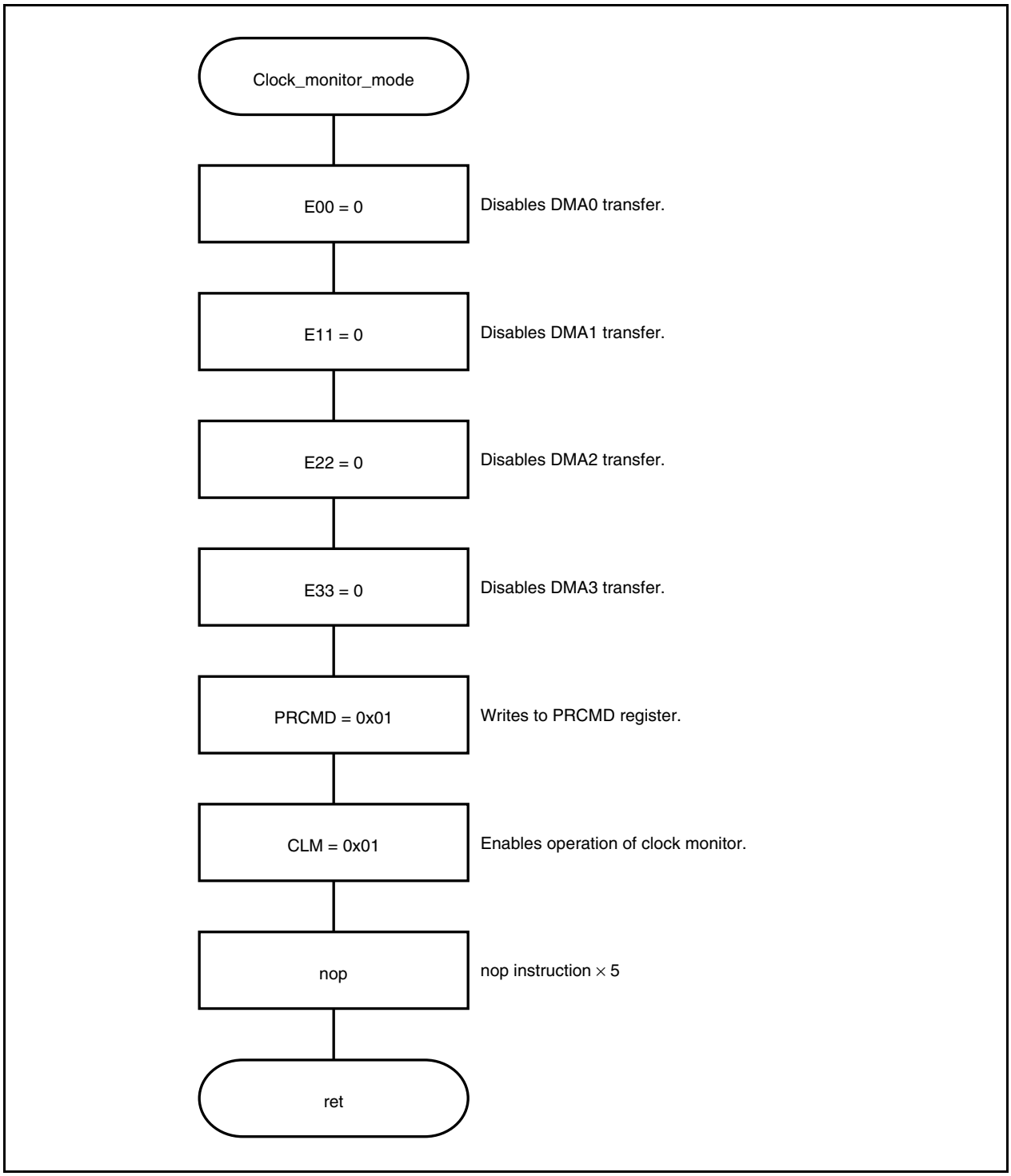


Figure 2-4. Clock Monitor Mode (4)



CHAPTER 3 16-BIT TIMER/EVENT COUNTER P (TMP)

3.1 Interval Timer Mode

[Function]	Starts 16-bit counter operation by setting the TP0CE bit. Outputs an interrupt request signal (INTTP0CC0) at an interval set by the TP0CCR0 register. Inverts the TOP00 pin output when the value set by the TP0CCR0 register and the count value of the 16-bit counter match. Can be implemented with TMP0 to TMP5.
[Function name]	timerp_interval_main
[Argument]	None
[Processing content]	Performs count operation of an fxx/32 count clock, generates an interrupt by inverting the TOP00 pin output upon the count subsequent to the count whose value matches the value of the TP0CCR0 register. The TOP00 pin starts output at high level.
[SFR used]	None
[call function]	timerp_port_set, timerp_interval_set, timerp_interval_start
[Variable]	None
[Interrupt]	timerp_TP0CC0_int
[Interrupt source]	INTTP0CC0
[File name]	timerp_interval\timerp_interval.c
[Caution]	None

[Function name]	timerp_port_set
[Argument]	None
[Processing content]	Sets port 3 to TOP00 output pin.
[SFRs used]	PFC3L: 0x04 (Sets to TOP00 output pin.) PFCE3L: 0x04 (Sets to TOP00 output pin.) PMC3L: 0x04 (Sets to TOP00 output pin.)
[call function]	None
[Variable]	None
[File name]	timerp_interval\timerp_interval.c
[Caution]	None

[Function name]	timerp_interval_set
[Argument]	None
[Processing content]	Sets TMP0 control register.
[SFRs used]	TP0CTL0.TP0CE: 0 (Disables TMP0 operation.) TP0CTL0: 0x05 (Sets count clock to fxx/32.) TP0CTL1: 0x00 (Sets timer mode to interval timer mode.) TP0IOC0: 0x01 (Sets TOP00 pin output level to high-level start, and enables timer output.) TP0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TP0CCMK0: 0 (Clears TMP0 capture 0/compare 0 match interrupt mask.)
[call function]	None
[Variable]	None
[File name]	timerp_interval\timerp_interval.c
[Caution]	None

The interval time can be calculated by the following formula.

Interval = (Set value of TP0CCR0 register + 1) × Count clock cycle

[Function name]	timerp_interval_start
[Argument]	None
[Processing content]	Starting function of interval timer mode
[Starting method]	Call this function after calling the timerp_interval_set function.
[SFR used]	TP0CTL0.TP0CE: 1 (Enables TMP0 operation)
[call function]	None
[Variable]	None
[File name]	timerp_interval\timerp_interval.c
[Caution]	None

Interrupt function

[Function name]	timerp_TP0CC0_int	
[Overview]	Defined by the user.	
[Factor]	INTTP0CC0	Match between the count value of the 16-bit counter and TP0CCR0
[call function]	None	
[Variable]	None	
[File name]	timerp_interval\timerp_interval .c	
[Caution]	None	

Figure 3-1. Interval Timer Mode (1)

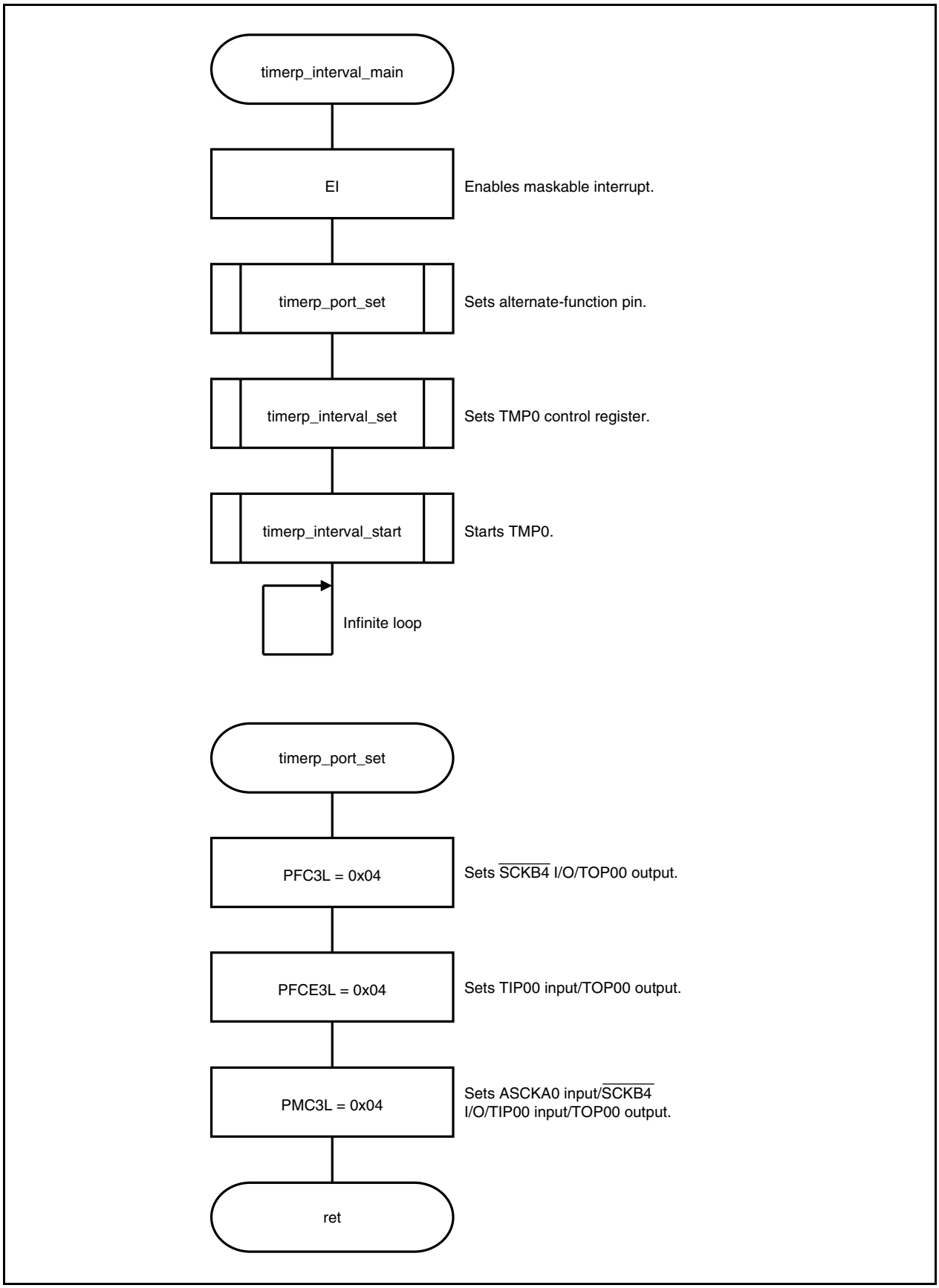
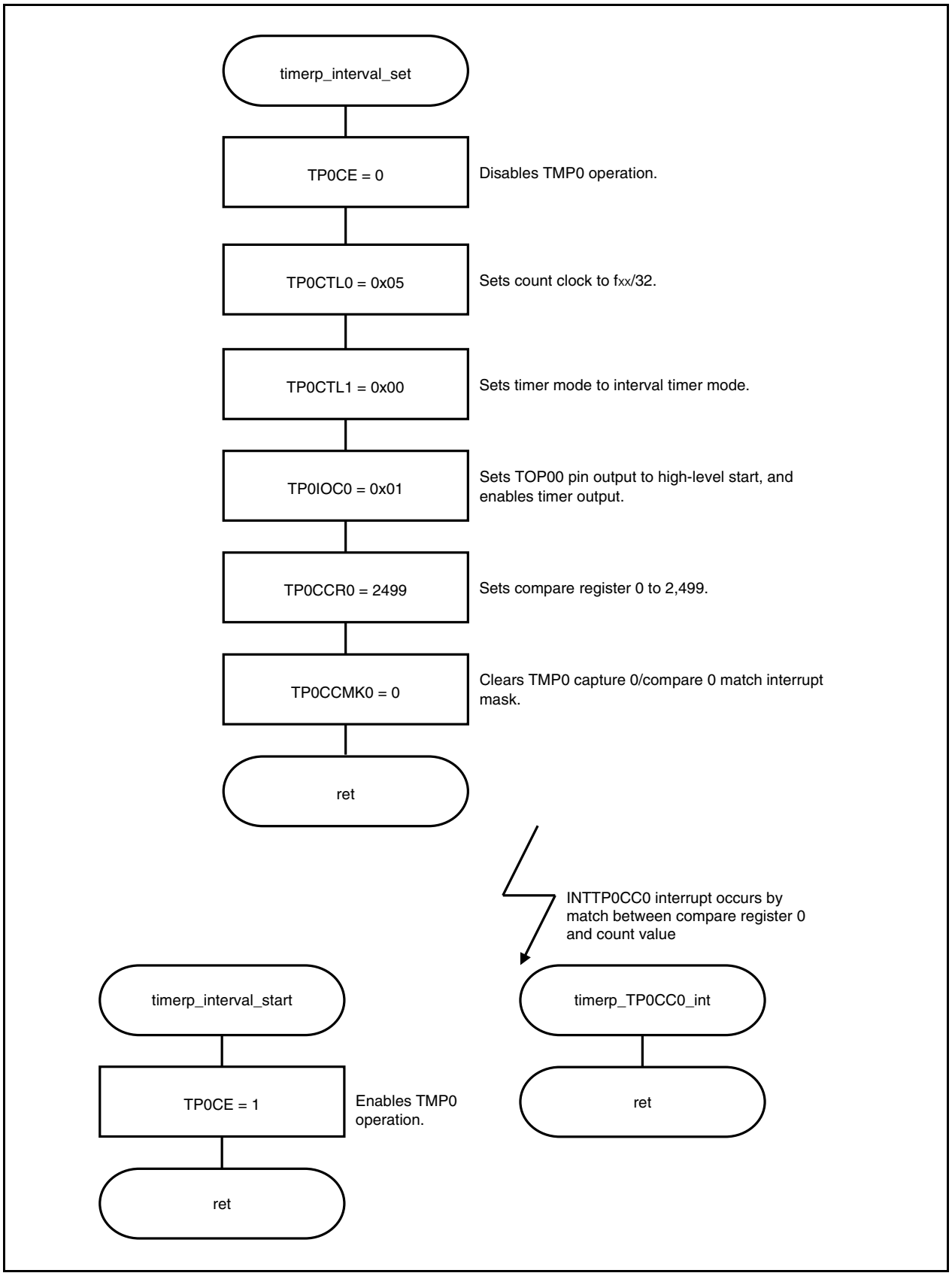


Figure 3-2. Interval Timer Mode (2)



3.2 External Event Count Mode

[Function]	Counts the valid edge of the external event count input (TIP00 pin) and generates an interrupt request signal (INTTP0CC0) upon a match between the count value set by the TP0CCR0 register and the count value. (Clears the 16-bit counter simultaneously.) Can be implemented with TMP0 to TMP5.
[Function name]	timerp_event_count_main
[Argument]	None
[Processing content]	Counts the valid edge of the external event count input, generates an interrupt upon the count (value set to TP0CCR0 register + 1) subsequent to the count whose value matches the value of the TP0CCR0 register, and clears the counter.
[SFR used]	None
[call function]	timerp_port_set, timerp_event_count_set, timerp_event_count_start
[Variable]	None
[Interrupt]	timerp_TP0CC0_int
[Interrupt source]	INTTP0CC0
[File name]	timerp_event_count\timerp_event_count.c,
[Caution]	None

[Function name]	timerp_port_set
[Argument]	None
[Processing content]	Sets port 3 to TIP00 input pin.
[SFRs used]	PFCE3L: 0x04 (Sets TIP00 input pin.) PMC3L: 0x04 (Sets TIP00 input pin.)
[call function]	None
[Variable]	None
[File name]	timerp_interval\timerp_event_count.c
[Caution]	None

[Function name]	timerp_event_count_set
[Argument]	None
[Processing content]	Sets TMP0 control register.
[SFRs used]	TP0CTL0.TP0CE: 0 (Disables TMP0 operation.) TP0CTL0: 0x00 (Sets count clock to fxx.) TP0CTL1: 0x01 (Sets timer mode to external event count mode.) TP0IOC2: 0x08 (Sets TIP00 pin valid edge to falling edge.) TP0CCR0: 40 (Sets 16-bit counter compare register 0 to 40.) TP0CCMK0: 0 (Clears TMP0 capture 0/compare 0 match interrupt mask.)
[call function]	None
[Variable]	None
[File name]	timerp_event_count\timerp_event_count.c
[Caution]	None

[Function name]	timerp_event_count_start
[Argument]	None
[Processing content]	Starting function of external event count mode.
[Starting method]	Call this function after calling the timerp_event_count_set function.
[SFR used]	TP0CTL0.TP0CE: 1 (Enables TMP0 operation.)
[call function]	None
[Variable]	None
[File name]	timerp_event_count\ timerp_event_count.c
[Caution]	None

Interrupt function

[Function name]	timerp_TP0CC0_int
[Overview]	Defined by the user.
[Factor]	INTTP0CC0 Match between the count value of the 16-bit counter and TP0CCR0.
[call function]	None
[Variable]	None
[File name]	timerp_event_count\ timerp_event_count.c
[Caution]	None

Figure 3-3. External Event Count Mode (1)

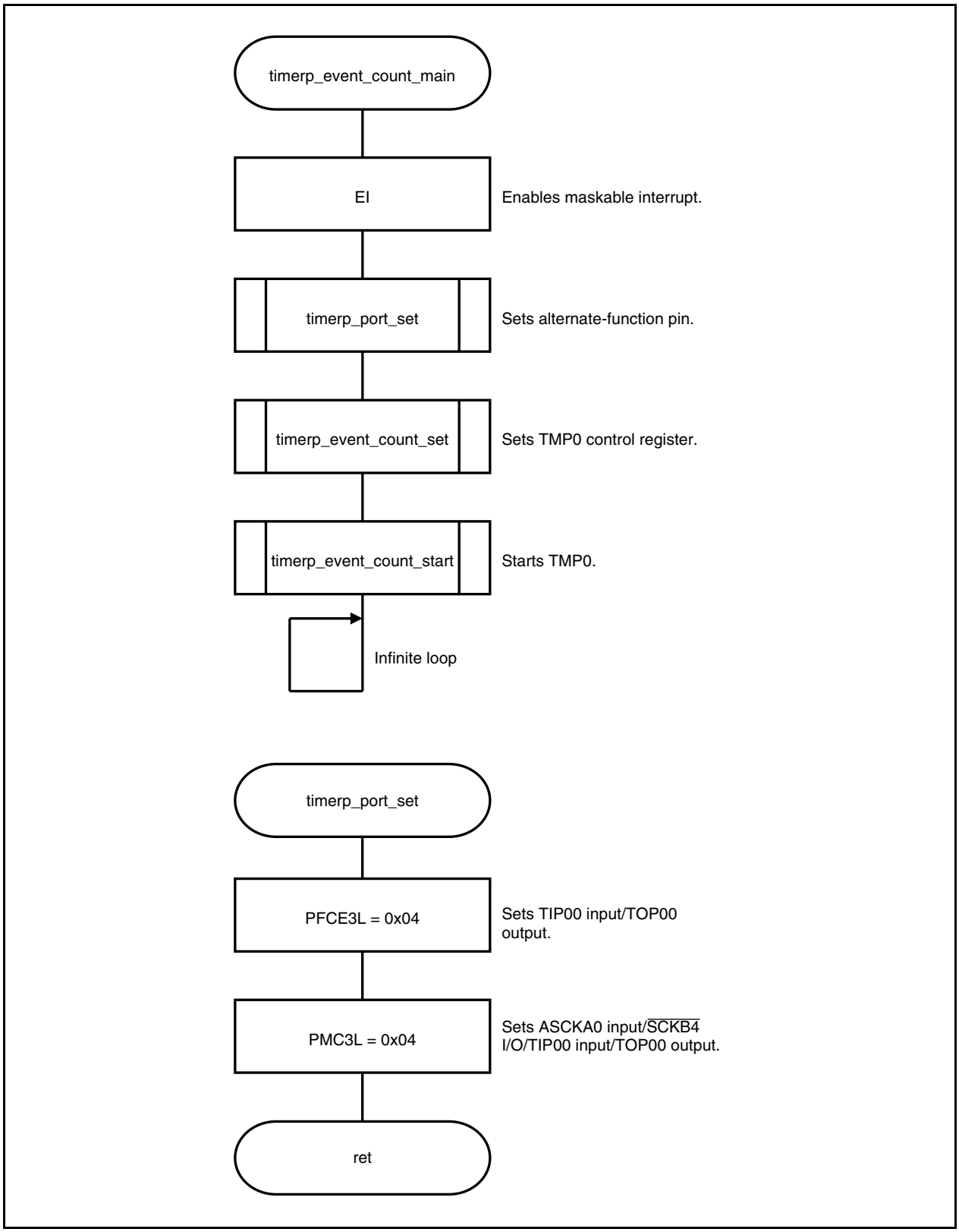
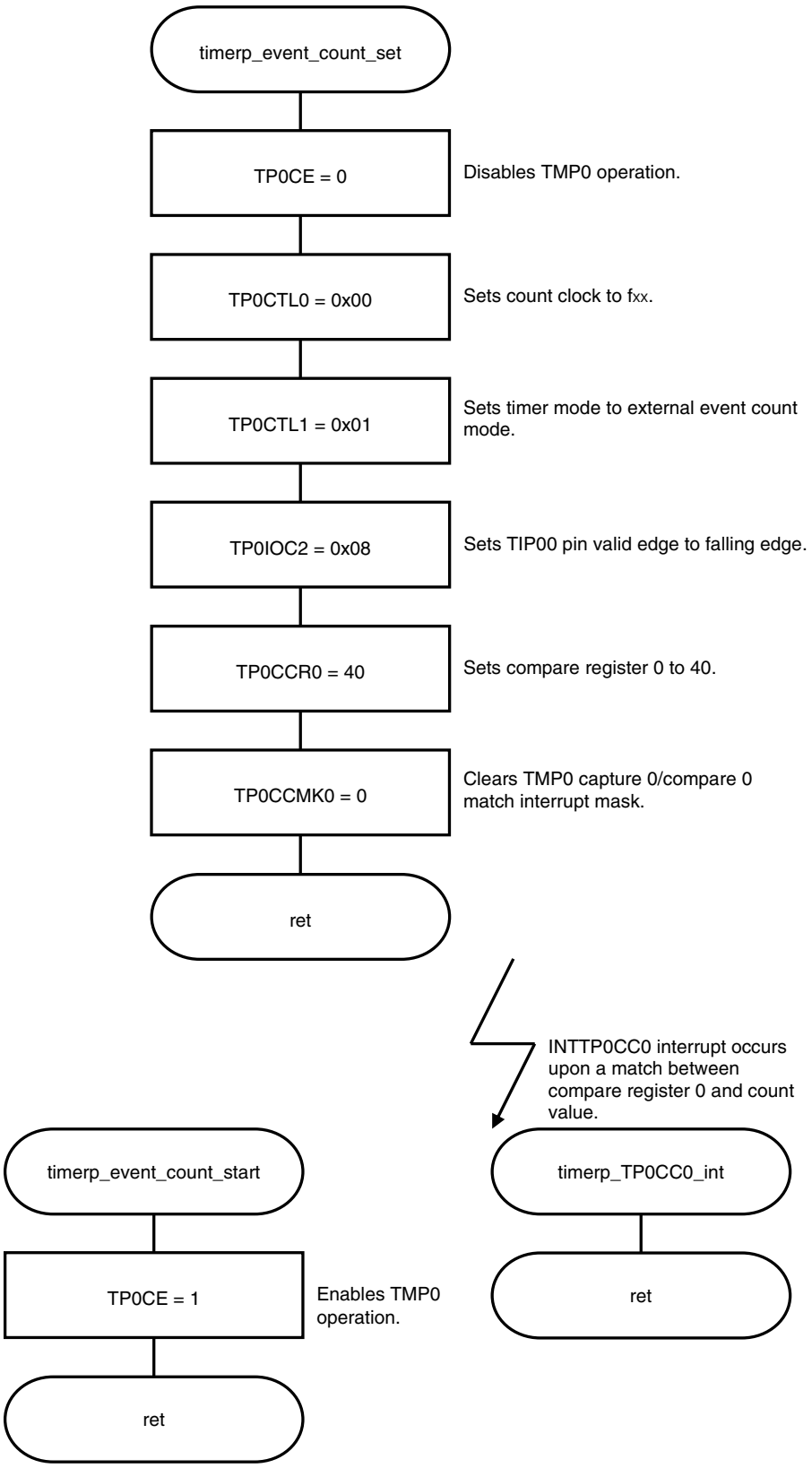


Figure 3-4. External Event Count Mode (2)



3.3 External Trigger Pulse Output Mode

[Function]	Starts operation of the 16-bit counter when the valid edge of the external trigger input (TIP00) is detected. Clears the 16-bit counter upon a compare match with the TP0CCR0 register. Outputs the PWM waveform from the TOP01 pin upon a match between the value set by the TP0CCR1 register and the count value of the 16-bit counter. When the trigger is generated again, clears the counter to 0000H and restarts. Can be implemented with TMP0 to TMP5.
[Function name]	timerp_trigger_pulse_main
[Argument]	None
[Processing content]	Starts count operation of an fxx/32 count clock when the valid edge of the external trigger input is detected, generates an interrupt upon the count subsequent to the count whose value matches the value of the TP0CCR0 register, and clears the counter. Generates an interrupt by inverting the TOP01 pin output upon the count subsequent to the count whose value matches the value of the TP0CCR1 register. TOP01 pin starts output at high level.
[SFR used]	None
[call function]	timerp_port_set, timerp_trigger_pulse_set, timerp_trigger_pulse_start
[Variable]	None
[Interrupts]	timerp_TP0CC0_int timerp_TP0CC1_int
[Interrupt sources]	INTTP0CC0 INTTP0CC1
[File name]	timerp_trigger_pulse\timerp_trigger_pulse.c
[Caution]	None

[Function name]	timerp_port_set
[Argument]	None
[Processing content]	Sets port 3 to TOP01 output pin and TIP00 input pin.
[SFRs used]	PFC3L: 0x08 (Sets TOP01 output pin, TIP00 input pin.) PFCE3L: 0x04 (Sets TOP01 output pin, TIP00 input pin.) PMC3L: 0x0C (Sets TIP00 input pin, TOP01 output pin.)
[call function]	None
[Variable]	None
[File name]	timerp_trigger_pulse\timerp_ trigger_pulse.c
[Caution]	None

[Function name]	timerp_trigger_pulse_set
[Argument]	None
[Processing content]	Sets TMP0 control register.
[SFRs used]	TP0CTL0.TP0CE: 0 (Disables TMP0 operation.) TP0CTL0: 0x05 (Sets count clock to fxx/32.) TP0CTL1: 0x02 (Sets timer mode to external trigger pulse output mode.) TP0IOC0: 0x04 (Sets TOP01 pin output to high-level start, and enables timer output.) TP0IOC2: 0x08 (Sets TIP00 pin valid edge to falling edge.) TP0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TP0CCR1: 1249 (Sets 16-bit counter compare register 1 to 1,249.) TP0CCMK0: 0 (Clears TMP0 capture 0/compare 0 match interrupt mask.) TP0CCMK1: 0 (Clears TMP0 capture 1/compare 1 match interrupt mask.)
[call function]	None
[Variable]	None
[File name]	timerp_trigger_pulse\timerp_trigger_pulse.c
[Caution]	None

The active level width, cycle and duty factor of the PWM waveform can be calculated by the following formula.

Active level width = (Set value of TP0CCR1 register) × Count clock cycle

Cycle = (Set value of TP0CCR0 register + 1) × Count clock cycle

Duty factor = (Set value of TP0CCR1 register)/(Set value of TP0CCR0 register + 1)

[Function name]	timerp_trigger_pulse_start
[Argument]	None
[Processing content]	Starting function of external trigger pulse output mode.
[Starting method]	Call this function after calling the timerp_trigger_pulse_set function.
[SFRs used]	TP0CTL0.TP0CE: 1 (Enables TMP0 operation.)
[call function]	None
[Variable]	None
[File name]	timerp_trigger_pulse\timerp_trigger_pulse.c
[Caution]	None

Interrupt function

[Function name]	timerp_TP0CC0_int	
[Overview]	Defined by the user.	
[Factor]	INTTP0CC0	Match between the count value of the 16-bit counter and TP0CCR0.
[call function]	None	
[Variable]	None	
[File name]	timerp_trigger_pulse\timerp_trigger_pulse.c	
[Caution]	None	

[Function name]	timerp_TP0CC1_int	
[Overview]	Defined by the user.	
[Factor]	INTTP0CC1	Match between the count value of the 16-bit counter and TP0CCR1.
[call function]	None	
[Variable]	None	
[File name]	timerp_trigger_pulse\timerp_trigger_pulse.c	
[Caution]	None	

Figure 3-5. External Trigger Pulse Output Mode (1)

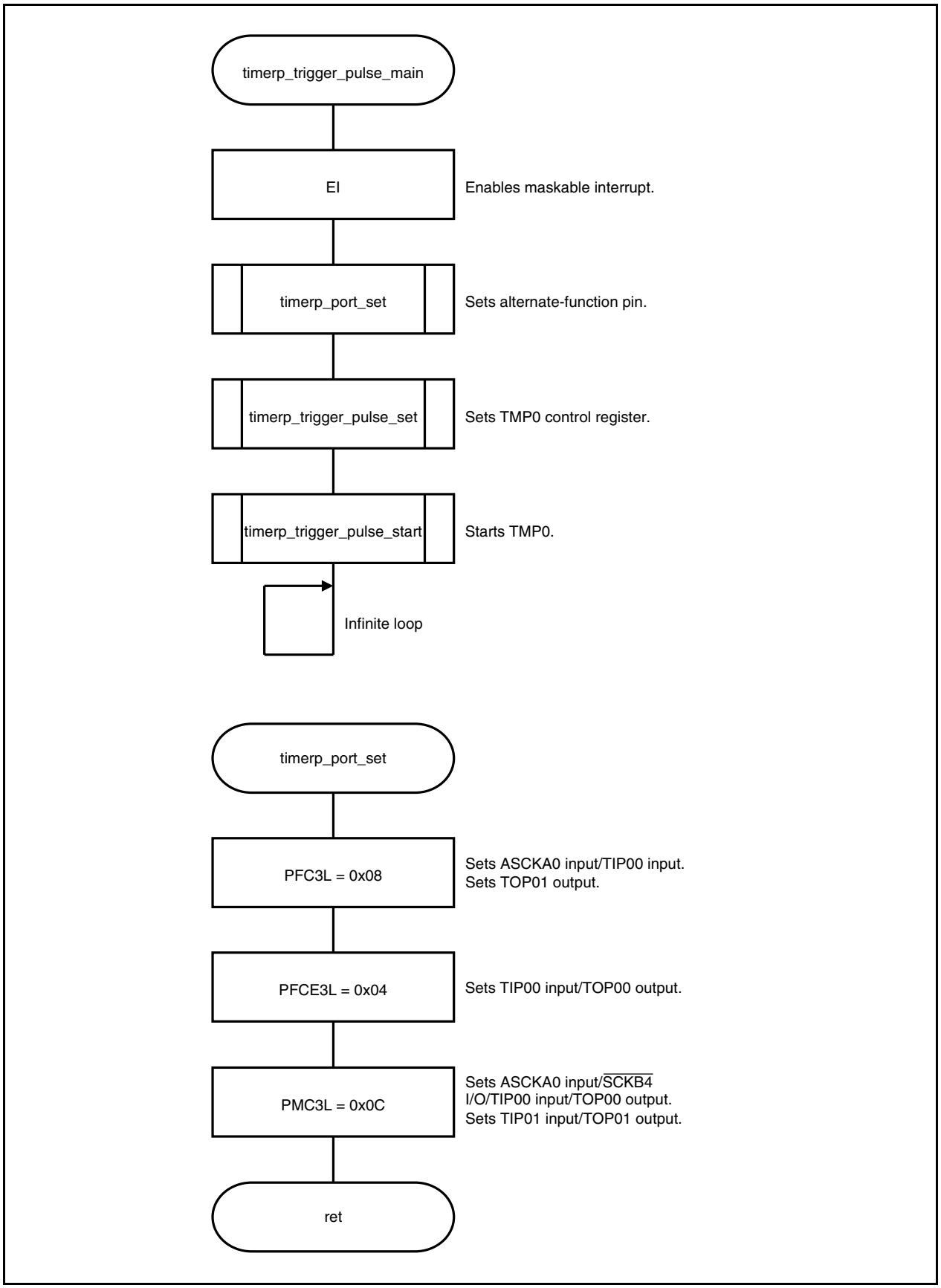


Figure 3-6. External Trigger Pulse Output Mode (2)

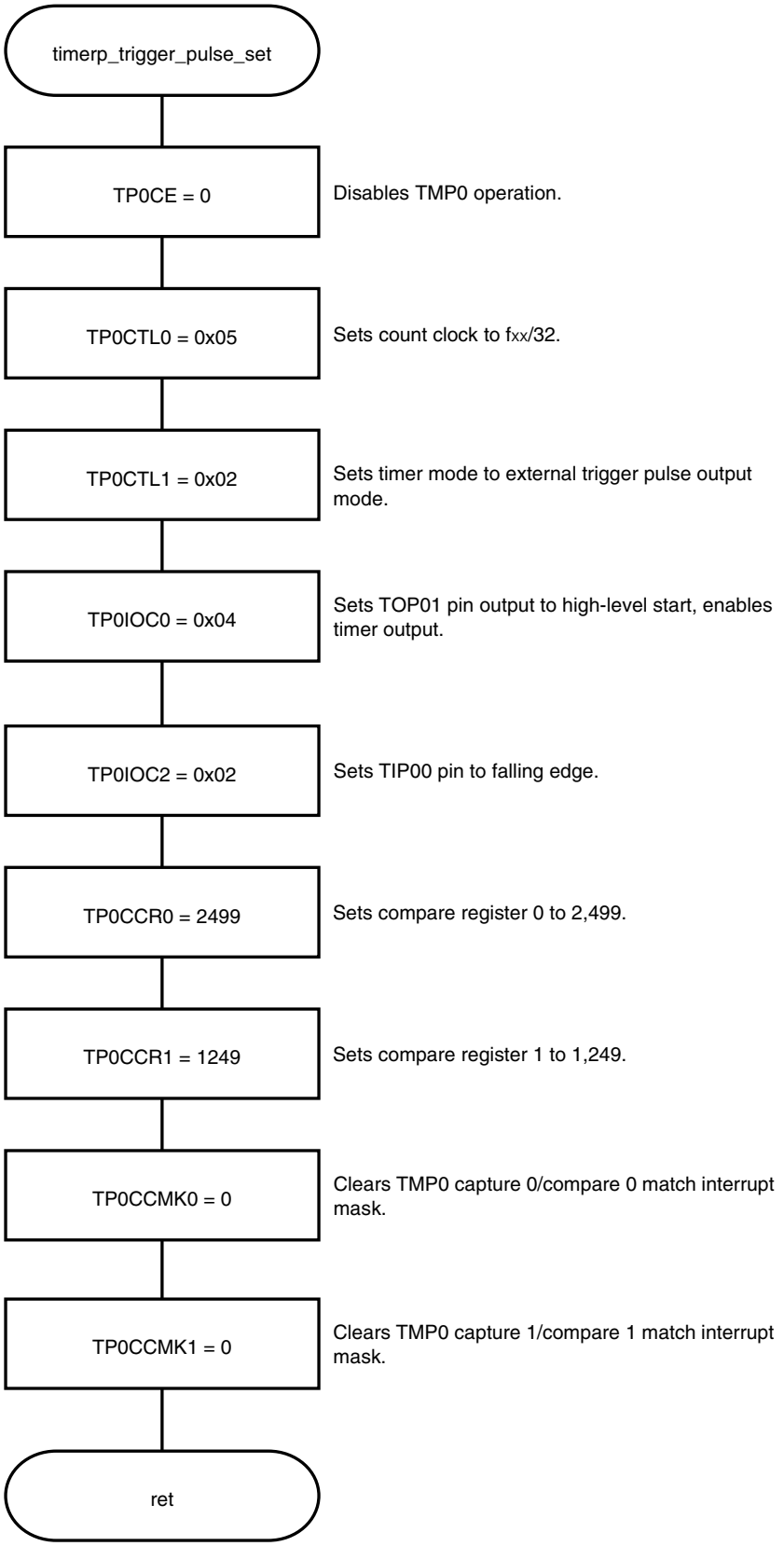
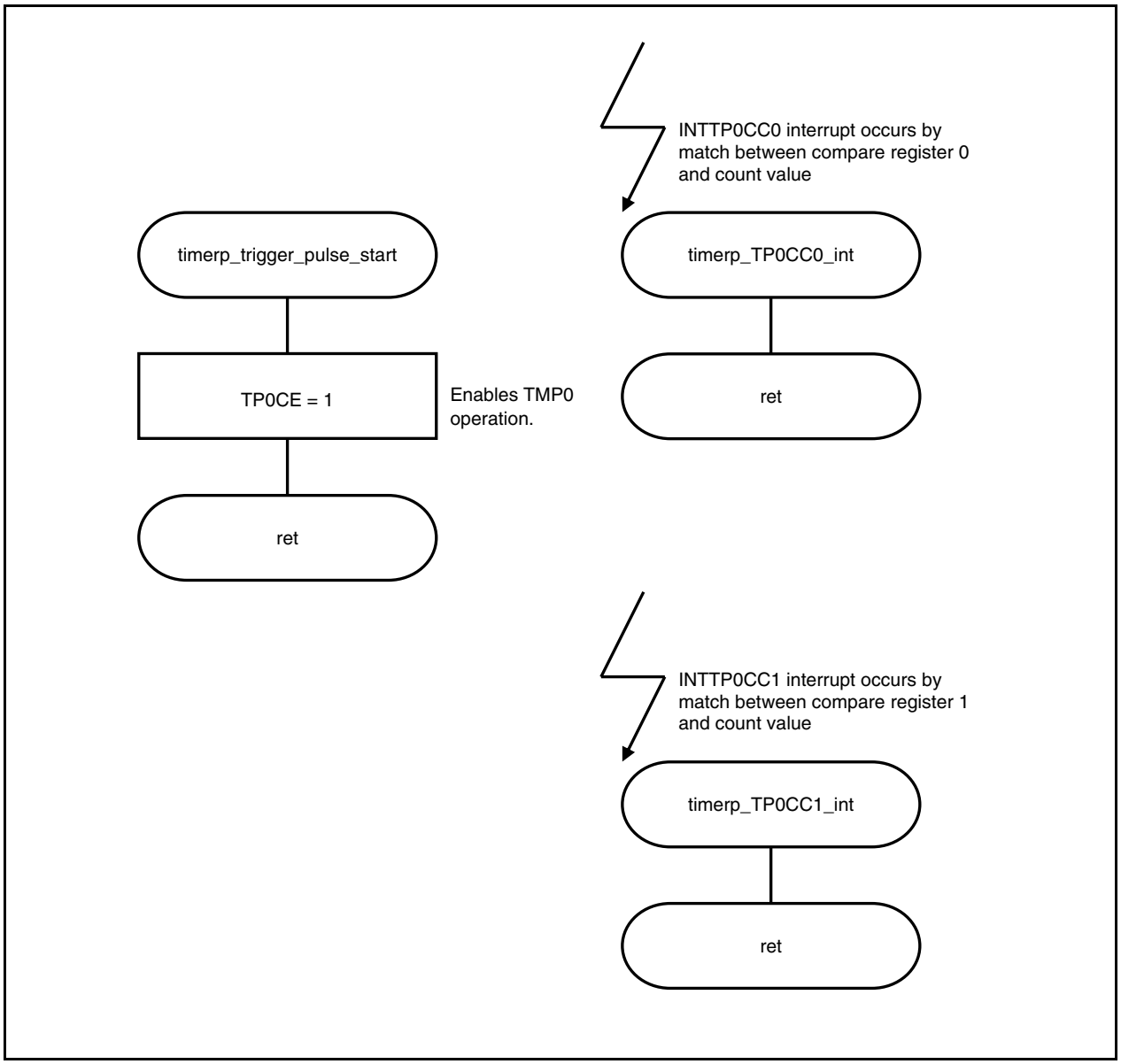


Figure 3-7. External Trigger Pulse Output Mode (3)



3.4 One-Shot Pulse Output Mode

[Function]	Starts operation of the 16-bit counter when the valid edge of the external trigger input (TIP00 pin) is detected. Activates the TOP01 pin output upon a match between the value set by the TP0CCR1 register and the count value of the 16-bit counter. Inverts the TOP01 pin output upon a match between the value set by the TP0CCR0 register and the count value of the 16-bit counter, and outputs a one-shot pulse. Stops the count after the one-shot pulse is output, and waits for a trigger. Ignores a trigger if it is generated again during one-shot pulse output. Can be implemented with TMP0 to TMP5.
[Function name]	timerp_1shot_pulse_main
[Argument]	None
[Processing content]	Starts count operation of an fxx/32 count clock upon valid edge detection of the external trigger input, generates an interrupt by inverting the TOP01 pin output upon the count subsequent to the count whose value matches the value of the TP0CCR0 register, clears the counter, and stops count operation. Generates an interrupt by inverting the TOP01 pin output upon the count subsequent to the count whose value matches the value of the TP0CCR1 register. TOP01 pin starts output at high level.
[SFRs used]	None
[call function]	timerp_port_set, timerp_1shot_pulse_set, timerp_1shot_pulse_start
[Variable]	None
[Interrupt]	timerp_TP0CC0_int timerp_TP0CC1_int
[Interrupt source]	INTTP0CC0 INTTP0CC1
[File name]	timerp_1shot_pulse\timerp_1shot_pulse.c,
[Caution]	None

[Function name]	timerp_port_set
[Argument]	None
[Processing content]	Sets port 3 to TOP01 output pin and TIP00 input pin.
[SFRs used]	PFC3L: 0x08 (Sets TOP01 output pin, TIP00 input pin.) PFCE3L: 0x04 (Sets TOP01 output pin, TIP00 input pin.) PMC3L: 0x0C (Sets TOP01 output pin, TIP00 input pin.)
[call function]	None
[Variable]	None
[File name]	timerp_1shot_pulse\timerp_1shot_pulse.c
[Caution]	None

[Function name]	timerp_1shot_pulse_set
[Argument]	None
[Processing content]	Sets TMP0 control register.
[SFRs used]	TP0CTL0.TP0CE: 0 (Disables TMP0 operation.) TP0CTL0: 0x05 (Sets count clock to fxx/32.) TP0CTL1: 0x03 (Sets timer mode to one-shot pulse mode.) TP0IOC0: 0x04 (Sets TOP01 pin output to high-level start, enables timer output.) TP0IOC2: 0x08 (Sets TIP00 pin valid edge to falling edge.) TP0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TP0CCR1: 1249 (Sets 16-bit counter compare register 1 to 1,249.) TP0CCMK0: 0 (Sets TMP0 capture 0/compare 0 match interrupt mask clear.) TP0CCMK1: 0 (Sets TMP0 capture 1/compare 1 match interrupt mask clear.)
[call function]	None
[Variable]	None
[File name]	timerp_1shot_pulse\timerp_1shot_pulse.c
[Caution]	None

The one-shot pulse output delay period and active level width can be calculated by the following formula.

Output delay period = (TP0CCR1 register set value) × count clock cycle
Active level width = (TP0CCR0 register set value – TP0CCR1 register set value + 1) × count clock cycle

[Function name]	timerp_1shot_pulse_start
[Argument]	None
[Processing content]	Starting function of one-shot pulse output mode
[Starting method]	Call this function after calling the timerp_1shot_pulse_set function.
[SFR used]	TP0CTL0.TP0CE: 1 (Enables TMP0 operation.)
[call function]	None
[Variable]	None
[File name]	timerp_1shot_pulse\timerp_1shot_pulse.c
[Caution]	None

Interrupt function

[Function name]	timerp_TP0CC0_int
[Overview]	Defined by the user.
[Factor]	INTTP0CC0 Match between the count value of the 16-bit counter and TP0CCR0
[call function]	None
[Variable]	None
[File name]	timerp_1shot_pulse\timerp_1shot_pulse.c
[Caution]	None

[Function name]	timerp_TP0CC1_int
[Overview]	Defined by the user.
[Factor]	INTTP0CC1 Match between the count value of the 16-bit counter and TP0CCR1
[call function]	None
[Variable]	None
[File name]	timerp_1shot_pulse\timerp_1shot_pulse.c
[Caution]	None

Figure 3-8. One-Shot Pulse Output Mode (1)

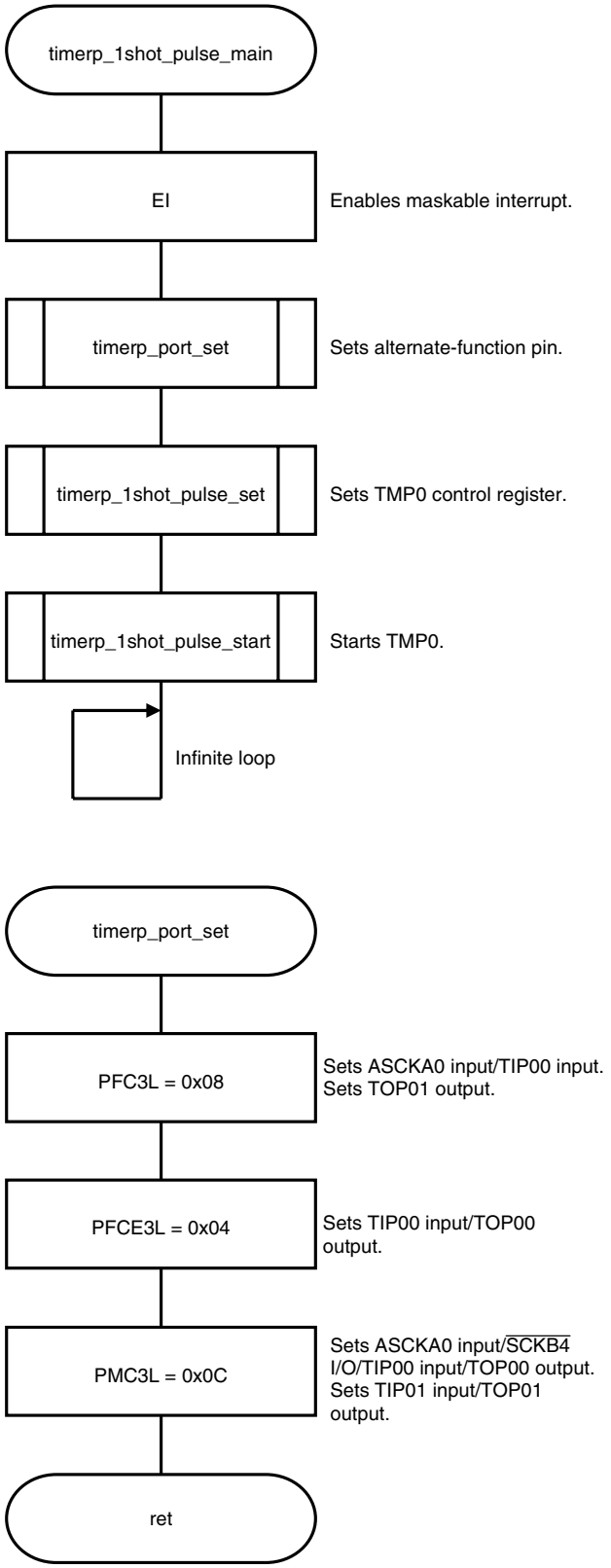


Figure 3-9. One-Shot Pulse Output Mode (2)

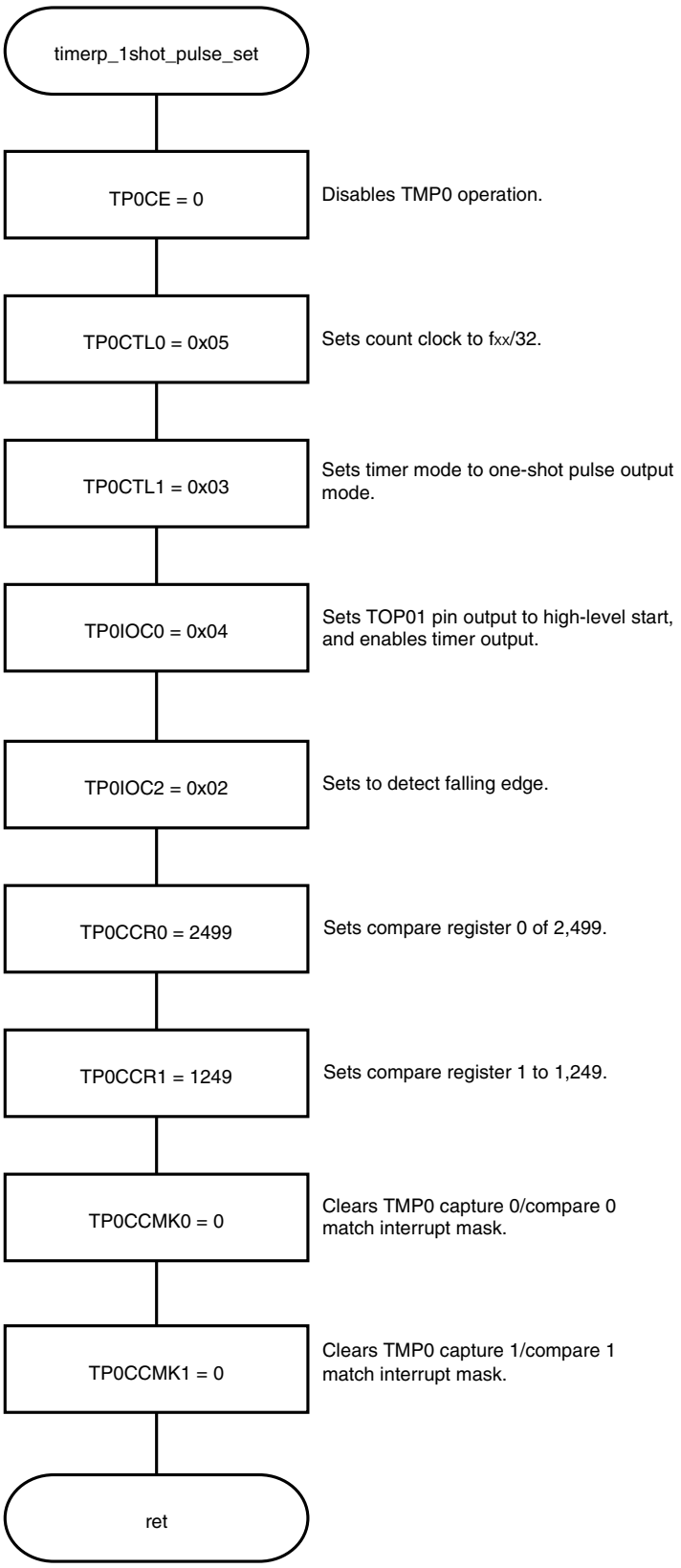
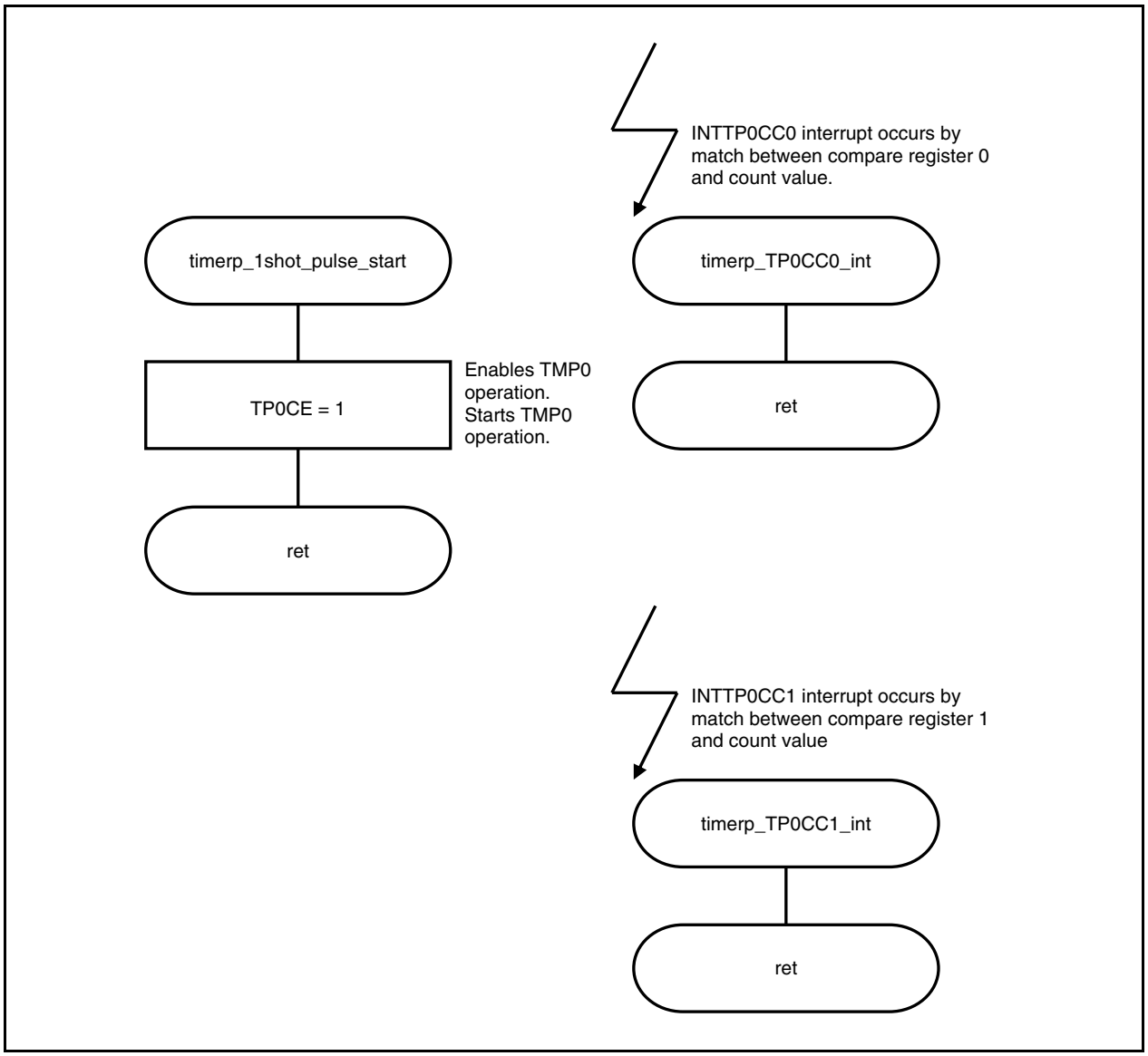


Figure 3-10. One-Shot Pulse Output Mode (3)



3.5 PWM Output Mode

[Function]	Starts operation of the 16-bit counter by setting the TP0CE bit. Outputs a PWM waveform with a 50% duty factor whose half cycle is equal to the set value of the TP0CCR0 register + 1, by clearing the 16-bit counter upon a compare match with the TP0CCR0 register and inverting the TOP00 pin output. Inverts the TOP01 pin output upon a match between the value set by the TP0CCR1 register and the count value of the 16-bit counter. Inverts the TOP01 pin output when the 16-bit counter is cleared. Can be implemented with TMP0 to TMP5.
[Function name]	timerp_pwm_output_main
[Argument]	None
[Processing content]	Performs count operation of an fxx/32 count clock, generates an interrupt by inverting the TOP00 and TOP01 pin outputs upon the count subsequent to the count whose value matches the value of the TP0CCR0 register, and clears the counter. Generates an interrupt by inverting the TOP01 pin output upon the count subsequent to the count whose value matches the value of the TP0CCR1 register. The TOP00 and TOP01 pins start output at high level.
[SFR used]	None
[call function]	timerp_port_set, timerp_pwm_output_set, timerp_pwm_output_start
[Variable]	None
[Interrupts]	timerp_TP0CC0_int timerp_TP0CC1_int
[Interrupt sources]	INTTP0CC0 INTTP0CC1
[File name]	timerp_pwm_output\timerp_pwm_output.c
[Caution]	None

[Function name]	timerp_port_set
[Argument]	None
[Processing content]	Sets port 3 to TOP00 output pin and TOP01 output pin.
[SFRs used]	PFC3L: 0x0C (Sets TOP00 output pin, TOP01 output pin) PFCE3L: 0x04 (Sets TOP00 output pin, TOP01 output pin) PMC3L: 0x0C (Sets TOP00 output pin, TOP01 output pin)
[call function]	None
[Variable]	None
[File name]	timerp_pwm_output\timerp_pwm_output.c
[Caution]	None

[Function name]	timerp_pwm_output_set
[Argument]	None
[Processing content]	Sets TMP0 control register.
[SFRs used]	TP0CTL0.TP0CE: 0 (Disables TMP0 operation.) TP0CTL0: 0x05 (Sets count clock to fxx/32.) TP0CTL1: 0x04 (Sets timer mode to one-shot pulse mode.) TP0IOC0: 0x05 (Sets TOP00 pin output to high-level start, enables timer output, Sets TOP01 pin output to high-level start, enables timer output.) TP0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TP0CCR1: 1249 (Sets 16-bit counter compare register 1 to 1,249.) TP0CCMK0: 0 (Sets TMP0 capture 0/compare 0 match interrupt mask clear.) TP0CCMK1: 0 (Sets TMP0 capture 1/compare 1 match interrupt mask clear.)
[call function]	None
[Variable]	None
[File name]	timerp_pwm_output\timerp_pwm_output.c
[Caution]	None

The active level width, cycle, and duty of the PWM waveform output from the TOP01 pin can be calculated by the following formula.

Active level width = (TP0CCR1 register set value) × count clock cycle

Cycle = (TP0CCR0 register set value + 1) × count clock cycle

Duty = (TP0CCR1 register set value) / (TP0CCR0 register set value + 1)

[Function name]	timerp_pwm_output_start
[Argument]	None
[Processing content]	Starting function of PWM output mode
[Starting method]	Call this function after calling the timerp_pwm_output_set function.
[SFR used]	TP0CTL0.TP0CE: 1 (Enables TMP0 operation.)
[call function]	None
[Variable]	None
[File name]	timerp_pwm_output\timerp_pwm_output.c
[Caution]	None

Interrupt function

[Function name]	timerp_TP0CC0_int	
[Overview]	Defined by the user.	
[Factor]	INTTP0CC0	Match between the count value of the 16-bit counter and TP0CCR0
[call function]	None	
[Variable]	None	
[File name]	timerp_pwm_output\timerp_pwm_output.c	
[Caution]	None	

[Function name]	timerp_TP0CC1_int	
[Overview]	Defined by the user.	
[Factor]	INTTP0CC1	Match between the count value of the 16-bit counter and TP0CCR1
[call function]	None	
[Variable]	None	
[File name]	timerp_pwm_output\timerp_pwm_output.c	
[Caution]	None	

Figure 3-11. PWM Output Mode (1)

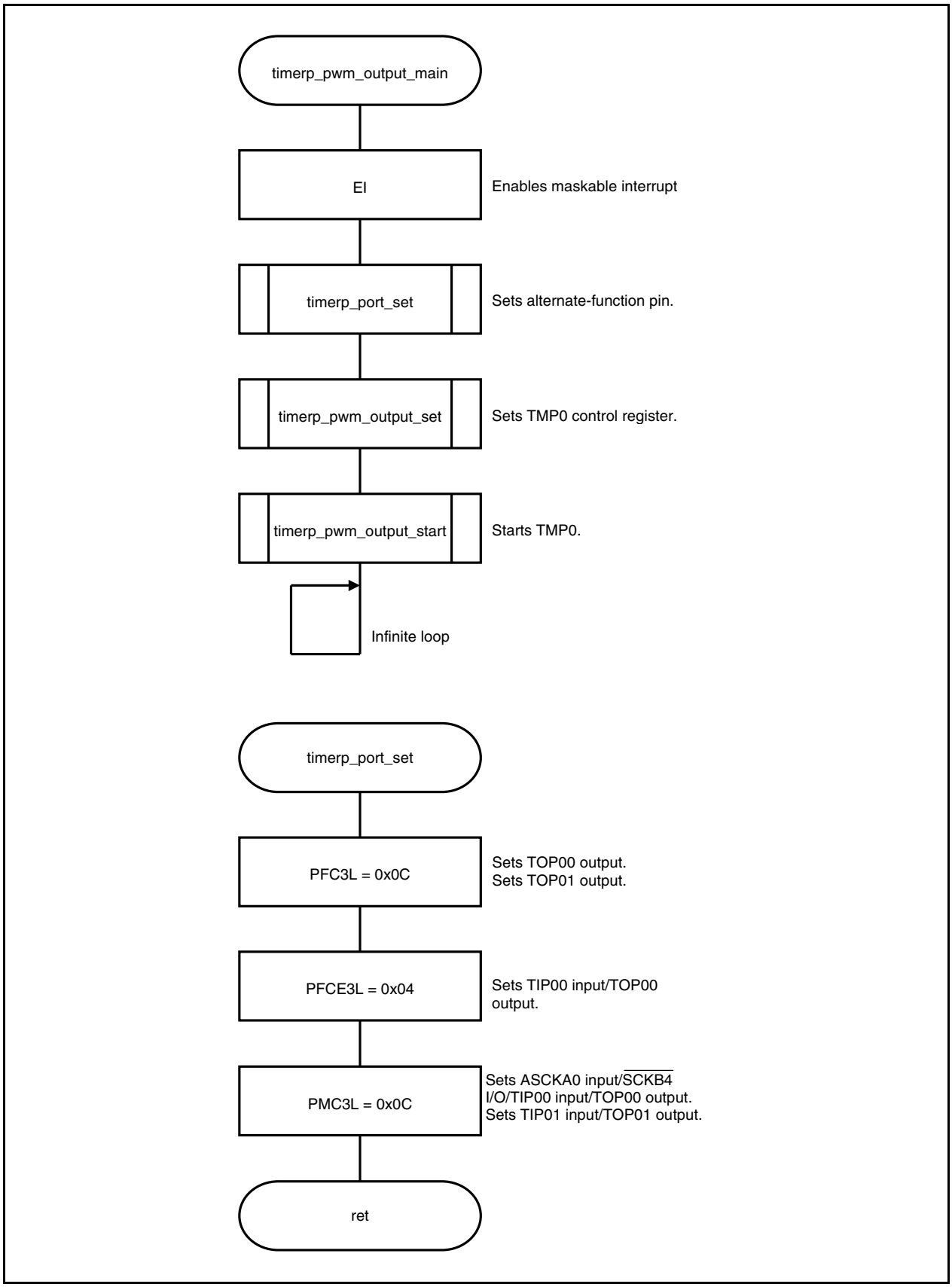


Figure 3-12. PWM Output Mode (2)

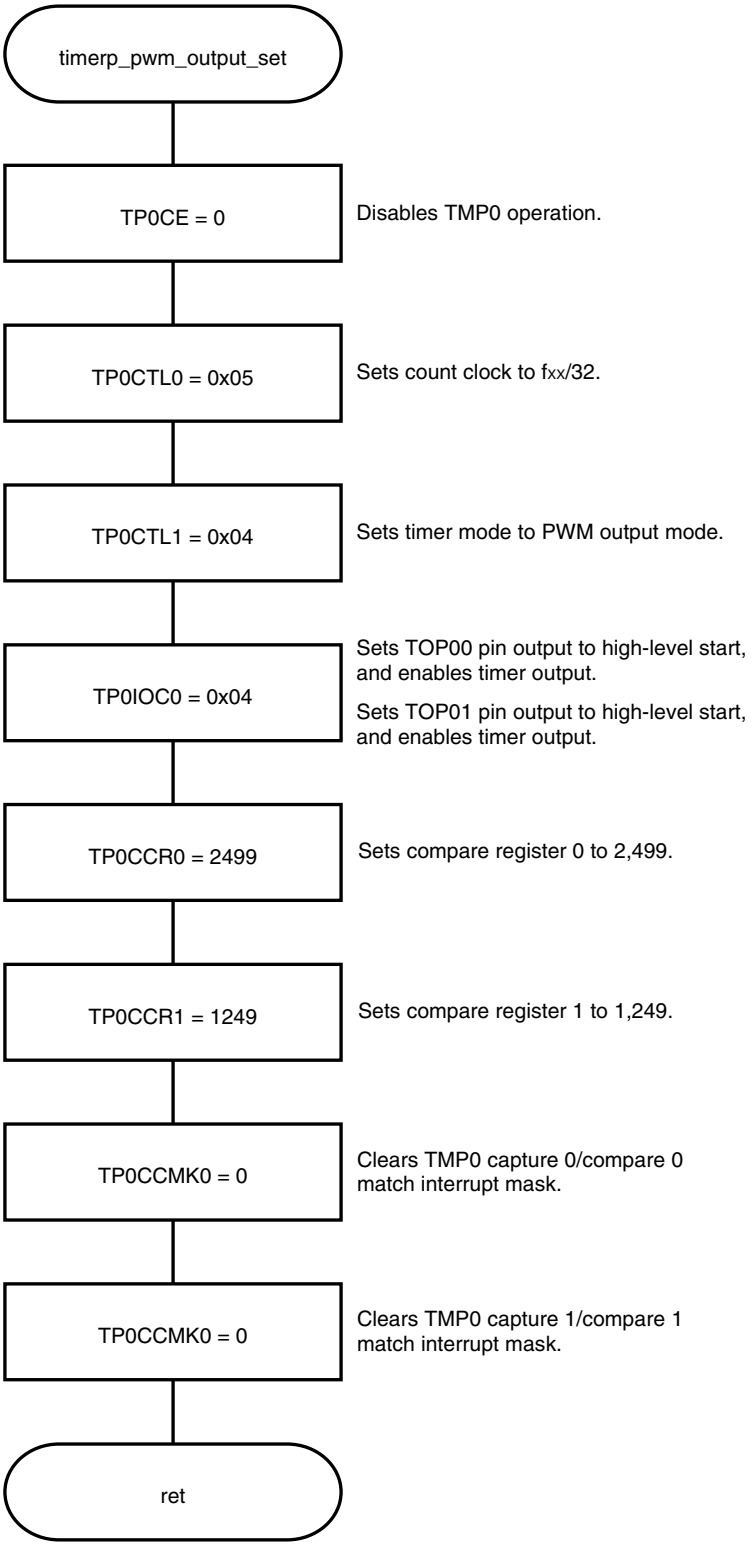
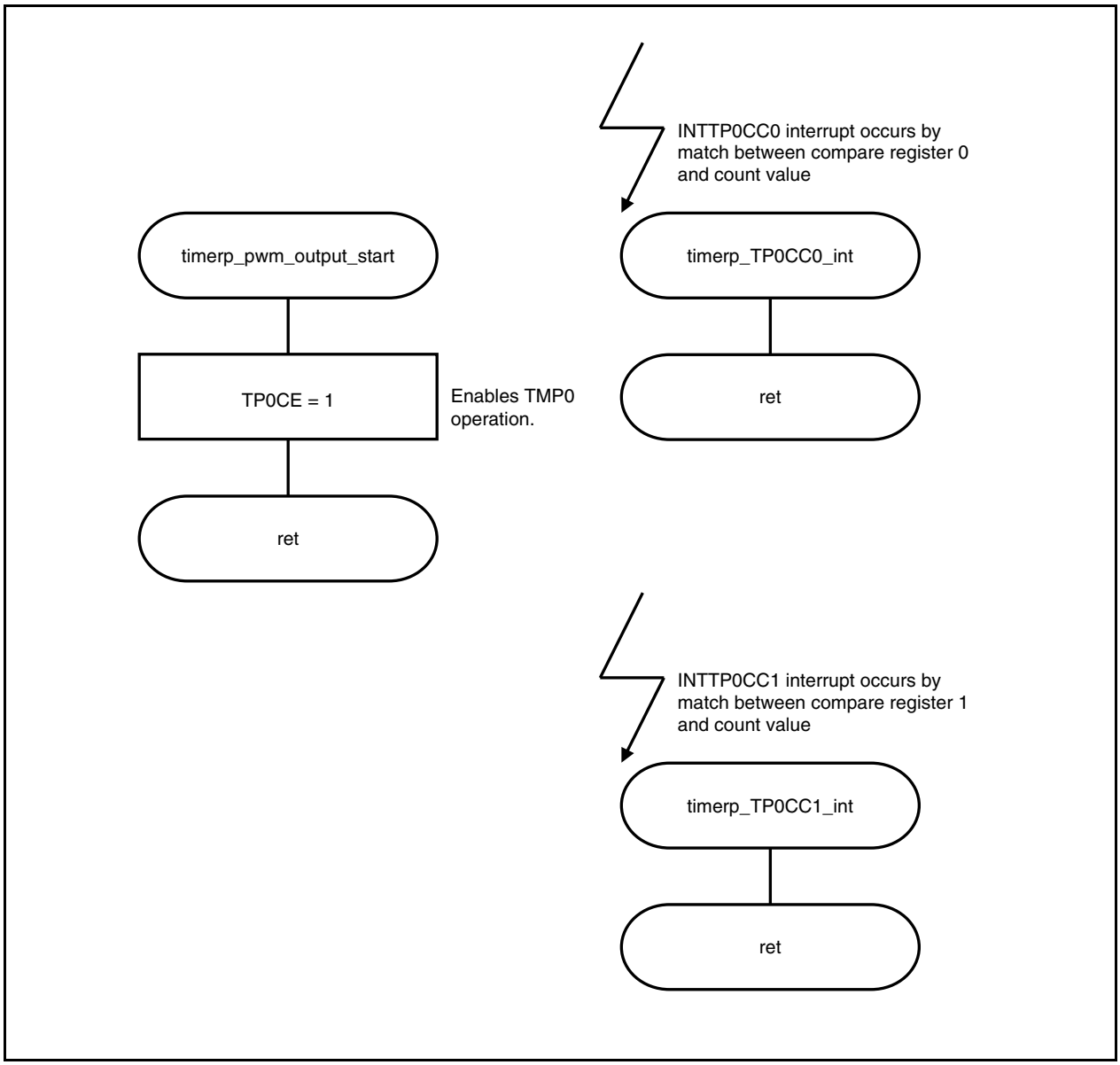


Figure 3-13. PWM Output Mode (3)



3.6 Free-Running Timer Mode

[Function]	Starts operation of the 16-bit counter by setting the TP0CE bit. Inverts the TOP00 pin output upon a compare match between the TP0CCR0 register and the count value of the 16-bit counter (compare function). Even if the TP0CCR0 register value and the count value of 16-bit counter match, the count value is not cleared. After counting up to FFFFH, generates an overflow interrupt request signal (INTTP0OV) at the next clock and clears to 0000H to continue counting. Stores the count value of the 16-bit counter when the valid edge of the capture trigger input (TIP01 pin) is detected (capture function). The capture function can be implemented with TMP0 to TMP5.
[Function name]	timerp_free_running_main
[Argument]	None
[Processing content]	Performs count operation of an fxx/32 count clock, generates an interrupt by inverting the TOP00 pin output upon the count subsequent to the count whose value matches the value of the TP0CCR0 register, and clears the counter. Generates an interrupt by capturing the counter value to the TP0CCR1 register when a valid edge is detected from the TIP01 pin. Generates an INTTP0OV interrupt when a counter overflow is detected. The TOP00 pin start output at high level.
[SFR used]	None
[call function]	timerp_port_set, timerp_free_running_set, timerp_free_running_start
[Variable]	None
[Interrupts]	timerp_TP0CC0_int timerp_TP0CC1_int timerp_TP0OV_int
[Interrupt sources]	INTTP0CC0 INTTP0CC1 INTTP0OV
[File name]	timerp_free_running\timerp_free_running.c
[Caution]	None

[Function name]	timerp_port_set
[Argument]	None
[Processing content]	Sets port 3 to TOP00 output pin, TIP01 input pin.
[SFRs used]	PFC3L: 0x04 (Sets TOP00 output pin, TIP01 input pin.) PFCE3L: 0x04 (Sets TOP00 output pin, TIP01 input pin.) PMC3L: 0x0C (Sets TIP01 input pin, TOP00 output pin.)
[call function]	None
[Variable]	None
[File name]	timerp_free_running\timerp_free_running.c
[Caution]	None

[Function name]	timerp_free_running_set
[Argument]	None
[Processing content]	Sets TMP0 control register.
[SFRs used]	TP0CTL0.TP0CE: 0 (Disables TMP0 operation.) TP0CTL0: 0x05 (Sets count clock to fxx/32.) TP0CTL1: 0x05 (Sets timer mode to free-running timer mode.) TP0IOC0: 0x01 (Sets TOP00 pin output to high-level start, enables timer output.) TP0IOC1: 0x08 (Sets TIP01 pin valid edge to falling edge.) TP0OPT0: 0x20 (Sets TP0CCR1 to capture register.) TP0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TP0CCMK0: 0 (Sets TMP0 capture 0/compare 0 match interrupt mask clear.) TP0CCMK1: 0 (Sets TMP0 capture 1/compare 1 match interrupt mask clear.) TP0OVMK: 0 (Sets to TMP0 overflow interrupt mask clear.)
[call function]	None
[Variable]	None
[File name]	timerp_free_running\timerp_free_running.c
[Caution]	None

[Function name]	timerp_ free_running _start
[Argument]	None
[Processing content]	Starting function of free-running timer mode.
[Starting method]	Call this function after calling the timerp_free_running_set function.
[SFR used]	TP0CTL0.TP0CE: 1 (Enables TMP0 operation.)
[call function]	None
[Variable]	None
[File name]	timerp_free_running\timerp_free_running.c
[Caution]	None

Interrupt function

[Function name]	timerp_TP0CC0_int
[Overview]	Defined by the user.
[Factor]	INTTP0CC0 Match between the count value of the 16-bit counter and TP0CCR0
[call function]	None
[Variable]	None
[File name]	timerp_free_running\timerp_free_running.c
[Caution]	None

[Function name]	timerp_TP0CC1_int
[Overview]	Defined by the user.
[Factor]	INTTP0CC1 TIP01 pin input valid edge detection.
[call function]	None
[Variable]	None
[File name]	timerp_free_running\timerp_free_running.c
[Caution]	None

[Function name]	timerp_TP0OV_int	
[Overview]	Defined by the user.	
[Factor]	INTTP0OV	16-bit counter overflow occurrence.
[call function]	None	
[Variable]	None	
[File name]	timerp_free_running\timerp_free_running.c	
[Caution]	None	

Figure 3-14. Free-Running Timer Mode (1)

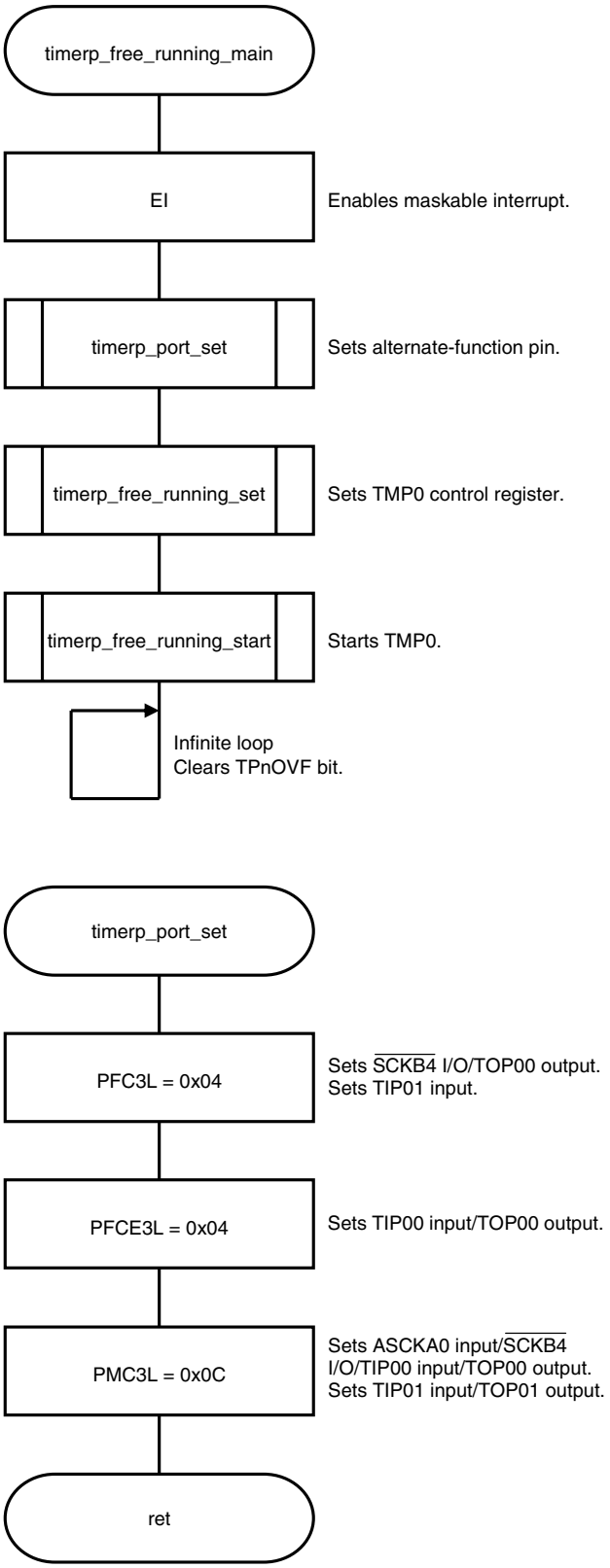


Figure 3-15. Free-Running Timer Mode (2)

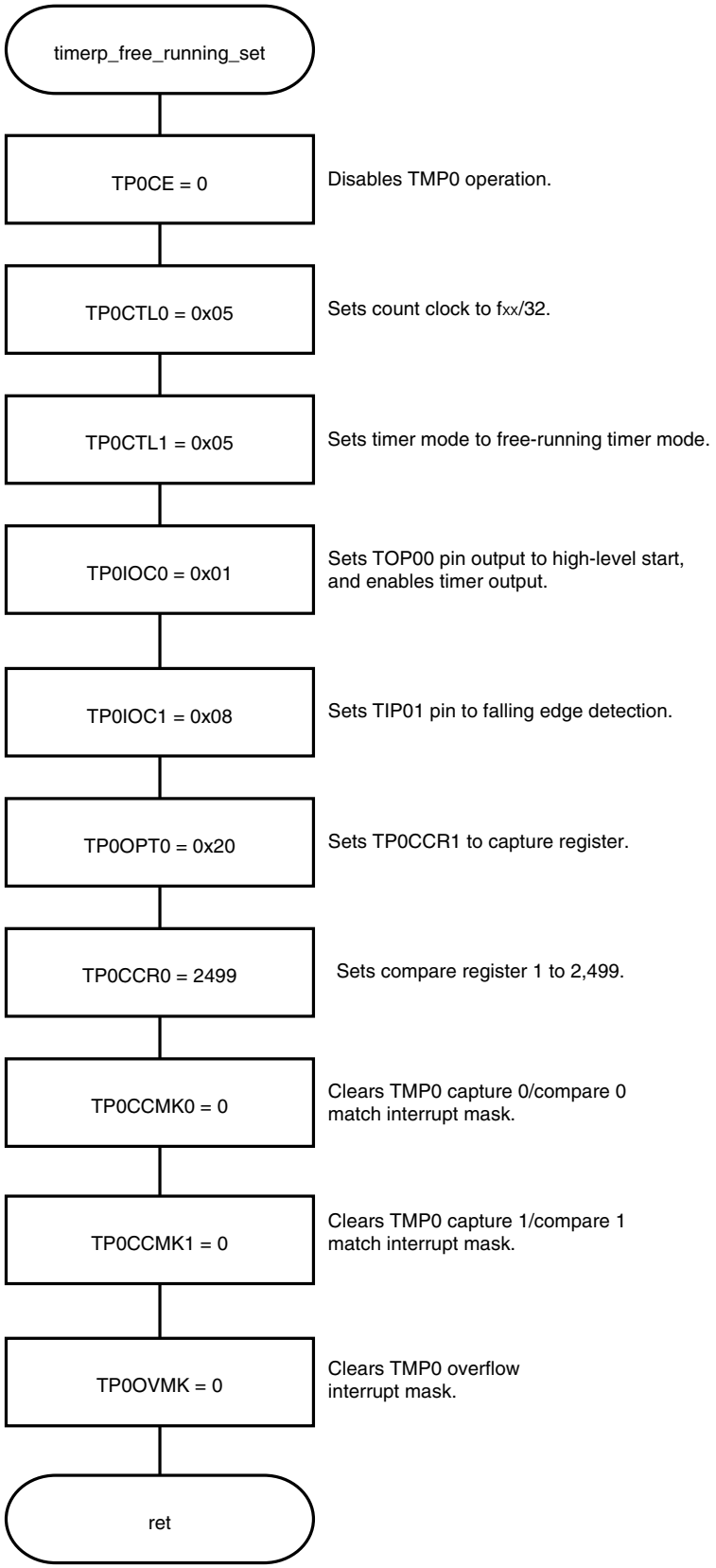
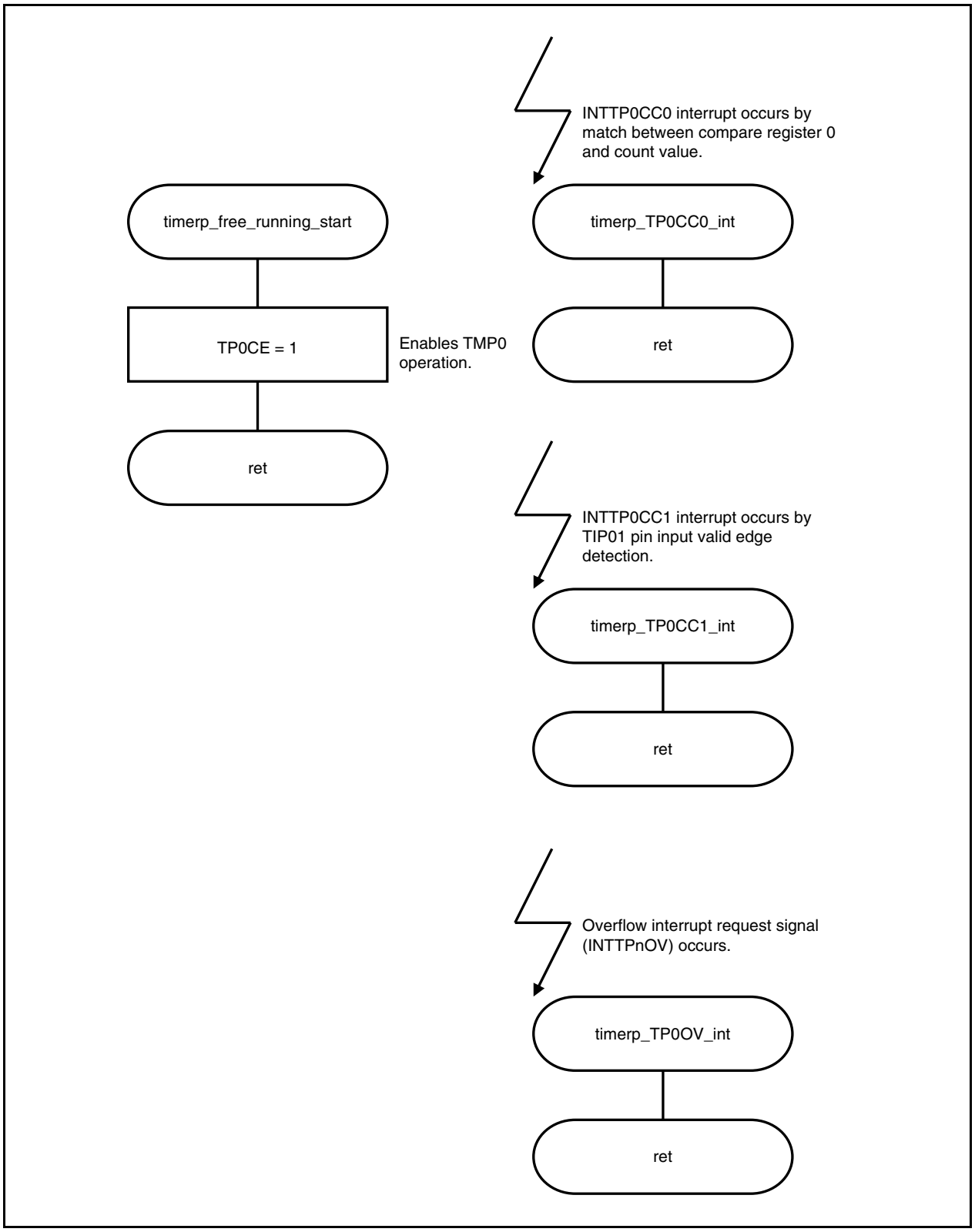


Figure 3-16. Free-Running Timer Mode (3)



3.7 Pulse Width Measurement Mode

[Function]	Starts operation of the 16-bit counter by setting the TP0CE bit. Clears the 16-bit counter by storing the count value to the TP0CCR0 register when the valid edge of the capture trigger input (TIP00 pin) is detected. Measures the valid edge interval of the TIP00 pin by generating an interrupt when the valid edge of the TIP00 pin input is detected and reading the TP0CCR0 register value. Can be implemented with TMP0 to TMP5.
[Function name]	timerp_pulse_measure
[Argument]	None
[Processing content]	Performs count operation of an fxx/32 count clock, generates an interrupt by storing the count value of the 16-bit counter to the TP0CCR0 register when the valid edge of the TIP00 pin input is detected, and clears the counter. Generates an INTTP0OV interrupt when a counter overflow is detected.
[SFR used]	None
[call function]	timerp_port_set, timerp_pulse_measure_set, timerp_ pulse_measure_start
[Variable]	None
[Interrupts]	timerp_TP0CC0_int timerp_TP0OV_int
[Interrupt sources]	INTTP0CC0 INTTP0OV
[File name]	timerp_pulse_measure\timerp_pulse_measure.c
[Caution]	None

[Function name]	timerp_port_set
[Argument]	None
[Processing content]	Sets port 3 to TIP00 input pin.
[SFRs used]	PFCE3L: 0x04 (Sets TIP00 input pin.) PMC3L: 0x04 (Sets TIP00 input pin.)
[call function]	None
[Variable]	None
[File name]	timerp_pulse_measure\timerp_pulse_measure.c
[Caution]	None

[Function name]	timerp_free_running_set
[Argument]	None
[Processing content]	Sets TMP0 control register.
[SFRs used]	TP0CTL0.TP0CE: 0 (Disables TMP0 operation.) TP0CTL0: 0x05 (Sets count clock to fxx/32.) TP0CTL1: 0x06 (Sets timer mode to pulse width measurement mode.) TP0IOC1: 0x02 (Sets TIP00 pin to falling edge.) TP0IOC2: 0x08 (Sets TIP00 pin to falling edge.) TP0CCMK0: 0 (Sets TMP0 capture 0/compare 0 match interrupt mask clear.) TP0OVMK: 0 (Sets to TMP0 overflow interrupt mask clear.)
[call function]	None
[Variable]	None
[File name]	timerp_pulse_measure\timerp_pulse_measure.c
[Caution]	None

The pulse width can be calculated by the following formula.

Pulse width = (TP0CCR0 register value + 1) × count clock cycle

The pulse width when the 16-bit counter overflow is detected can be calculated by the following formula.

Pulse width = (TP0CCR0 register value + 0x10001) × count clock cycle

[Function name]	timerp_pulse_measure_start
[Argument]	None
[Processing content]	Starting function of pulse width measurement mode
[Starting method]	Call this function after calling the timerp_pulse_measure_set function.
[SFR used]	TP0CTL0.TP0CE: 1 (Enables TMP0 operation.)
[call function]	None
[Variable]	None
[File name]	timerp_pulse_measure\timerp_pulse_measure.c
[Caution]	None

Interrupt function

[Function name]	timerp_TP0CC0_int	
[Overview]	Defined by the user.	
[Factor]	INTTP0CC0	TIP00 pin input valid edge detection
[call function]	None	
[Variable]	None	
[File name]	timerp_pulse_measure\timerp_pulse_measure.c	
[Caution]	None	

[Function name]	timerp_TP0OV_int	
[Overview]	Defined by the user.	
[Factor]	INTTP0OV	16-bit counter overflow occurrence.
[call function]	None	
[Variable]	None	
[File name]	timerp_pulse_measure\timerp_pulse_measure.c	
[Caution]	None	

Figure 3-17. Pulse Width Measurement Mode (1)

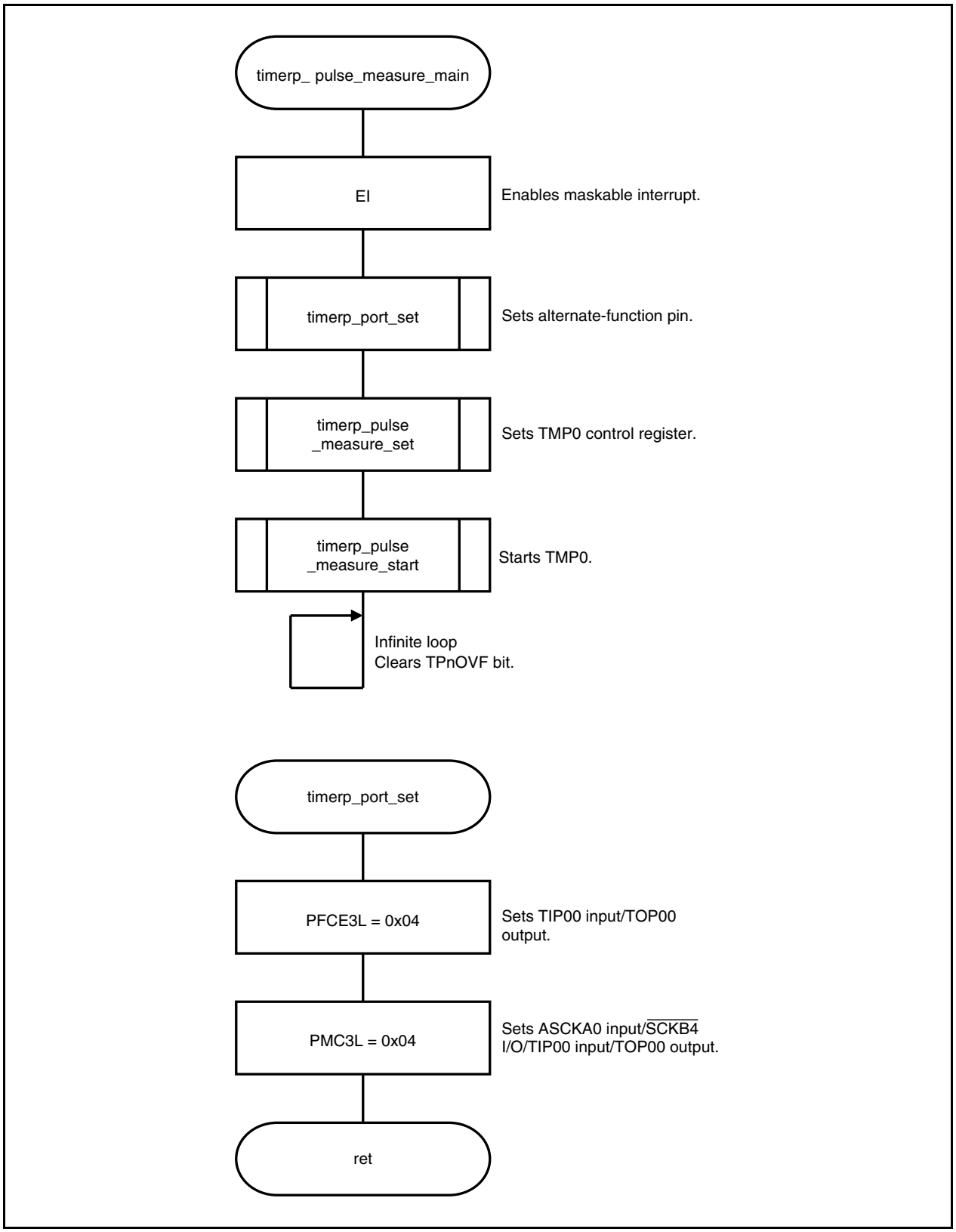


Figure 3-18. Pulse Width Measurement Mode (2)

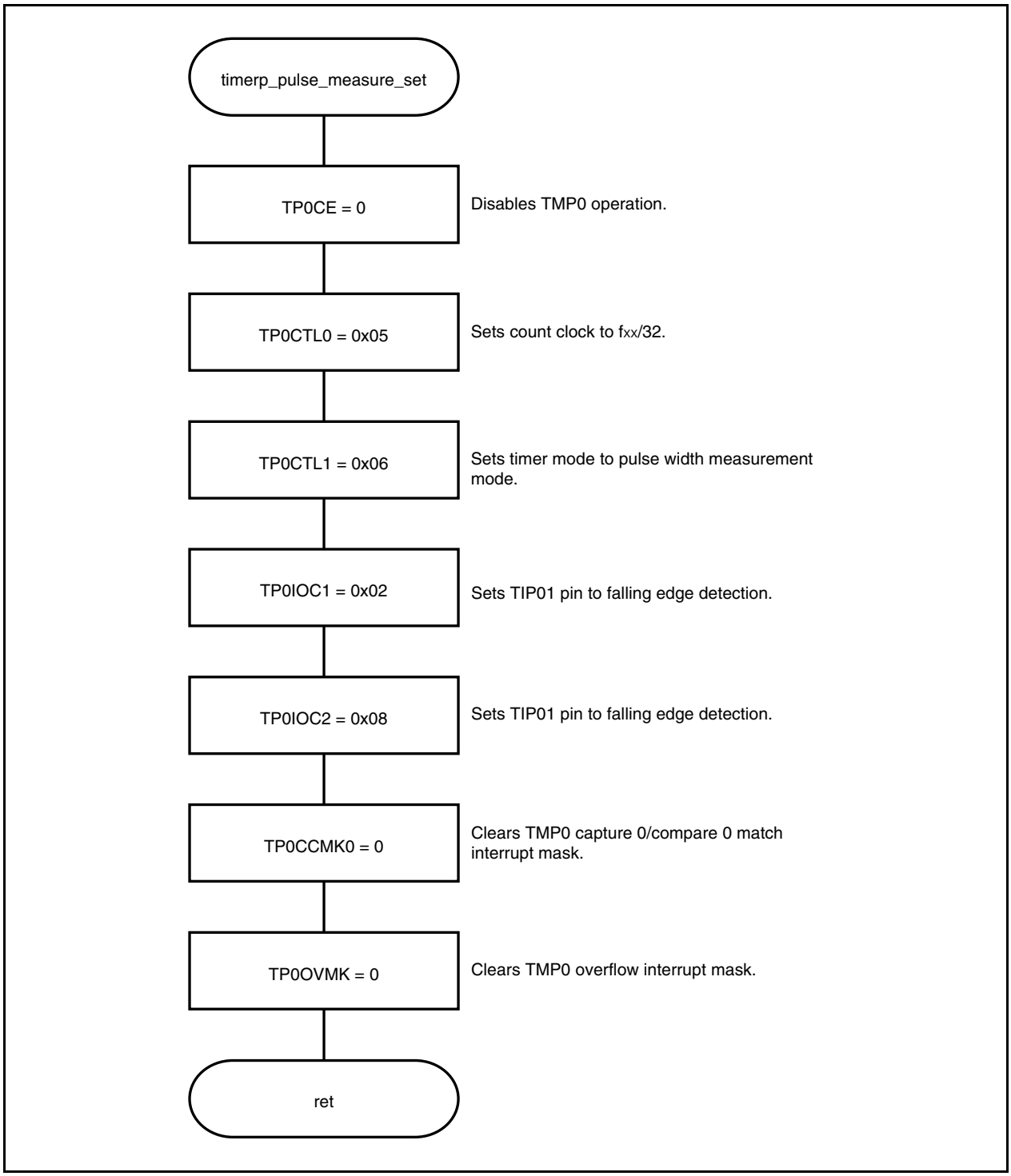
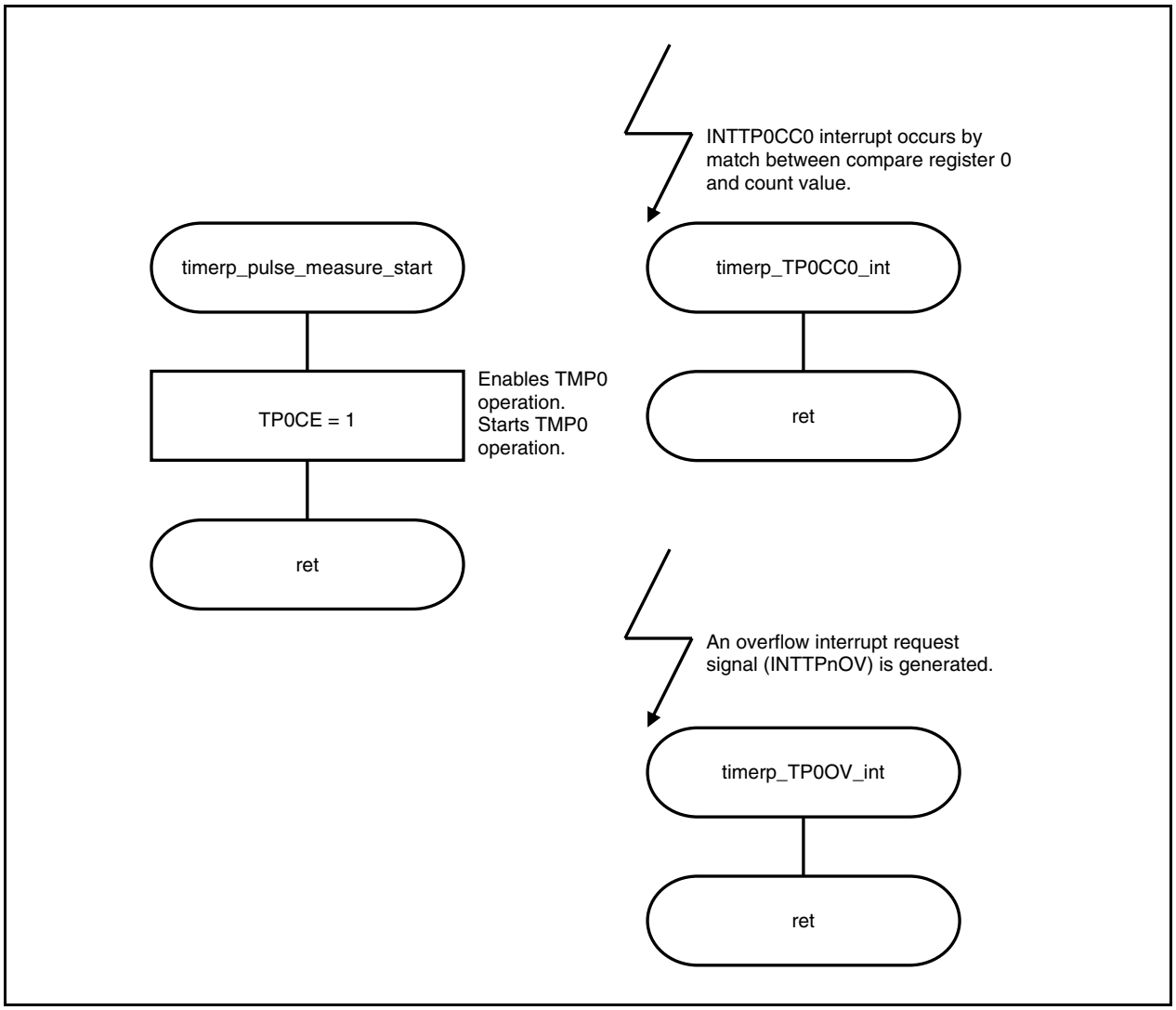


Figure 3-19. Pulse Width Measurement Mode (3)



4.1 Interval Timer Mode

[Function]	Starts 16-bit counter operation by setting the TQ0CE bit. Outputs an interrupt request signal (INTTQ0CC0) at an interval set by the TQ0CCR0 register. Inverts the TOQ00 pin output when the value set by the TQ0CCR0 register and the count value of the 16-bit counter match.
[Function name]	timerq_interval_main
[Argument]	None
[Processing content]	Performs count operation of an fxx/32 count clock, generates an interrupt by inverting the TOQ00 pin output upon the count subsequent to the count whose value matches the value of the TQ0CCR0 register. The TOQ00 pin starts output at high level.
[SFR used]	None
[call function]	timerq_port_set, timerq_interval_set, timerq_interval_start
[Variable]	None
[Interrupt]	timerq_TQ0CC0_int
[Interrupt source]	INTTQ0CC0
[File name]	timerq_interval\timerq_interval.c
[Caution]	None

[Function name]	timerq_port_set
[Argument]	None
[Processing content]	Sets port 5 to TOQ00 output pin.
[SFRs used]	PFC5: 0x00 (Sets to TOQ00 output pin.) PFCE5: 0x08 (Sets to TOQ00 output pin.) PMC5: 0x08 (Sets to TOQ00 output pin.)
[call function]	None
[Variable]	None
[File name]	timerq_interval\timerq_interval.c
[Caution]	None

[Function name]	timerq_interval_set
[Argument]	None
[Processing content]	Sets TMQ0 control register.
[SFRs used]	TQ0CTL0.TQ0CE: 0 (Disables TMQ0 operation.) TQ0CTL0: 0x05 (Sets count clock to fxx/32.) TQ0CTL1: 0x00 (Sets timer mode to interval timer mode.) TQ0IOC0: 0x01 (Sets TOQ00 pin output level to high-level start, enables timer output.) TQ0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499) TQ0CCMK0: 0 (Clears TMQ0 capture 0/compare 0 match interrupt mask.)
[call function]	None
[Variable]	None
[File name]	timerq_interval\timerq_interval.c
[Caution]	None

The interval time can be calculated by the following formula.

Interval = (Set value of TQ0CCR0 register + 1) × Count clock cycle

[Function name]	timerq_interval_start
[Argument]	None
[Processing content]	Starting function of interval timer mode
[Starting method]	Call this function after calling the timerq_interval_set function.
[SFR used]	TQ0CTL0.TQ0CE: 1 (Enables TMQ0 operation)
[call function]	None
[Variable]	None
[File name]	timerq_interval\timerq_interval.c
[Caution]	None

Interrupt function

[Function name]	timerq_TQ0CC0_int
[Overview]	Defined by the user.
[Factor]	INTTQ0CC0 Match between the count value of the 16-bit counter and TQ0CCR0
[call function]	None
[Variable]	None
[File name]	timerq_interval\timerq_interval.c
[Caution]	None

Figure 4-1. Interval Timer Mode (1)

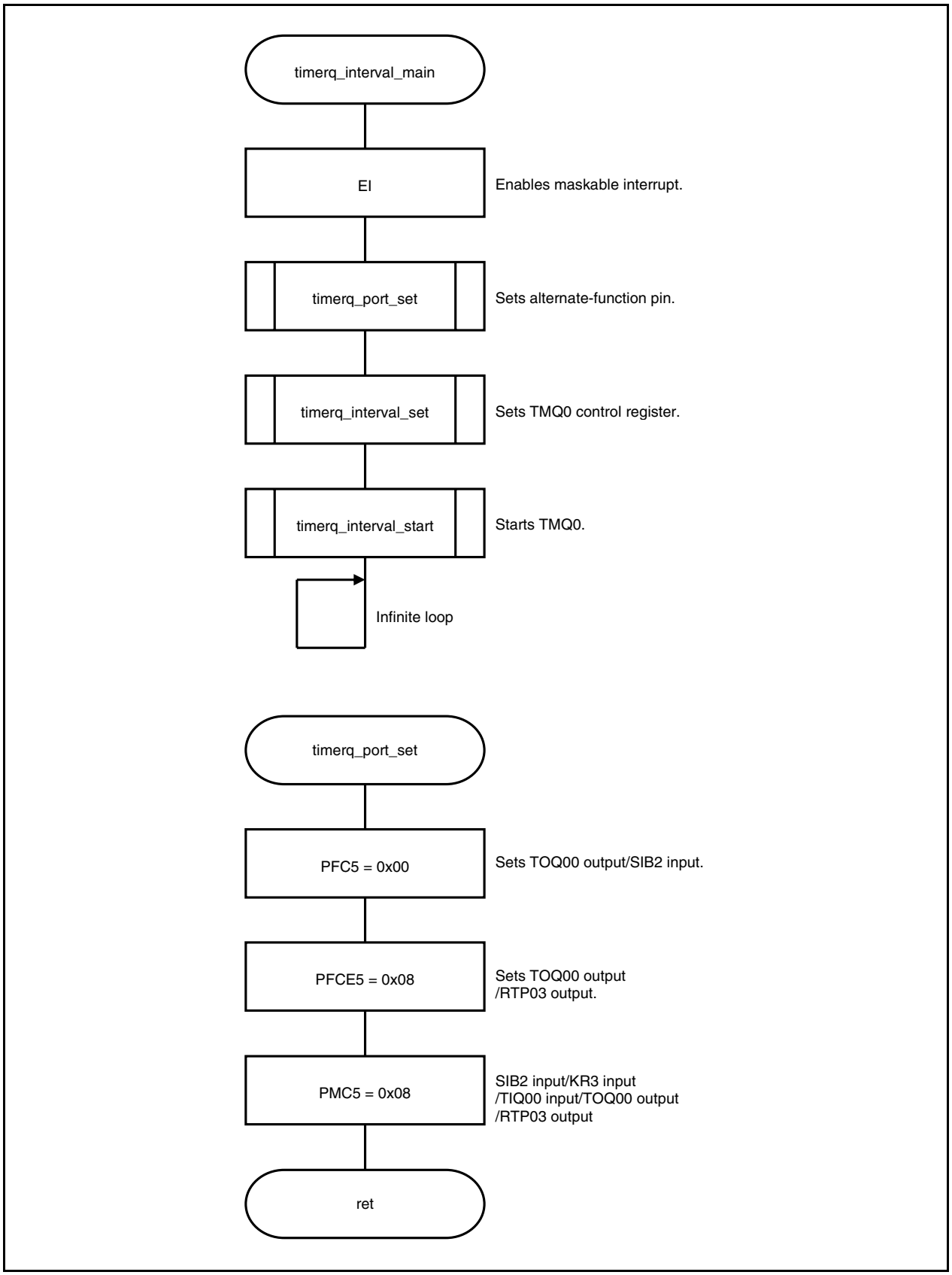
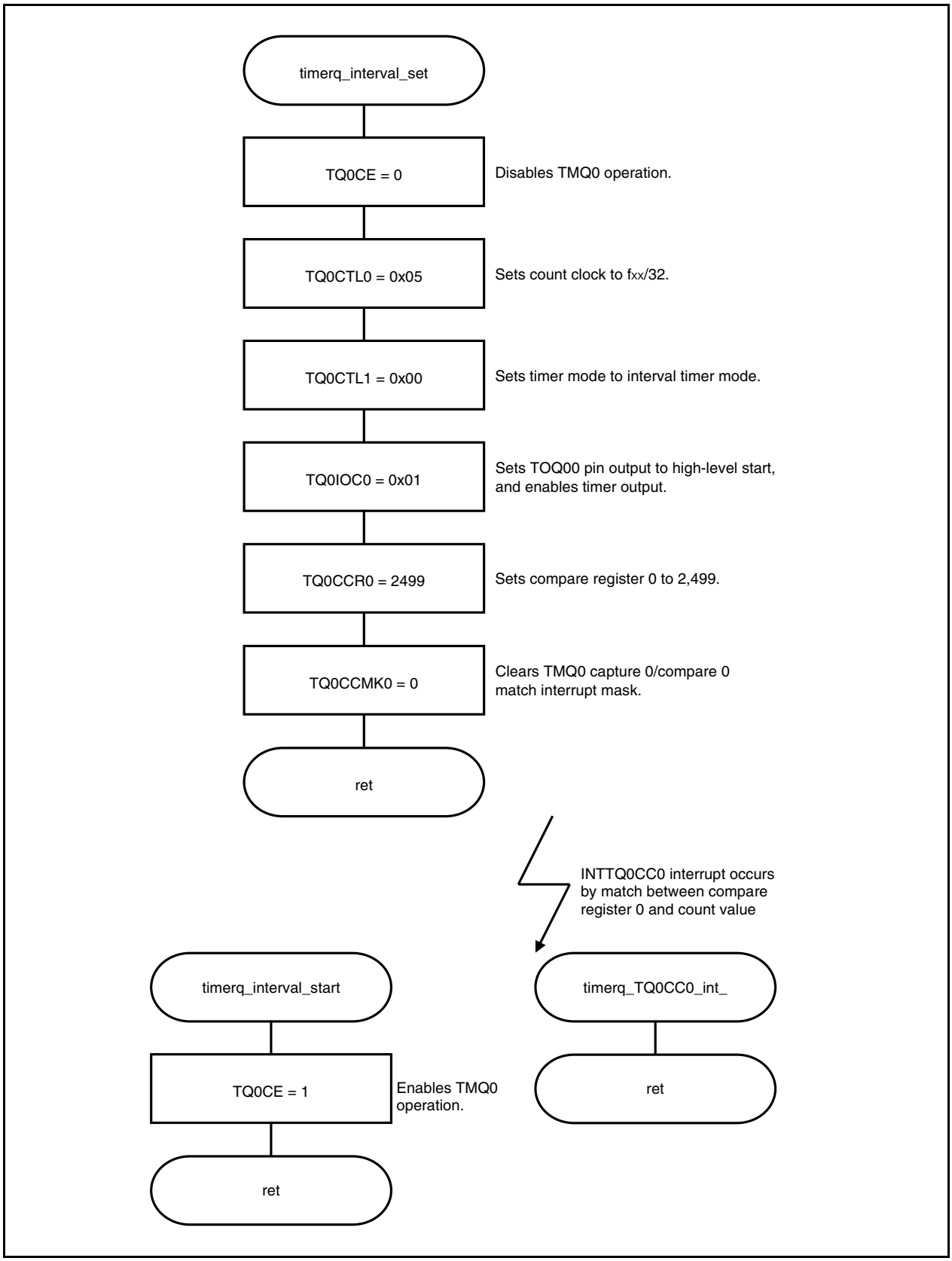


Figure 4-2. Interval Timer Mode (2)



4.2 External Event Count Mode

[Function]	Counts the valid edge of the external event count input (TIQ00 pin) and generates an interrupt request signal (INTTQ0CC0) upon a match between the count value set by the TQ0CCR0 register and the count value. (Clears the 16-bit counter simultaneously.)
[Function name]	timerq_event_count_main
[Argument]	None
[Processing content]	Counts the valid edge of the external event count input, generates an interrupt upon the count (value set to TQ0CCR0 register + 1) subsequent to the count whose value matches the value of the TQ0CCR0 register, and clears the counter.
[SFR used]	None
[call function]	timerq_port_set, timerq_event_count_set, timerq_event_count_start
[Variable]	None
[Interrupt]	timerq_TQ0CC0_int
[Interrupt source]	INTTQ0CC0
[File name]	timerq_event_count\timerq_event_count.c
[Caution]	None

[Function name]	timerq_port_set
[Argument]	None
[Processing content]	Sets port 5 to TIQ00 input pin.
[SFRs used]	PFC5: 0x08 (Sets TIQ00 input pin.) PMC5: 0x08 (Sets TIQ00 input pin.)
[call function]	None
[Variable]	None
[File name]	timerq_interval\timerq_event_count.c
[Caution]	None

[Function name]	timerq_event_count_set
[Argument]	None
[Processing content]	Sets TMQ0 control register.
[SFRs used]	TQ0CTL0.TQ0CE: 0 (Enables TMQ0 operation.) TQ0CTL0: 0x00 (Sets count clock to f_{XX}.) TQ0CTL1: 0x01 (Sets timer mode to external event count mode.) TQ0IOC2: 0x08 (Sets TIQ00 pin valid edge to falling edge.) TQ0CCR0: 40 (Sets 16-bit counter compare register 0 to 40.) TQ0CCMK0: 0 (Clears TMQ0 capture 0/compare 0 match interrupt mask.)
[call function]	None
[Variable]	None
[File name]	timerq_event_count\timerq_event_count.c
[Caution]	None

[Function name]	timerq_event_count_start
[Argument]	None
[Processing content]	Starting function of external event count mode.
[Starting method]	Call this function after calling the timerq_event_count_set function.
[SFR used]	TQ0CTL0.TQ0CE: 1 (Enables TMQ0 operation.)
[call function]	None
[Variable]	None
[File name]	timerq_event_count\timerq_event_count.c
[Caution]	None

Interrupt function

[Function name]	timerq_TQ0CC0_int
[Overview]	Defined by the user.
[Factor]	INTTQ0CC0 Match between the count value of the 16-bit counter and TQ0CCR0.
[call function]	None
[Variable]	None
[File name]	timerq_event_count\timerq_event_count.c
[Caution]	None

Figure 4-3. External Event Count Mode (1)

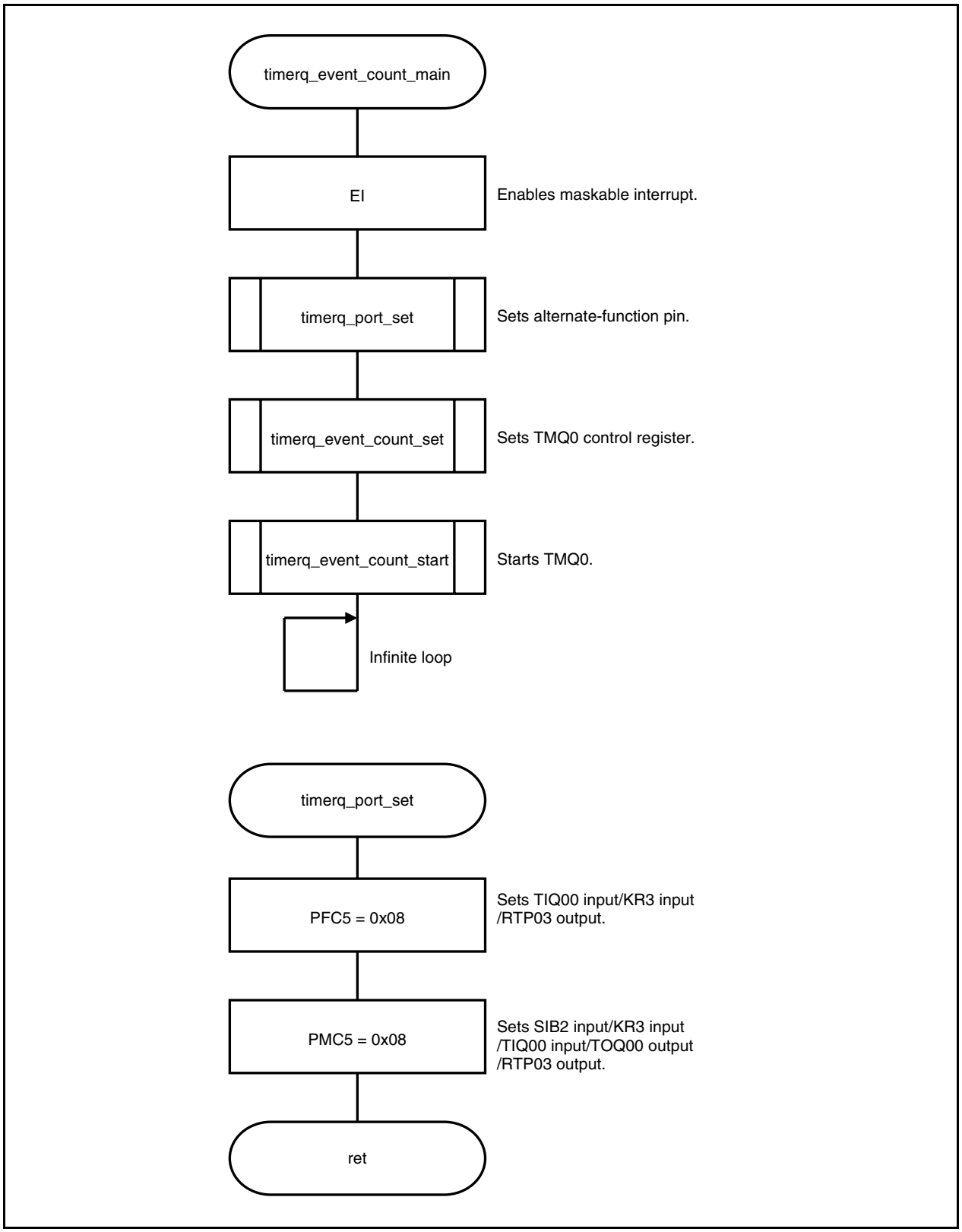
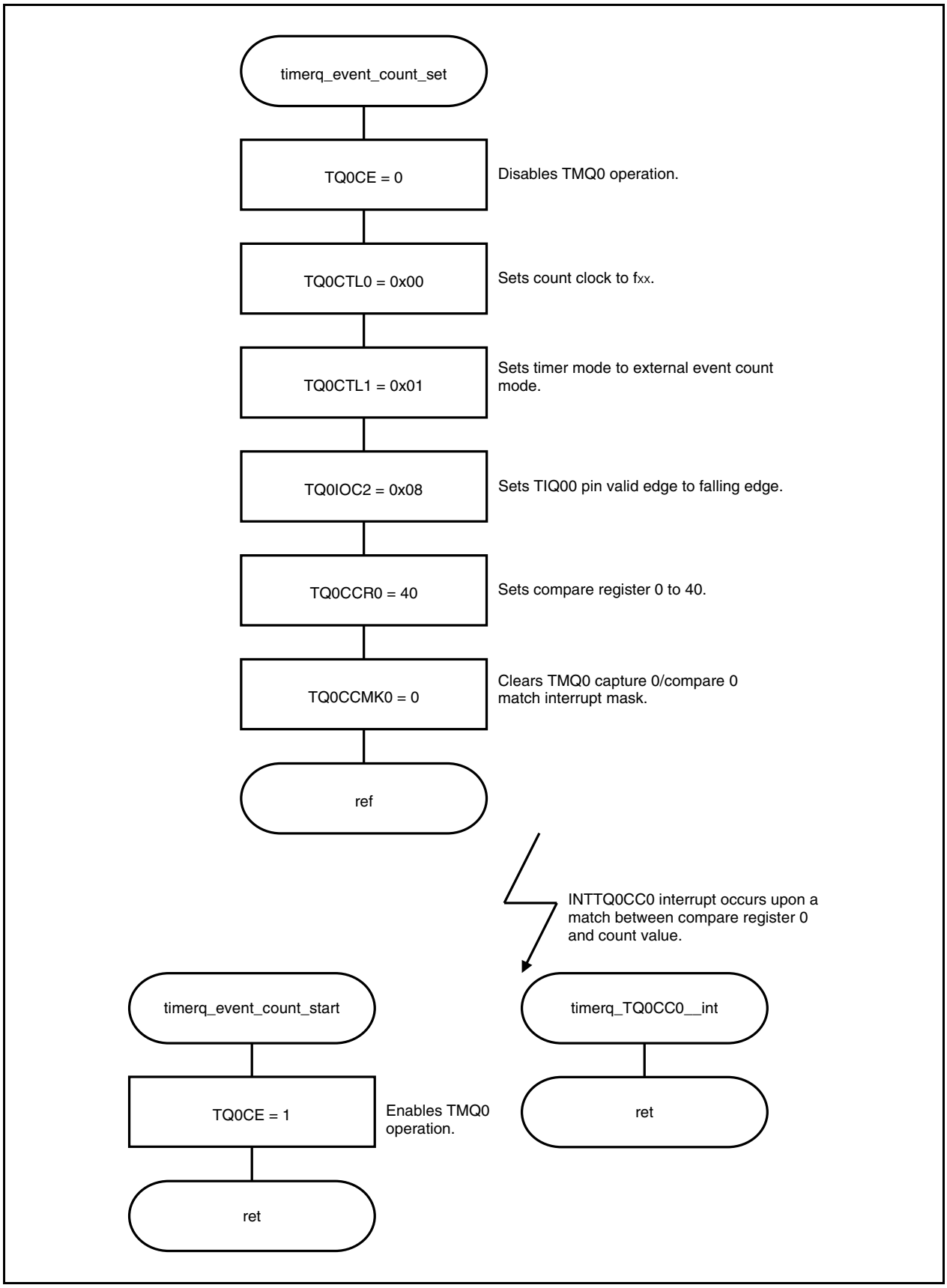


Figure 4-4. External Event Count Mode (2)



4.3 External Trigger Pulse Output Mode

[Function]	Starts operation of the 16-bit counter when the valid edge of the external trigger input (TIQ00) is detected. Clears the 16-bit counter upon a compare match with the TQ0CCR0 register. Outputs the PWM waveform from the TOQ01 pin upon a match between the value set by the TQ0CCR1 register and the count value of the 16-bit counter. When the trigger is generated again, clears the counter to 0000H and restarts.
[Function name]	timerq_trigger_pulse_main
[Argument]	None
[Processing content]	Starts count operation of an fxx/32 count clock when the valid edge of the external trigger input is detected, generates an interrupt upon the count subsequent to the count whose value matches the value of the TQ0CCR0 register, and clears the counter. Generates an interrupt by inverting the TOQ01 pin output upon the count subsequent to the count whose value matches the value of the TQ0CCR1 register. TOQ01 pin starts output at high level.
[SFR used]	None
[call function]	timerq_port_set, timerq_trigger_pulse_set, timerq_trigger_pulse_start
[Variable]	None
[Interrupts]	timerq_TQ0CC0_int timerq_TQ0CC1_int
[Interrupt sources]	INTTQ0CC0 INTTQ0CC1
[File name]	timerq_trigger_pulse\timerq_trigger_pulse.c
[Caution]	None

[Function name]	timerq_port_set
[Argument]	None
[Processing content]	Sets port 5 to TOQ01 output pin and TIQ00 input pin.
[SFRs used]	PFC5: 0x08 (Sets TOQ01 output pin, TIQ00 input pin.) PFCE5: 0x01 (Sets TOQ01 output pin, TIQ00 input pin.) PMC5: 0x09 (Sets TIQ00 input pin, TOQ01 output pin.)
[call function]	None
[Variable]	None
[File name]	timerq_trigger_pulse\timerq_ trigger_pulse.c
[Caution]	None

[Function name]	timerq_trigger_pulse_set
[Argument]	None
[Processing content]	Sets TMQ0 control register.
[SFRs used]	TQ0CTL0.TQ0CE: 0 (Disables TMQ0 operation.) TQ0CTL0: 0x05 (Sets count clock to fxx/32.) TQ0CTL1: 0x02 (Sets timer mode to external trigger pulse output mode.) TQ0IOC0: 0x04 (Sets TOQ01 pin output to high-level start, and enables timer output.) TQ0IOC2: 0x08 (Sets TIQ00 pin valid edge to falling edge.) TQ0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TQ0CCR1: 1249 (Sets 16-bit counter compare register 1 to 1,249.) TQ0CCMK0: 0 (Clears TMQ0 capture 0/compare 0 match interrupt mask.) TQ0CCMK1: 0 (Clears TMQ0 capture 1/compare 1 match interrupt mask.)
[call function]	None
[Variable]	None
[File name]	timerq_trigger_pulse\timerq_trigger_pulse.c
[Caution]	None

The active level width, cycle and duty factor of the PWM waveform can be calculated by the following formula.

Active level width = (Set value of TQ0CCR1 register) × Count clock cycle

Cycle = (Set value of TQ0CCR0 register + 1) × Count clock cycle

Duty factor = (Set value of TQ0CCR1 register) / (Set value of TQ0CCR0 register + 1)

[Function name]	timerq_trigger_pulse_start
[Argument]	None
[Processing content]	Starting function of external trigger pulse output mode.
[Starting method]	Call this function after calling the timerq_trigger_pulse_set function.
[SFR used]	TQ0CTL0.TQ0CE: 1 (Enables TMQ0 operation.)
[call function]	None
[Variable]	None
[File name]	timerq_trigger_pulse\timerq_trigger_pulse.c
[Caution]	None

Interrupt function

[Function name]	timerq_TQ0CC0_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0CC0	Match between the count value of the 16-bit counter and TQ0CCR0.
[call function]	None	
[Variable]	None	
[File name]	timerq_trigger_pulse\timerq_trigger_pulse.c	
[Caution]	None	

[Function name]	timerq_TQ0CC1_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0CC1	Match between the count value of the 16-bit counter and TQ0CCR1.
[call function]	None	
[Variable]	None	
[File name]	timerq_trigger_pulse\timerq_trigger_pulse.c	
[Caution]	None	

Figure 4-5. External Trigger Pulse Output Mode (1)

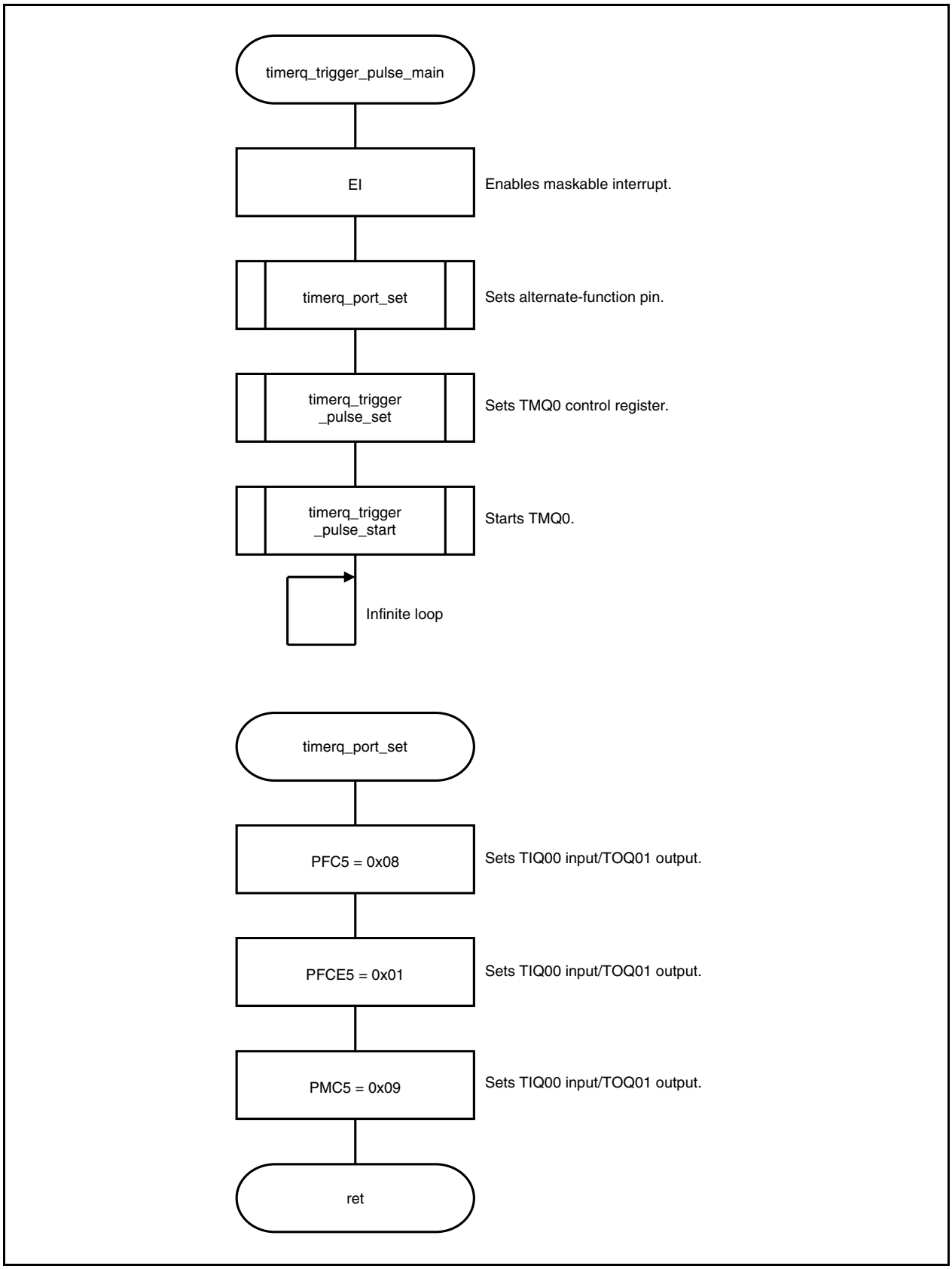


Figure 4-6. External Trigger Pulse Output Mode (2)

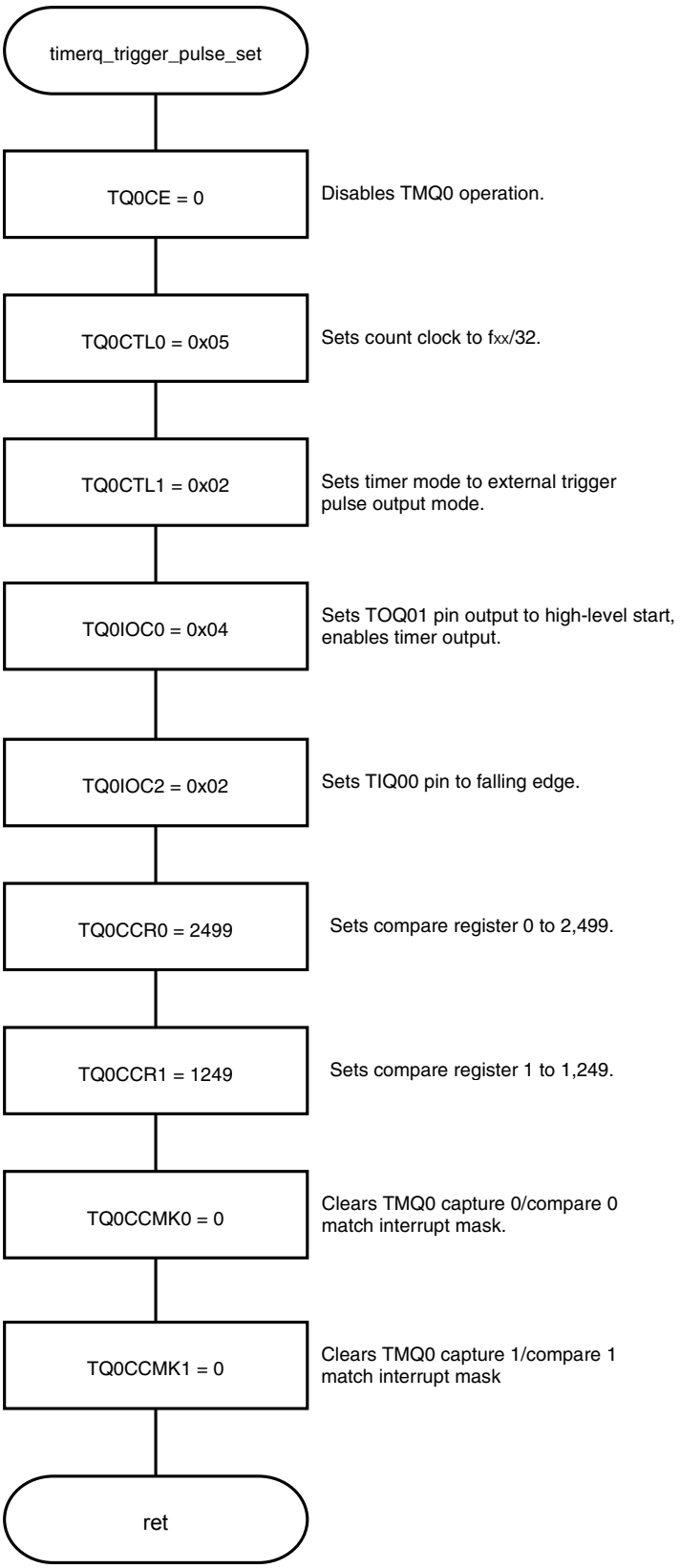
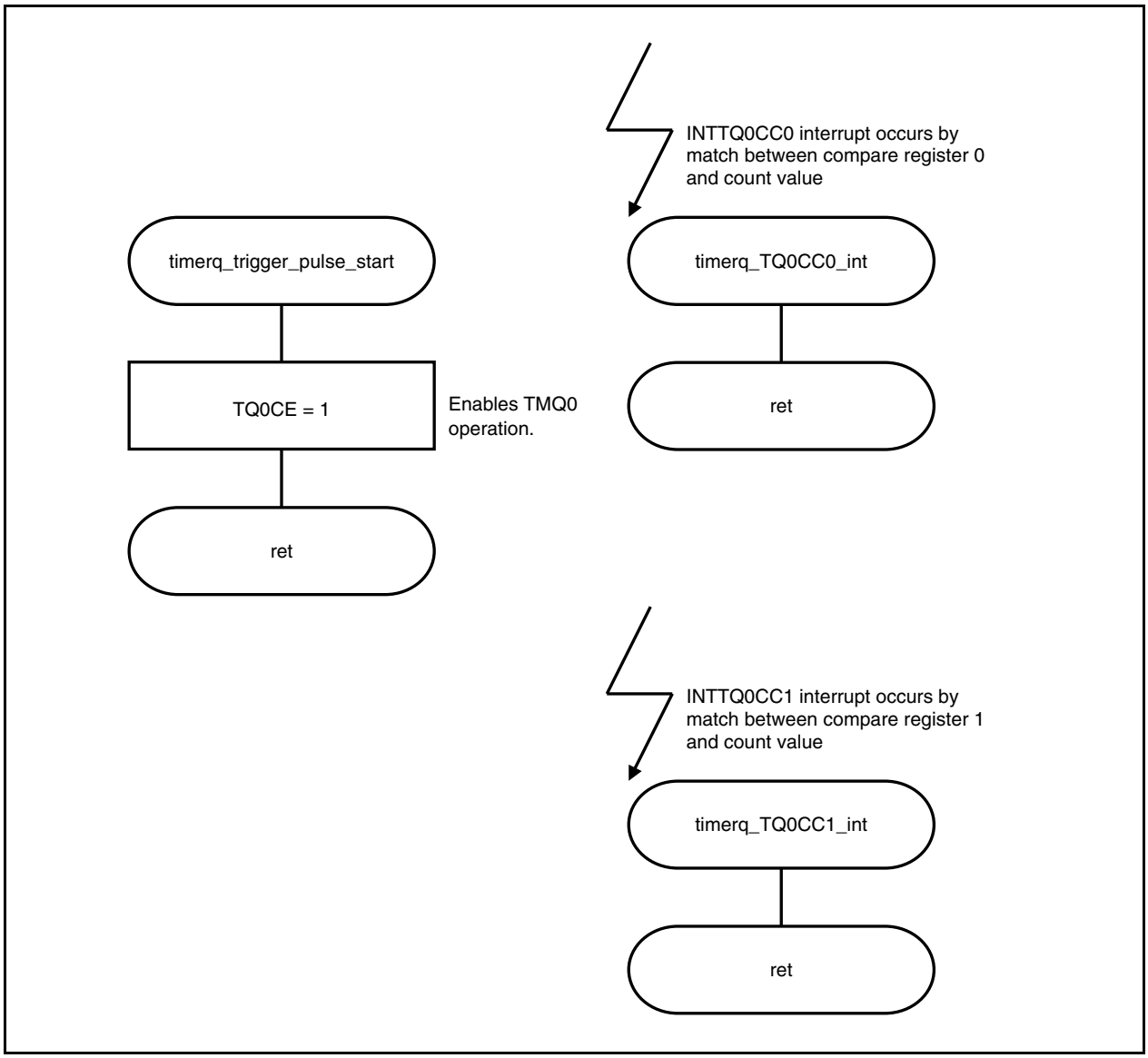


Figure 4-7. External Trigger Pulse Output Mode (3)



4.4 One-Shot Pulse Output Mode

[Function]	Starts operation of the 16-bit counter when the valid edge of the external trigger input (TIQ00 pin) is detected. Activates the TOQ01 pin output upon a match between the value set by the TQ0CCR1 register and the count value of the 16-bit counter. Inverts the TOQ01 pin output upon a match between the value set by the TQ0CCR1 register and the count value of the 16-bit counter, and outputs a one-shot pulse. Stops the count after the one-shot pulse is output, and waits for a trigger. Ignores a trigger if it is generated again during one-shot pulse output.
[Function name]	timerq_1shot_pulse_main
[Argument]	None
[Processing content]	Starts count operation of an fxx/32 count clock upon valid edge detection of the external trigger input, generates an interrupt by inverting the TOQ01 pin output upon the count subsequent to the count whose value matches the value of the TQ0CCR0 register, clears the counter, and stops count operation. Generates an interrupt by inverting the TOQ01 pin output upon the count subsequent to the count whose value matches the value of the TQ0CCR1 register. TOQ01 pin starts output at high level.
[SFR used]	None
[call function]	timerq_port_set, timerq_1shot_pulse_set, timerq_1shot_pulse_start
[Variable]	None
[Interrupts]	timerq_TQ0CC0_int timerq_TQ0CC1_int
[Interrupt sources]	INTTQ0CC0 INTTQ0CC1
[File name]	timerq_1shot_pulse\timerq_1shot_pulse.c
[Caution]	None

[Function name]	timerq_port_set
[Argument]	None
[Processing content]	Sets port 5 to TOQ01 output pin and TIQ00 input pin.
[SFRs used]	PFC5: 0x08 (Sets TOQ01 output pin, TIQ00 input pin.) PFCE5: 0x01 (Sets TOQ01 output pin, TIQ00 input pin.) PMC5: 0x09 (Sets TIQ00 input pin, TOQ01 output pin.)
[call function]	None
[Variable]	None
[File name]	timerq_1shot_pulse\timerq_1shot_pulse.c
[Caution]	None

[Function name]	timerq_1shot_pulse_set
[Argument]	None
[Processing content]	Sets TMQ0 control register.
[SFRs used]	TQ0CTL0.TQ0CE: 0 (Disables TMQ0 operation.) TQ0CTL0: 0x05 (Sets count clock to fxx/32.) TQ0CTL1: 0x03 (Sets timer mode to one-shot pulse mode.) TQ0IOC0: 0x04 (Sets TOQ01 pin output to high-level start, enables timer output.) TQ0IOC2: 0x08 (Sets TIQ00 pin valid edge to falling edge.) TQ0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TQ0CCR1: 1249 (Sets 16-bit counter compare register 1 to 1,249.) TQ0CCMK0: 0 (Sets TMQ0 capture 0/compare 0 match interrupt mask clear.) TQ0CCMK1: 0 (Sets TMQ0 capture 1/compare 1 match interrupt mask clear.)
[call function]	None
[Variable]	None
[File name]	timerq_1shot_pulse\timerq_1shot_pulse.c
[Caution]	None

The one-shot pulse output delay period and active level width can be calculated by the following formula.

Output delay period = (TQ0CCR1 register set value) × count clock cycle

Active level width = (TQ0CCR0 register set value – TQ0CCR1 register set value + 1) × count clock cycle

[Function name]	timerq_1shot_pulse_start
[Argument]	None
[Processing content]	Starting function of one-shot pulse output mode
[Starting method]	Call this function after calling the timerq_1shot_pulse_set function.
[SFR used]	TQ0CTL0.TQ0CE: 1 (Enables TMQ0 operation.)
[call function]	None
[Variable]	None
[File name]	timerq_1shot_pulse\timerq_1shot_pulse.c
[Caution]	None

Interrupt function

[Function name]	timerq_TQ0CC0_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0CC0	Match between the count value of the 16-bit counter and TQ0CCR0
[call function]	None	
[Variable]	None	
[File name]	timerq_1shot_pulse\timerq_1shot_pulse.c	
[Caution]	None	

[Function name]	timerq_TQ0CC1_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0CC1	Match between the count value of the 16-bit counter and TQ0CCR1
[call function]	None	
[Variable]	None	
[File name]	timerq_1shot_pulse\timerq_1shot_pulse.c	
[Caution]	None	

Figure 4-8. One-Shot Pulse Output Mode (1)

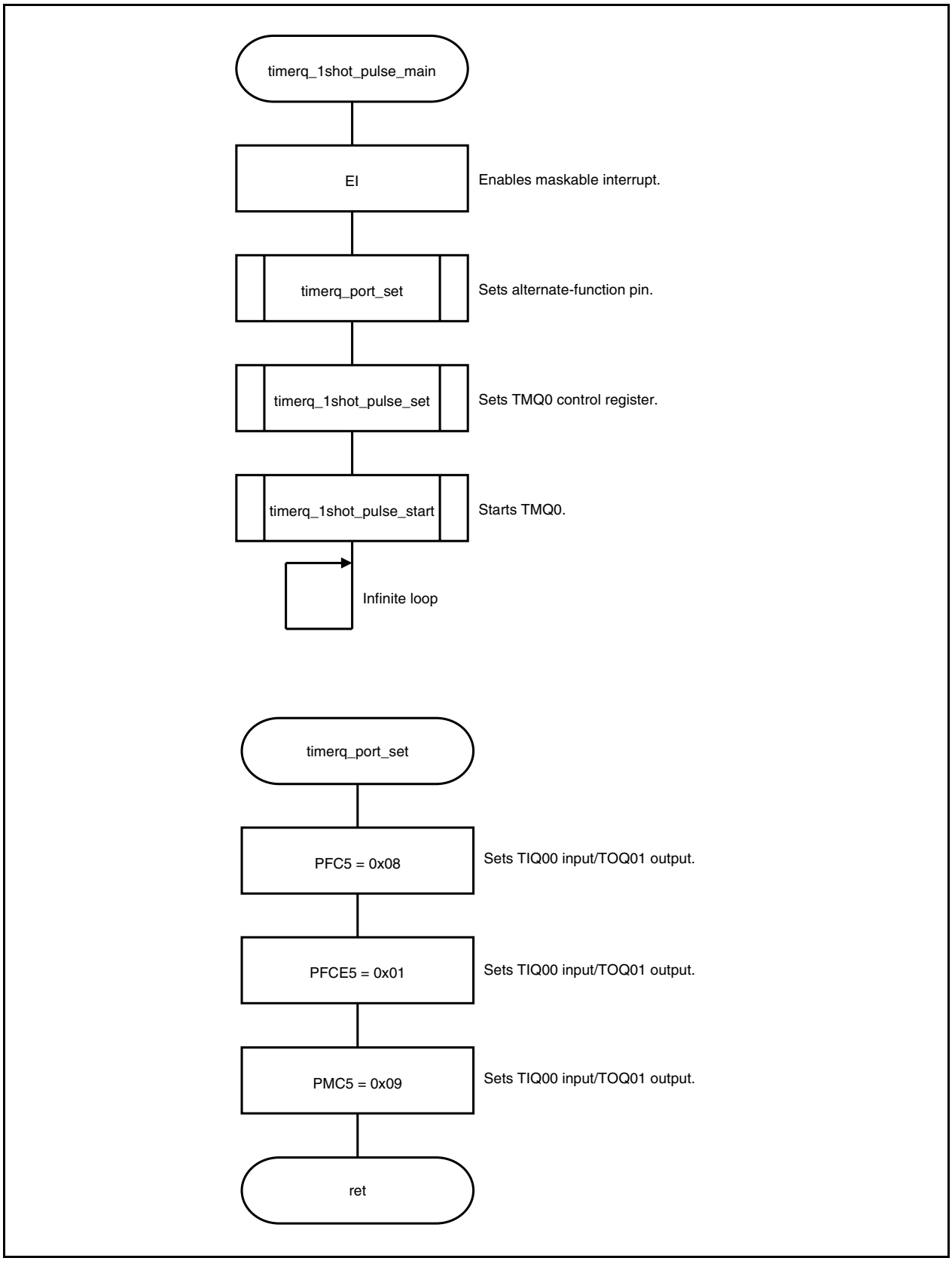


Figure 4-9. One-Shot Pulse Output Mode (2)

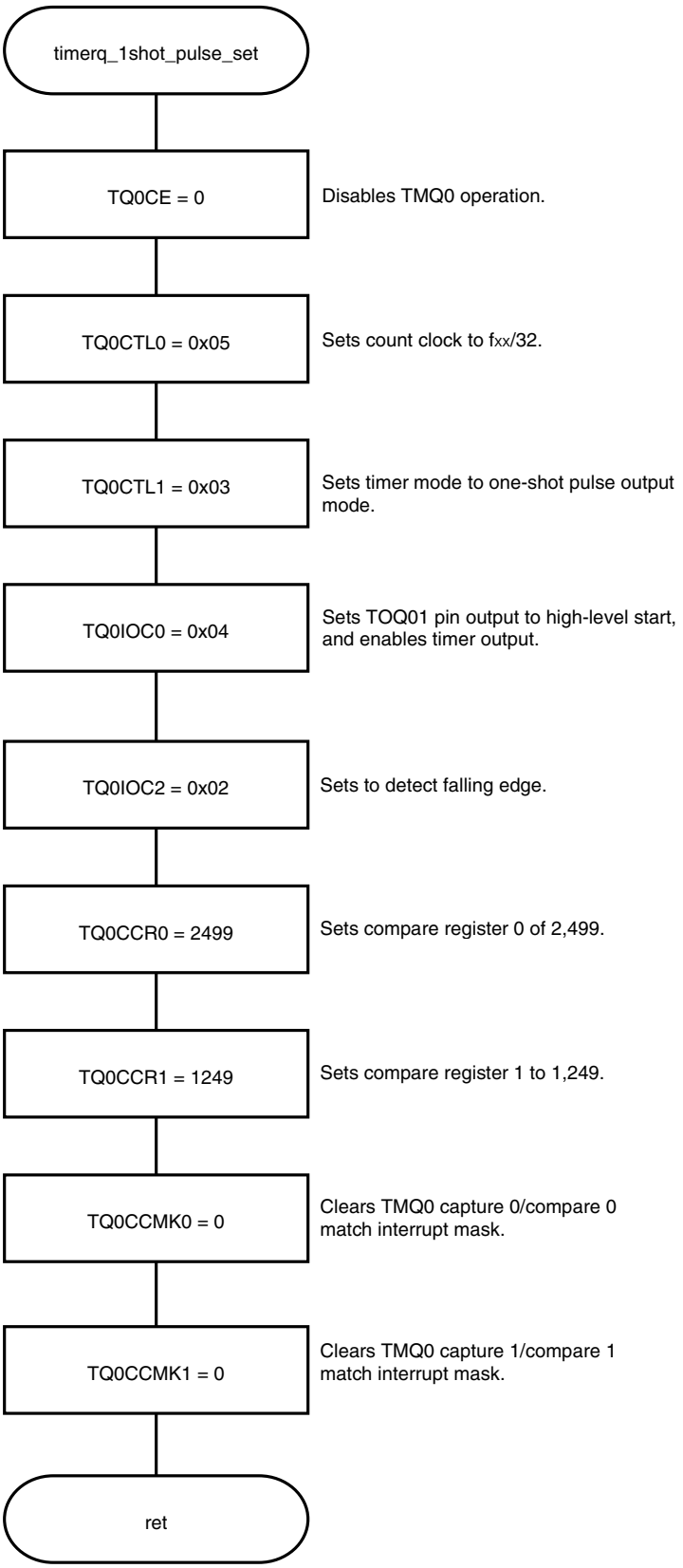
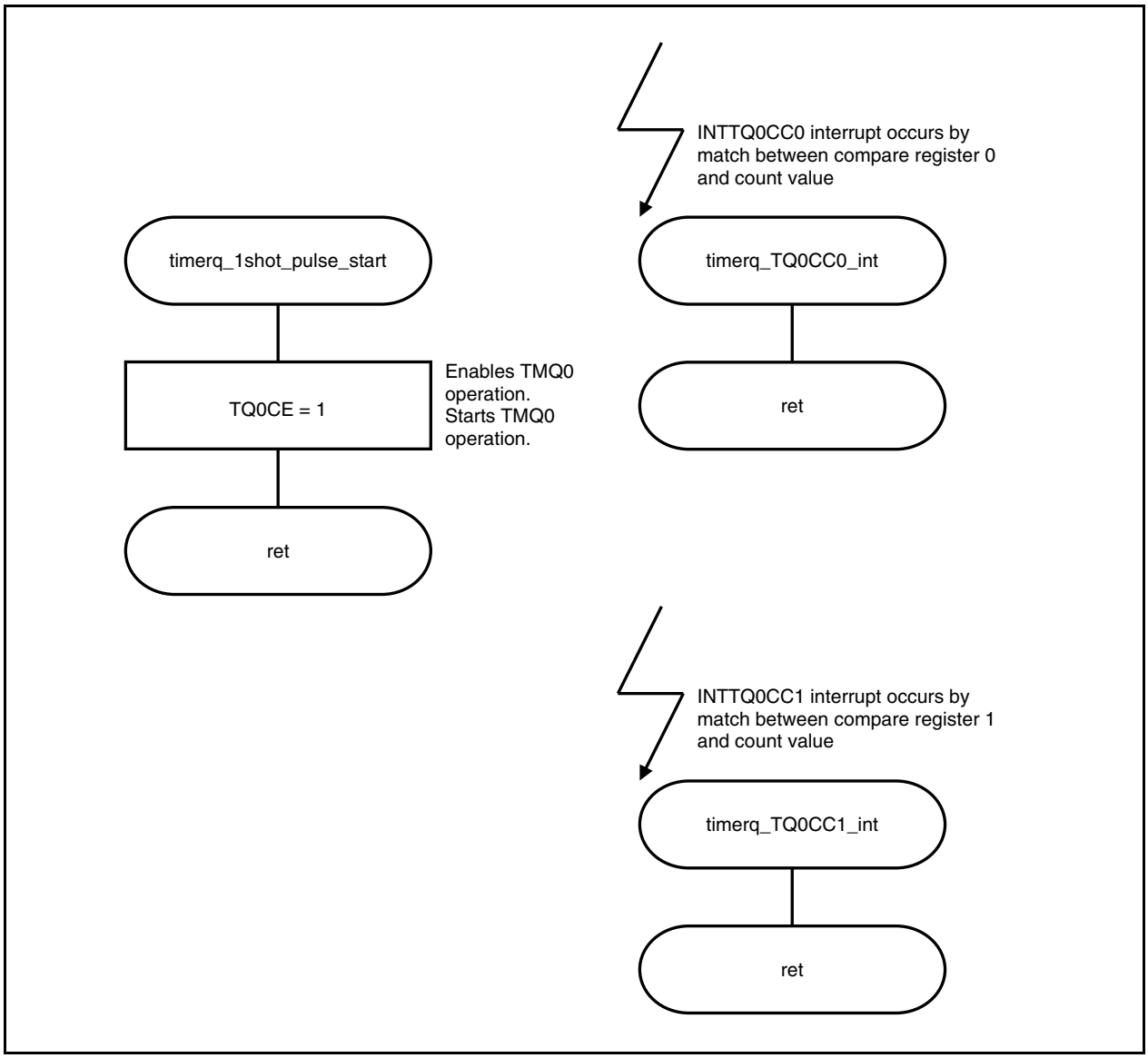


Figure 4-10. One-Shot Pulse Output Mode (3)



4.5 PWM Output Mode

[Function]	Starts operation of the 16-bit counter by setting the TQ0CE bit. Outputs a PWM waveform with a 50% duty factor whose half cycle is equal to the set value of the TQ0CCR0 register + 1, by clearing the 16-bit counter upon a compare match with the TQ0CCR0 register and inverting the TOQ00 pin. Inverts the TOQ01 pin output upon a match between the value set by the TQ0CCR1 register and the count value of the 16-bit counter. Inverts the TOQ01 pin output when the 16-bit counter is cleared.
[Function name]	timerq_pwm_output_main
[Argument]	None
[Processing content]	Performs count operation of an fxx/32 count clock, generates an interrupt by inverting the TOQ00 and TOQ01 pin outputs upon the count subsequent to the count whose value matches the value of the TQ0CCR0 register, and clears the counter. Generates an interrupt by inverting the TOQ01 pin output upon the count subsequent to the count whose value matches the value of the TQ0CCR1 register. The TOQ00 and TOQ01 pins start output at high level.
[SFR used]	None
[call function]	timerq_port_set, timerq_pwm_output_set, timerq_pwm_output_start
[Variable]	None
[Interrupts]	timerq_TQ0CC0_int timerq_TQ0CC1_int
[Interrupt sources]	INTTQ0CC0 INTTQ0CC1
[File name]	timerq_pwm_output\timerq_pwm_output.c,
[Caution]	None

[Function name]	timerq_port_set
[Argument]	None
[Processing content]	Sets port 5 to TOQ00 output pin and TOQ01 output pin.
[SFRs used]	PFC5: 0x08 (Sets TOQ01 output pin, TOQ00 output pin) PFCE5: 0x09 (Sets TOQ01 output pin, TOQ00 output pin) PMC5: 0x09 (Sets TOQ00 output pin, TOQ01 output pin)
[call function]	None
[Variable]	None
[File name]	timerq_pwm_output\timerq_pwm_output.c
[Caution]	None

[Function name]	timerq_pwm_output_set
[Argument]	None
[Processing content]	Sets TMQ0 control register.
[SFRs used]	TQ0CTL0.TQ0CE: 0 (Disables TMQ0 operation.) TQ0CTL0: 0x05 (Sets count clock to fxx/32.) TQ0CTL1: 0x04 (Sets timer mode to one-shot pulse mode.) TQ0IOC0: 0x05 (Sets TOQ00 pin output to high-level start, enables timer output, sets TOQ01 pin output to high-level start, enables timer output.) TQ0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TQ0CCR1: 1249 (Sets 16-bit counter compare register 1 to 1,249.) TQ0CCMK0: 0 (Sets TMQ0 capture 0/compare 0 match interrupt mask clear.) TQ0CCMK1: 0 (Sets TMQ0 capture 1/compare 1 match interrupt mask clear.)
[call function]	None
[Variable]	None
[File name]	timerq_pwm_output\timerq_pwm_output.c
[Caution]	None

The active level width, cycle, and duty of the PWM waveform output from the TOQ01 pin can be calculated by the following formula.

Active level width = (TQ0CCR1 register set value) × count clock cycle

Cycle = (TQ0CCR0 register set value + 1) × count clock cycle

Duty = (TQ0CCR1 register set value) / (TQ0CCR0 register set value + 1)

[Function name]	timerq_pwm_output_start
[Argument]	None
[Processing content]	Starting function of PWM output mode
[Starting method]	Call this function after calling the timerq_pwm_output_set function.
[SFRs used]	TQ0CTL0.TQ0CE: 1 (Enables TMQ0 operation.)
[call function]	None
[Variable]	None
[File name]	timerq_pwm_output\timerq_pwm_output.c
[Caution]	None

Interrupt function

[Function name]	timerq_TQ0CC0_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0CC0	Match between the count value of the 16-bit counter and TQ0CCR0
[call function]	None	
[Variable]	None	
[File name]	timerq_pwm_output\timerq_pwm_output.c	
[Caution]	None	

[Function name]	timerq_TQ0CC1_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0CC1	Match between the count value of the 16-bit counter and TQ0CCR1
[call function]	None	
[Variable]	None	
[File name]	timerq_pwm_output\timerq_pwm_output.c	
[Caution]	None	

Figure 4-11. PWM Output Mode (1)

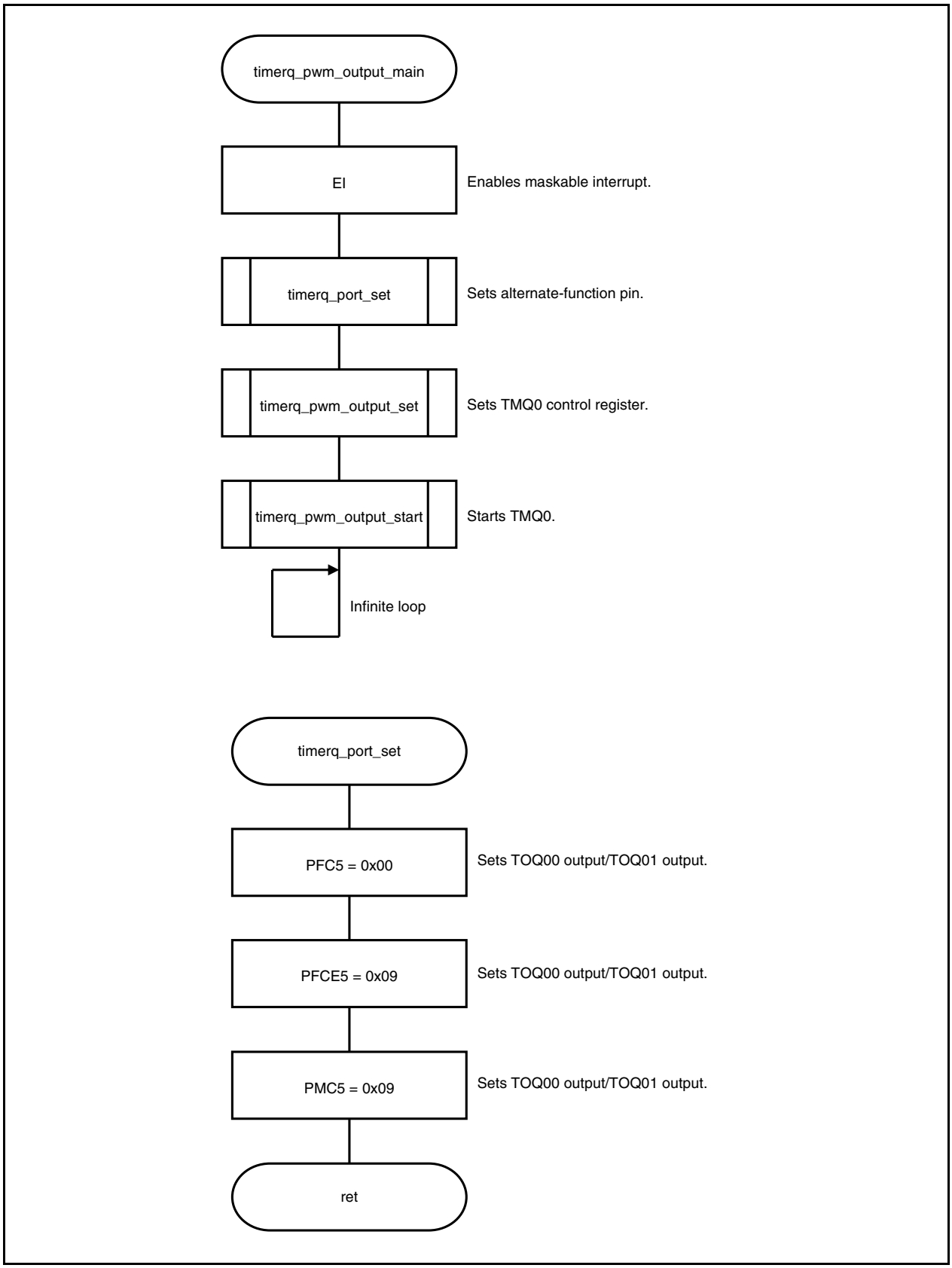


Figure 4-12. PWM Output Mode (2)

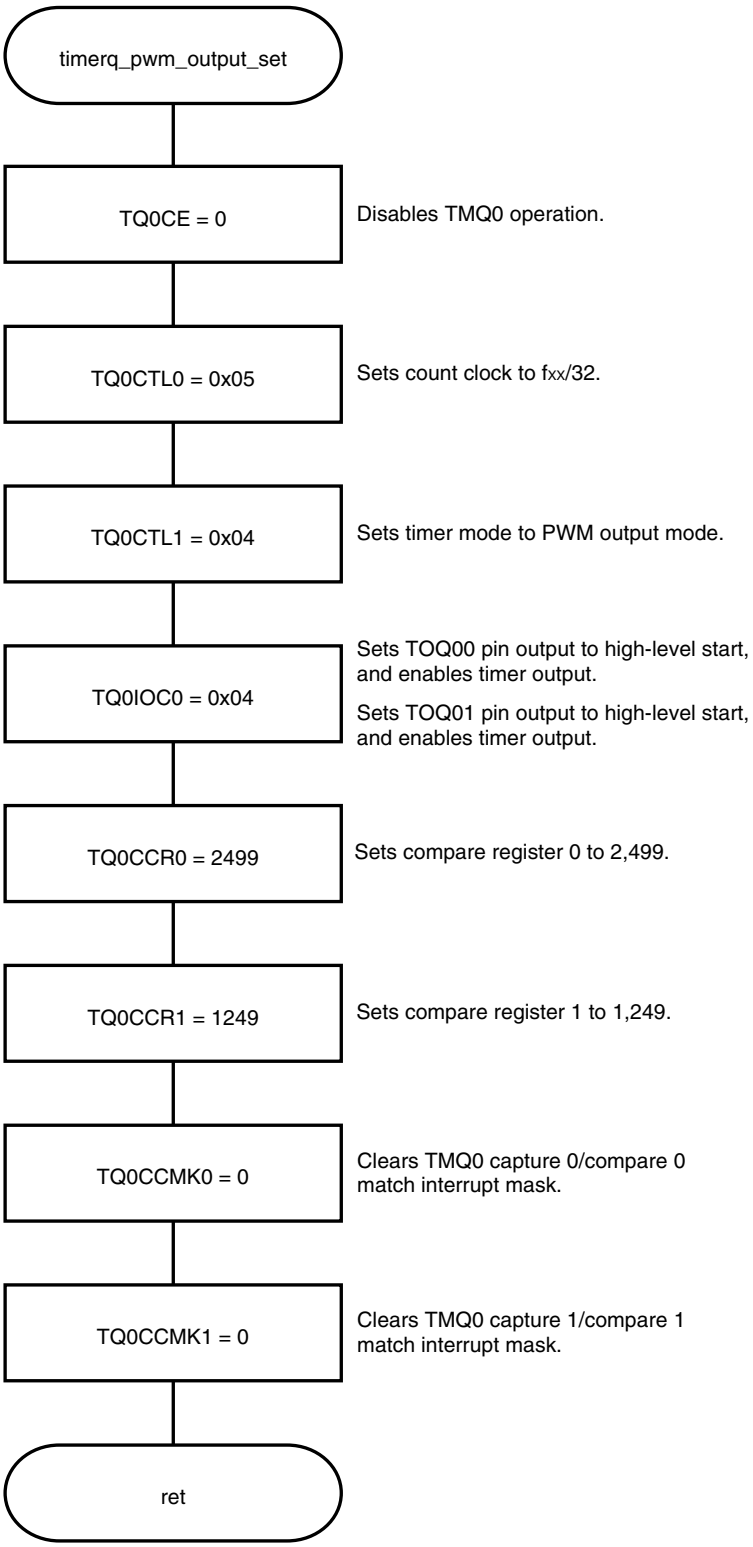
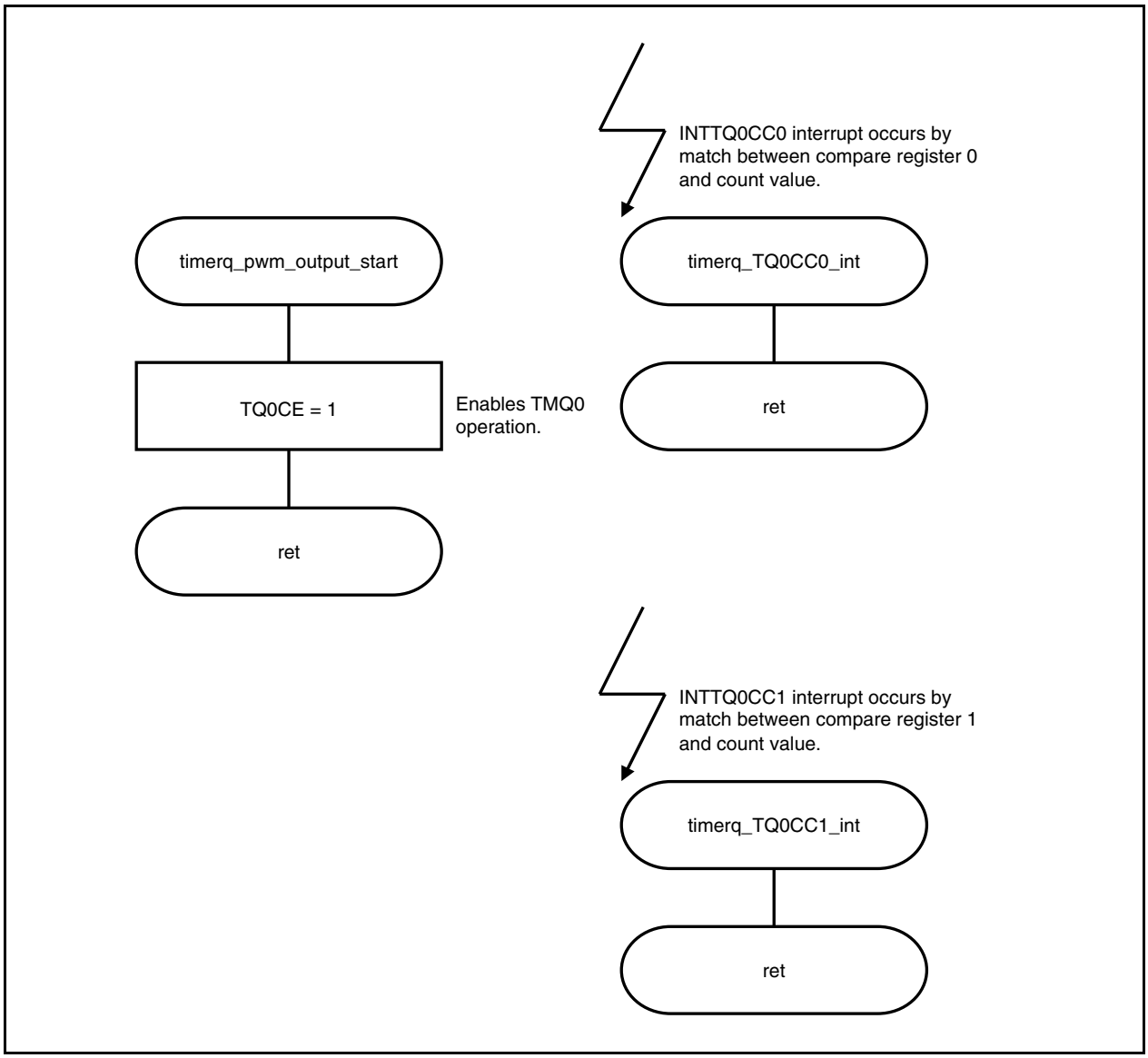


Figure 4-13. PWM Output Mode (3)



4.6 Free-Running Timer Mode

[Function]	Starts operation of the 16-bit counter by setting the TQ0CE bit. Inverts the TOQ00 pin output upon a compare match between the TQ0CCR0 register and the count value of the 16-bit counter (compare function). Even if the TQ0CCR0 register value and the count value of 16-bit counter match, the count value is not cleared. After counting up to FFFFH, generates an overflow interrupt request signal (INTTQ0OV) at the next clock and clears to 0000H to continue counting. Stores the count value of the 16-bit counter when the valid edge of the capture trigger input (TIQ01 pin) is detected (capture function).
[Function name]	timerq_free_running_main
[Argument]	None
[Processing content]	Performs count operation of an fxx/32 count clock, generates an interrupt by inverting the TOQ00 pin output upon the count subsequent to the count whose value matches the value of the TQ0CCR0 register, and clears the counter. Generates an interrupt by capturing the counter value to the TQ0CCR1 register when a valid edge is detected from the TIQ01 pin. Generates the INTTQ0OV interrupt when a counter overflow is detected. The TOQ00 pin start output at high level.
[SFR used]	None
[call function]	timerq_port_set, timerq_free_running_set, timerq_free_running_start
[Variable]	None
[Interrupts]	timerq_TQ0CC0_int timerq_TQ0CC1_int timerq_TQ0OV_int
[Interrupt sources]	INTTQ0CC0 INTTQ0CC1 INTTQ0OV
[File name]	timerq_free_running\timerq_free_running.c
[Caution]	None

[Function name]	timerq_port_set
[Argument]	None
[Processing content]	Sets port 5 to TOQ00 output pin, TIQ01 input pin.
[SFRs used]	PFC5: 0x01 (Sets TOQ00 output pin, TIQ01 input pin.) PFCE5: 0x08 (Sets TOQ00 output pin, TIQ01 input pin.) PMC5: 0x09 (Sets TIQ01 input pin, TOQ00 output pin.)
[call function]	None
[Variable]	None
[File name]	timerq_free_running\timerq_free_running.c
[Caution]	None

[Function name]	timerq_free_running_set
[Argument]	None
[Processing content]	Sets TMQ0 control register
[SFRs used]	TQ0CTL0.TQ0CE: 0 (Disables TMQ0 operation.) TQ0CTL0: 0x05 (Sets count clock to fxx/32.) TQ0CTL1: 0x05 (Sets timer mode to free-running timer mode.) TQ0IOC0: 0x01 (Sets TOQ00 pin output to high-level start, enables timer output.) TQ0IOC1: 0x08 (Sets TIQ01 pin valid edge to falling edge.) TQ0OPT0: 0x20 (Sets TQ0CCR1 to capture register.) TQ0CCR0: 2499 (Sets 16-bit counter compare register 0 to 2,499.) TQ0CCMK0: 0 (Sets TMQ0 capture 0/compare 0 match interrupt mask clear.) TQ0CCMK1: 0 (Sets TMQ0 capture 1/compare 1 match interrupt mask clear.) TQ0OVMK: 0 (Sets to TMQ0 overflow interrupt mask clear.)
[call function]	None
[Variable]	None
[File name]	timerq_free_running\timerq_free_running.c
[Caution]	None

[Function name]	timerq_free_running_start
[Argument]	None
[Processing content]	Starting function of free-running timer mode.
[Starting method]	Call this function after calling the timerq_free_running_set function.
[SFR used]	TQ0CTL0.TQ0CE: 1 (Enables TMQ0 operation.)
[call function]	None
[Variable]	None
[File name]	timerq_free_running\timerq_free_running.c
[Caution]	None

Interrupt function

[Function name]	timerq_TQ0CC0_int
[Overview]	Defined by the user.
[Factor]	INTTQ0CC0 Match between the count value of the 16-bit counter and TQ0CCR0
[call function]	None
[Variable]	None
[File name]	timerq_free_running\timerq_free_running.c
[Caution]	None

[Function name]	timerq_TQ0CC1_int
[Overview]	Defined by the user.
[Factor]	INTTQ0CC1 TIQ01 pin input valid edge detection.
[call function]	None
[Variable]	None
[File name]	timerq_free_running\timerq_free_running.c
[Caution]	None

[Function name]	timerq_TQ0OV_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0OV	16-bit counter overflow occurrence.
[call function]	None	
[Variable]	None	
[File name]	timerq_free_running\timerq_free_running.c	
[Caution]	None	

Figure 4-14. Free-Running Timer Mode (1)

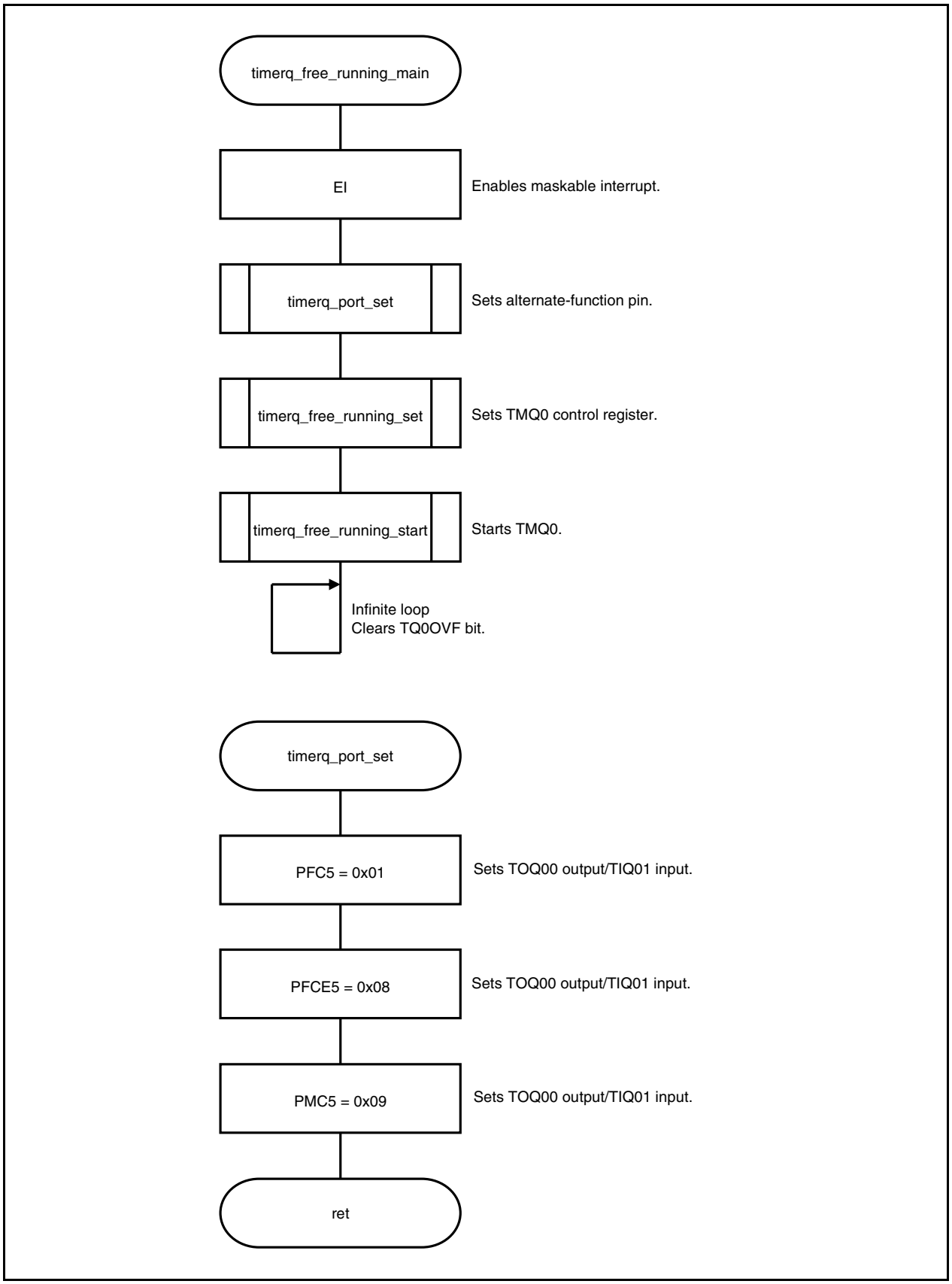


Figure 4-15. Free-Running Timer Mode (2)

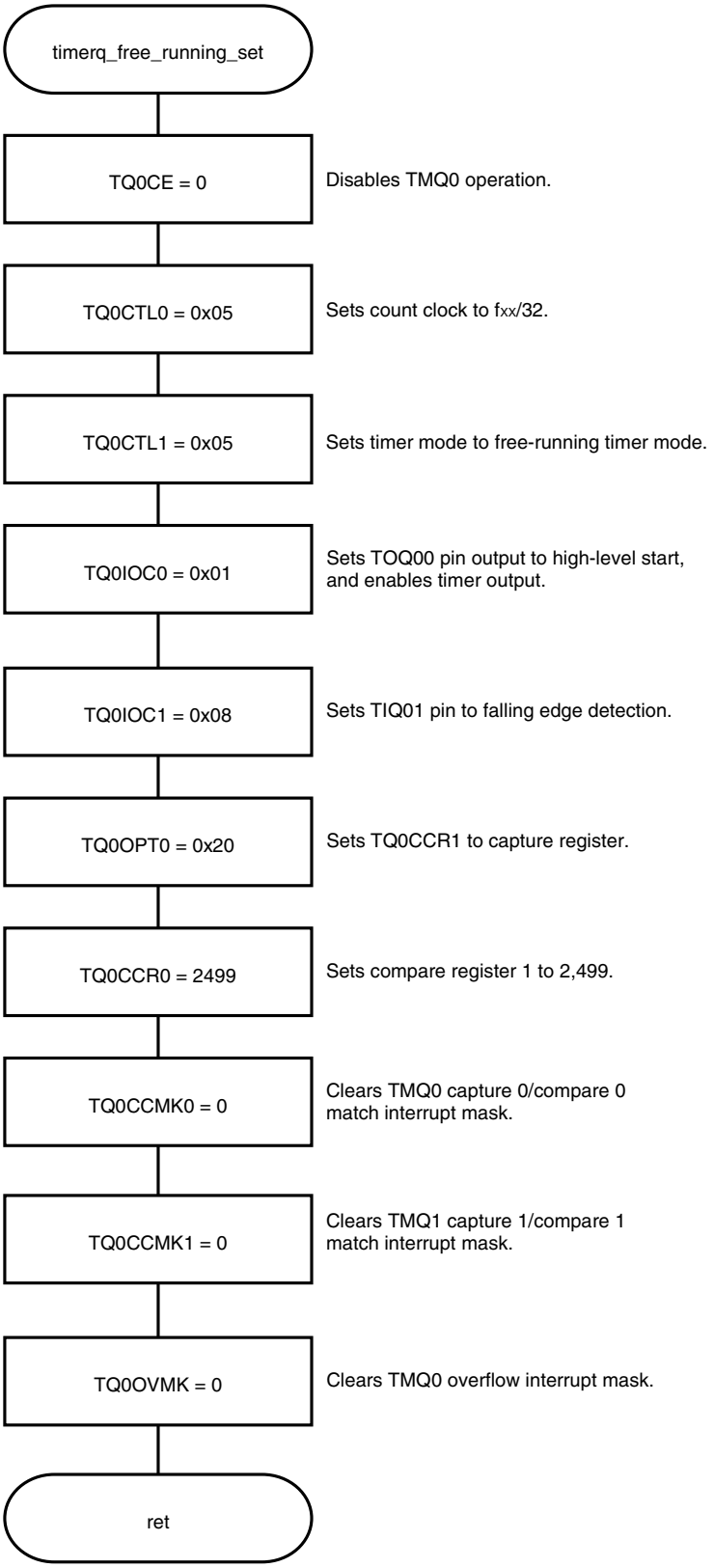
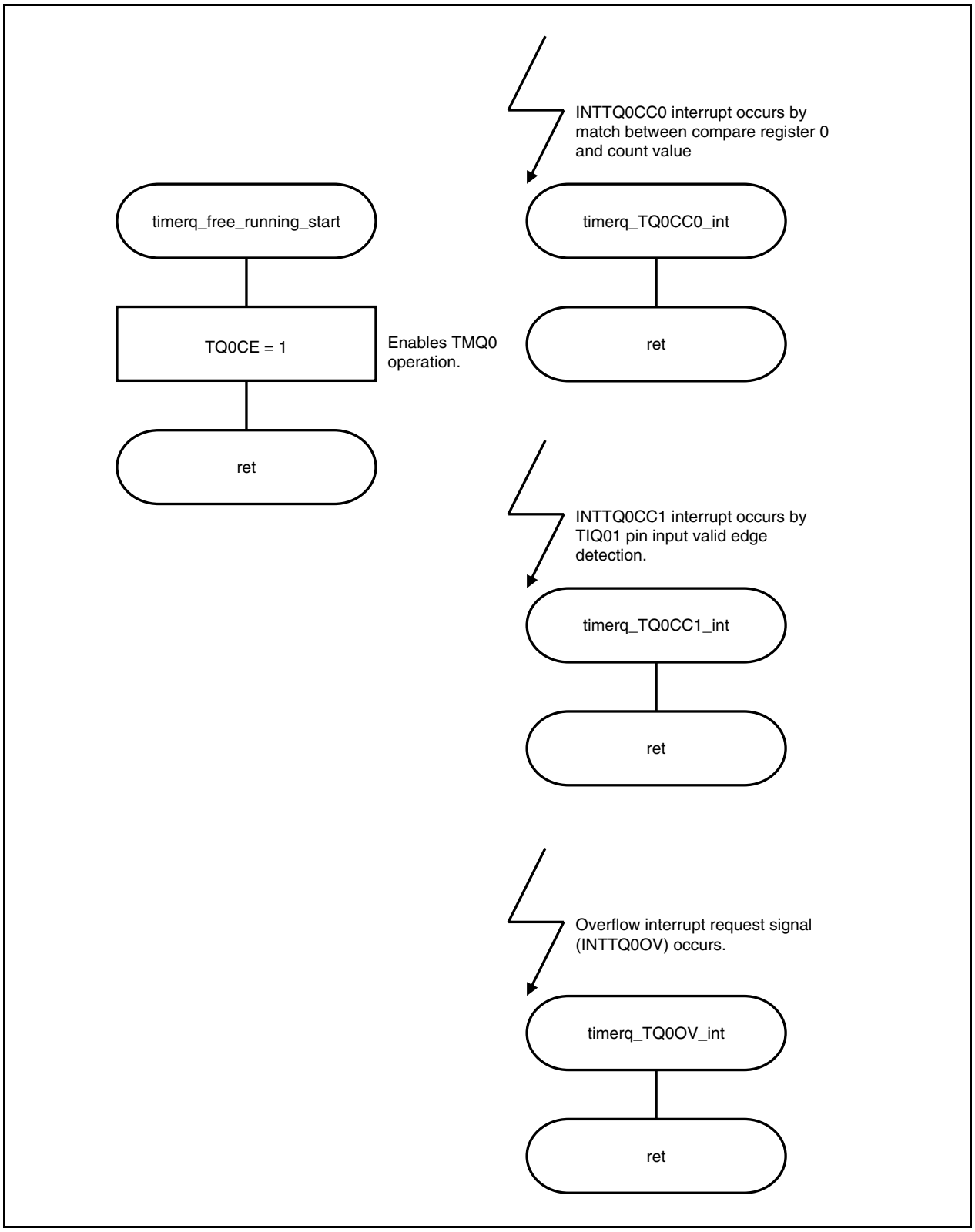


Figure 4-16. Free-Running Timer Mode (3)



4.7 Pulse Width Measurement Mode

[Function]	Starts operation of the 16-bit counter by setting the TQ0CE bit. Clears the 16-bit counter by storing the count value to the TQ0CCR0 register when the valid edge of the capture trigger input (TIQ00 pin) is detected. Measures the valid edge interval of the TIQ00 pin by generating an interrupt when the valid edge of the TIQ00 pin input is detected and reading the TQ0CCR0 register value.
[Function name]	timerq_pulse_measure
[Argument]	None
[Processing content]	Performs count operation of an fxx/32 count clock, generates an interrupt by storing the count value of the 16-bit counter to the TQ0CCR0 register when the valid edge of the TIQ00 pin input is detected, and clears the counter. Generates an INTTQ0OV interrupt when a counter overflow is detected.
[SFRs used]	None
[call function]	timerq_port_set, timerq_pulse_measure_set, timerq_ pulse_measure_start
[Variable]	None
[Interrupts]	timerq_TQ0CC0_int timerq_TQ0OV_int
[Interrupt sources]	INTTQ0CC0 INTTQ0OV
[File name]	timerq_pulse_measure\timerq_pulse_measure.c,
[Caution]	None

[Function name]	timerq_port_set
[Argument]	None
[Processing content]	Sets port 5 to TIQ00 input pin.
[SFRs used]	PFC5: 0x08 (Sets TIQ00 input pin.) PMC5: 0x08 (Sets TIQ00 input pin.)
[call function]	None
[Variable]	None
[File name]	timerq_pulse_measure\timerq_pulse_measure.c
[Caution]	None

[Function name]	timerq_free_running_set
[Argument]	None
[Processing content]	Sets TMQ0 control register.
[SFRs used]	TQ0CTL0.TQ0CE: 0 (Disables TMQ0 operation.) TQ0CTL0: 0x05 (Sets count clock to fxx/32.) TQ0CTL1: 0x06 (Sets timer mode to pulse width measurement mode.) TQ0IOC1: 0x02 (Sets TIQ00 pin to falling edge.) TQ0IOC2: 0x08 (Sets TIQ00 pin to falling edge.) TQ0CCMK0: 0 (Sets TMQ0 capture 0/compare 0 match interrupt mask clear.) TQ0OVMK: 0 (Sets to TMQ0 overflow interrupt mask clear.)
[call function]	None
[Variable]	None
[File name]	timerq_pulse_measure\timerq_pulse_measure.c
[Caution]	None

The pulse width can be calculated by the following formula.

Pulse width = (TQ0CCR0 register value + 1) × count clock cycle

The pulse width when the 16-bit counter overflow is detected can be calculated by the following formula.

Pulse width = (TQ0CCR0 register value + 0x10001) × count clock cycle

[Function name]	timerq_pulse_measure_start
[Argument]	None
[Processing content]	Starting function of pulse width measurement mode
[Starting method]	Call this function after calling the timerq_pulse_measure_set function.
[SFR used]	TQ0CTL0.TQ0CE: 1 (Enables TMQ0 operation.)
[call function]	None
[Variable]	None
[File name]	timerq_pulse_measure\timerq_pulse_measure.c
[Caution]	None

Interrupt function

[Function name]	timerq_TQ0CC0_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0CC0	TIQ00 pin input valid edge detection
[call function]	None	
[Variable]	None	
[File name]	timerq_pulse_measure\timerq_pulse_measure.c	
[Caution]	None	

[Function name]	timerq_TQ0OV_int	
[Overview]	Defined by the user.	
[Factor]	INTTQ0OV	16-bit counter overflow occurrence.
[call function]	None	
[Variable]	None	
[File name]	timerq_pulse_measure\timerq_pulse_measure.c	
[Caution]	None	

Figure 4-17. Pulse Width Measurement Mode (1)

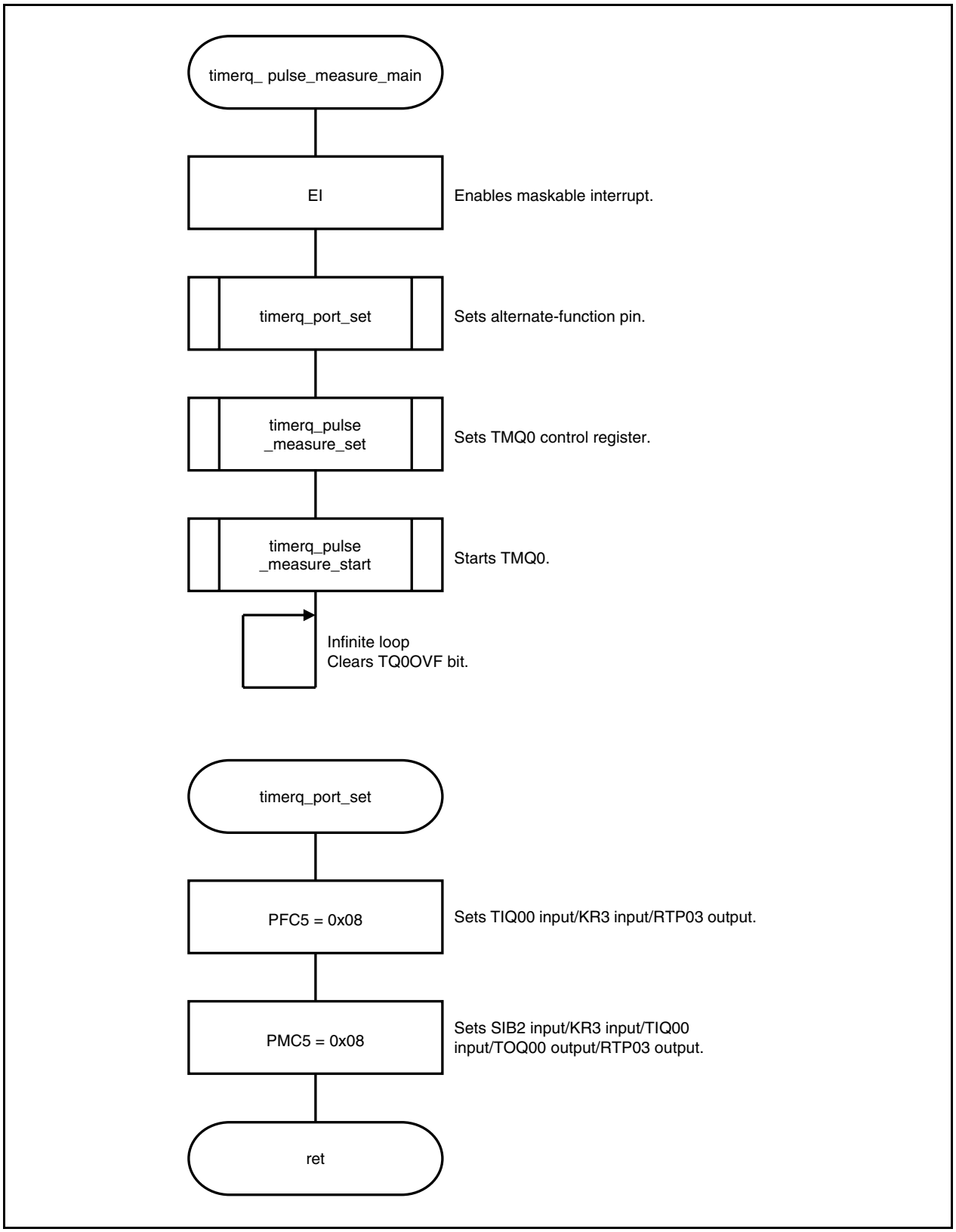


Figure 4-18. Pulse Width Measurement Mode (2)

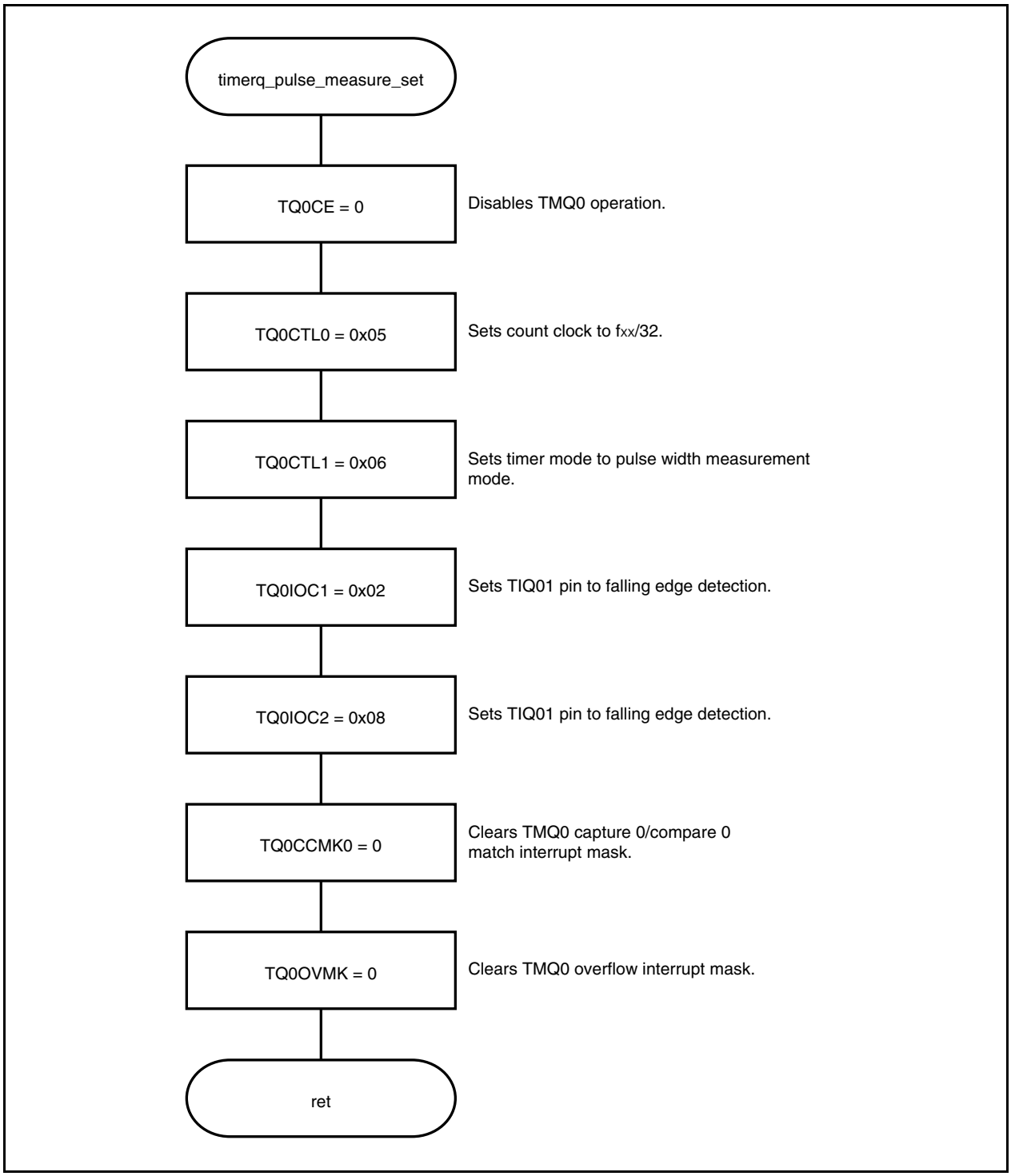
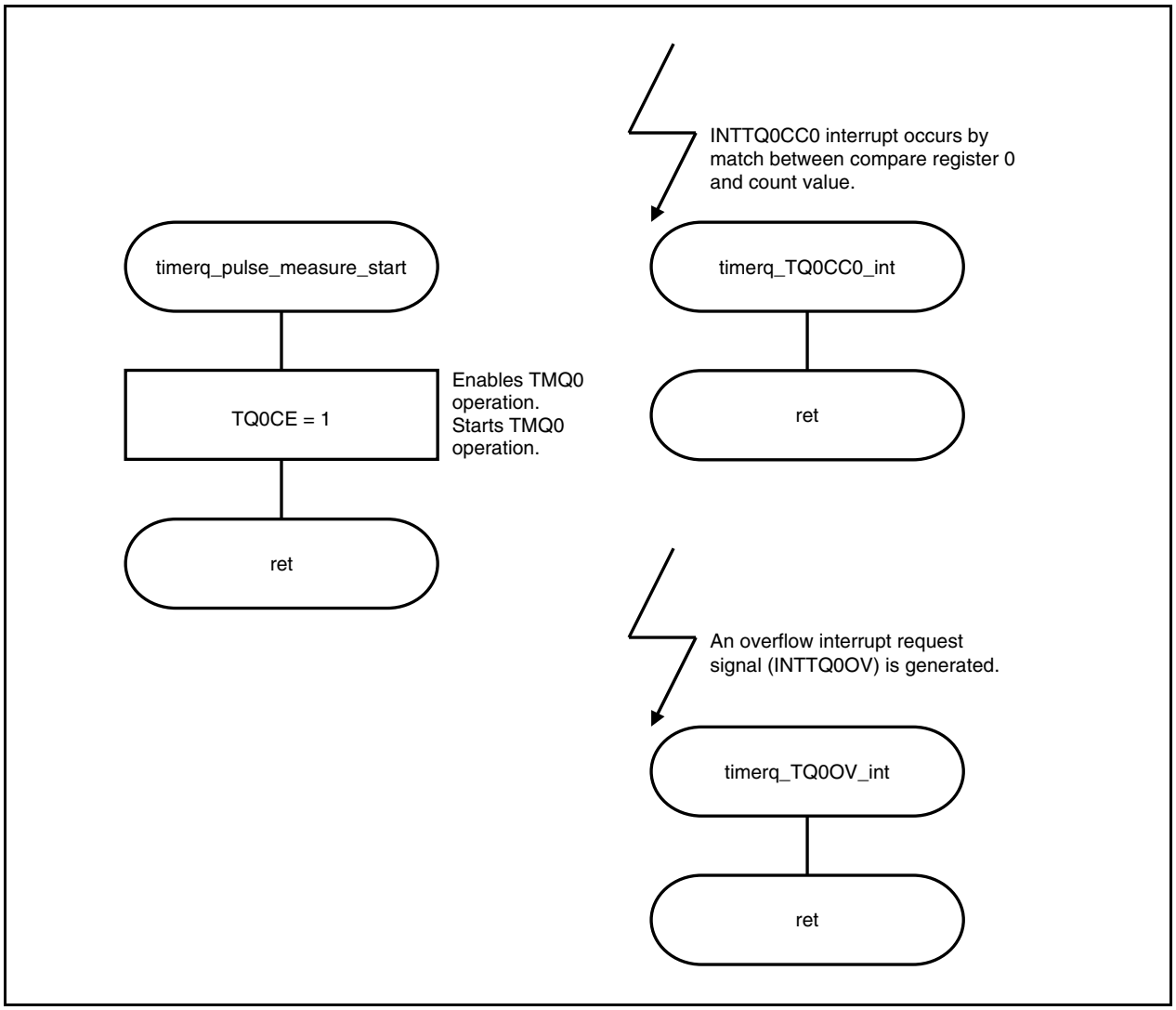


Figure 4-19. Pulse Width Measurement Mode (3)



CHAPTER 5 16-BIT INTERVAL TIMER M (TMM)

5.1 Interval Timer Mode

[Function]	Starts 16-bit counter operation by setting the TM0CE bit. Outputs an interrupt request signal (INTTM0EQ0) at an interval set by the TM0CMP0 register.
[Function name]	timerm_interval_main
[Argument]	None
[Processing content]	Performs count operation of an fxx/4 count clock and generates an interrupt upon the count timing subsequent to the count whose value matches the value of the TM0CMP0 register.
[SFR used]	None
[call function]	timerm_interval_set, timerm_interval_start
[Variable]	None
[Interrupt]	timerm_TM0EQ0_int
[Interrupt source]	INTTM0EQ0
[File name]	timerm_interval\timerm_interval.c
[Caution]	None

[Function name]	timerm_interval_set
[Argument]	None
[Processing content]	Sets the TMM0 control register.
[SFRs used]	TM0CTL0.TM0CE: 0 (Disables TMM0 operation.) TM0CTL0: 0x02 (Sets count clock to fxx/4.) TM0CMP0: 1249 (Sets 16-bit counter compare register 0 to 1,249.) TM0EQMK0: 0 (Clears TMM0 compare match interrupt mask.)
[call function]	None
[Variable]	None
[File name]	timerm_interval\timerm_interval.c
[Caution]	None

The setting time is determined by the following formula (during clock-through mode (5 MHz))

Synchronizing clock: 5 MHz = 1/5 ⇒ 0.2 μs
Count clock frequency (for fxx/4): 0.2 μs × 4 ⇒ 0.8 μs
Value of compare register when generating an interrupt every millisecond: 1000 μs/0.8 μs ⇒ 1250 – 1 = 1249

[Function name]	timerm_interval_start
[Argument]	None
[Processing content]	Starting function of interval timer mode
[Starting method]	Call after timerm_interval_set function.
[SFR used]	TM0CTL0.TM0CE: 1 (Enables TMM0 operation.)
[call function]	None
[Variable]	None
[File name]	timerm_interval\timerm_interval.c
[Caution]	None

Interrupt function

[Function name]	timerm_TM0EQ0_int
[Overview]	Defined by the user.
[Factor]	INTTM0EQ0 Match between the count value of the 16-bit counter and TM0CMP0
[call function]	None
[Variable]	None
[File name]	timerm_interval\timerm_interval.c
[Caution]	None

Figure 5-1. Interval Timer Mode (1)

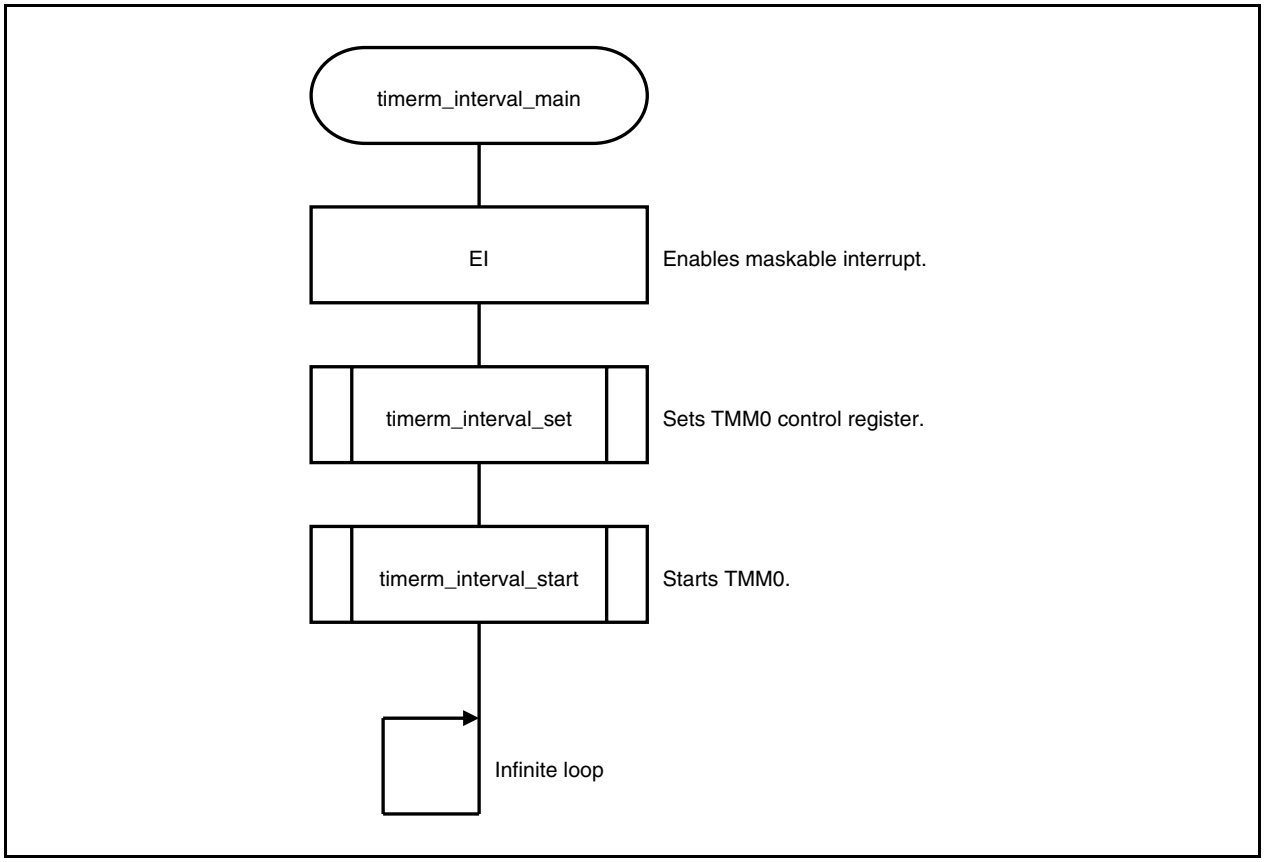
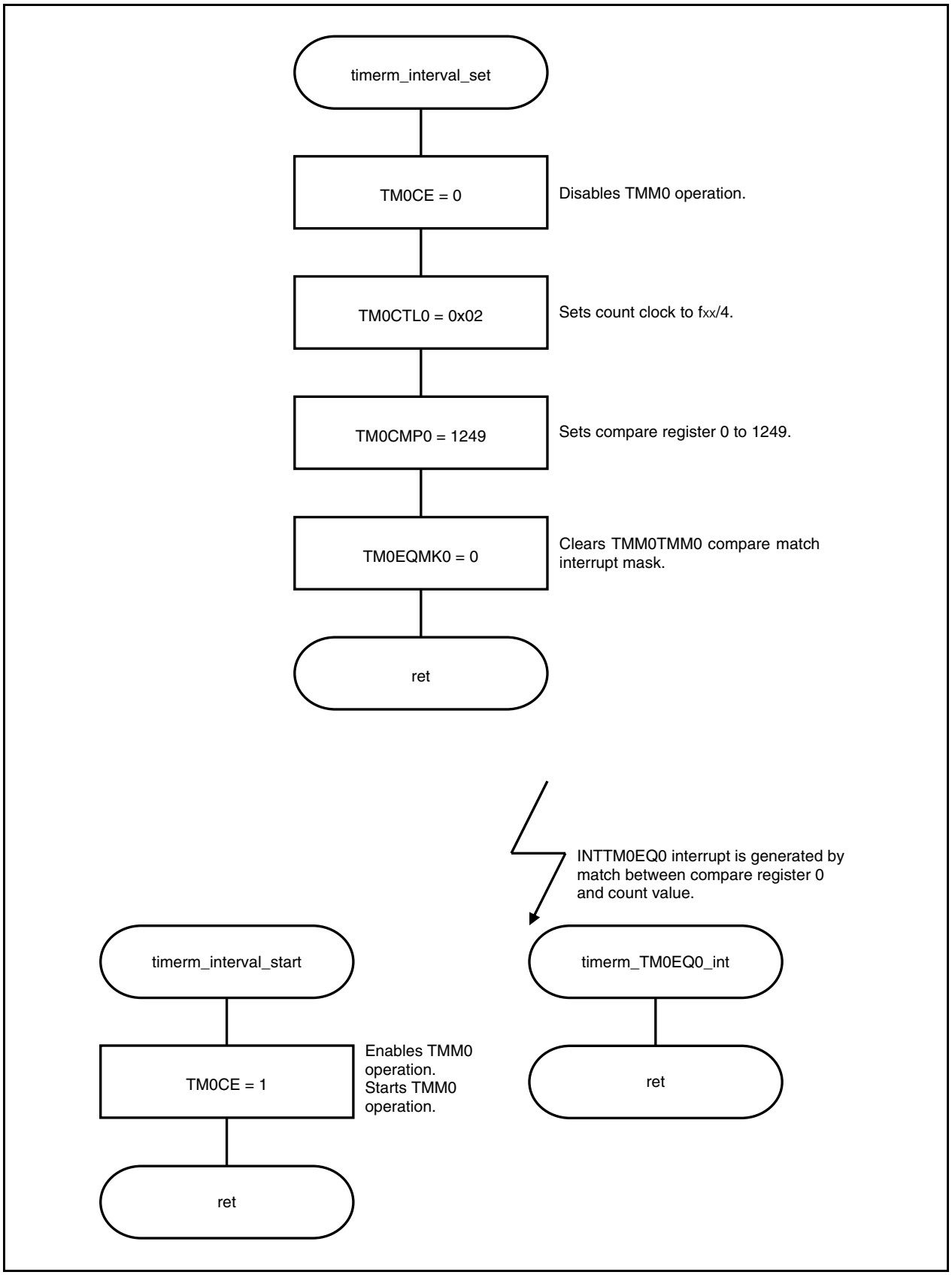


Figure 5-2. Interval Timer Mode (2)



CHAPTER 6 WATCH TIMER FUNCTION

6.1 Watch Timer

[Function]	Starts count operation by setting the WTM.WTM1 and WTM0 bits to 11. Operates as a watch timer with time intervals of 0.5 second using the main clock.
[Function name]	timer_time_main
[Argument]	None
[Processing content]	Generates an interrupt request signal (INTWT) at the time interval of 0.5 second.
[SFR used]	None
[call functions]	timer_time_set, timer_time_start
[Variable]	None
[Interrupt]	timer_WT_int
[Interrupt source]	INTWT
[File name]	timer_time\timer_time.c
[Caution]	Some time is required before the first watch timer interrupt request signal (INTWT) is generated after operation is enabled (WTM.WTM1 and WTM.WTM0 bits = 1).

[Function name]	timer_time_set
[Argument]	None
[Processing content]	Sets the watch timer control register.
[SFRs used]	WTM.WTM0: 0 (Sets to clear after the 5-bit counter operation is stopped.) WTM.WTM1: 0 (Sets to disable the watch timer.) WTM: 0x00 (Sets the setting time of watch flag to 0.5 second.) PRSCM0: 0x13 (Sets the prescaler compare register 0 to 19.) PRSM0: 0x12 (Enables the main clock operation and sets the watch timer source clock to fx/4.) WTMK: 0 (Sets to clear the reference time interrupt mask of watch timer.)
[call function]	None
[Variable]	None
[File name]	timer_time\timer_time.c
[Caution]	None

[Function name]	timer_time_start
[Argument]	None
[Processing content]	Starting function of watch timer
[Starting method]	Call this function after calling the timer_time_set function.
[SFRs used]	WTM.WTM0: 1 (Sets to start the 5-bit counter.) WTM.WTM1: 1 (Sets to enable the operation of watch timer.)
[call function]	None
[Variable]	None
[File name]	timer_time\timer_time.c
[Caution]	None

Interrupt function

[Function name]	timer_WT_int
[Overview]	Defined by the user.
[Factor]	INTWT elapsed time of 0.5 second
[call function]	None
[Variable]	None
[File name]	timer_time\timer_time.c
[Caution]	None

Figure 6-1. Watch Timer (1)

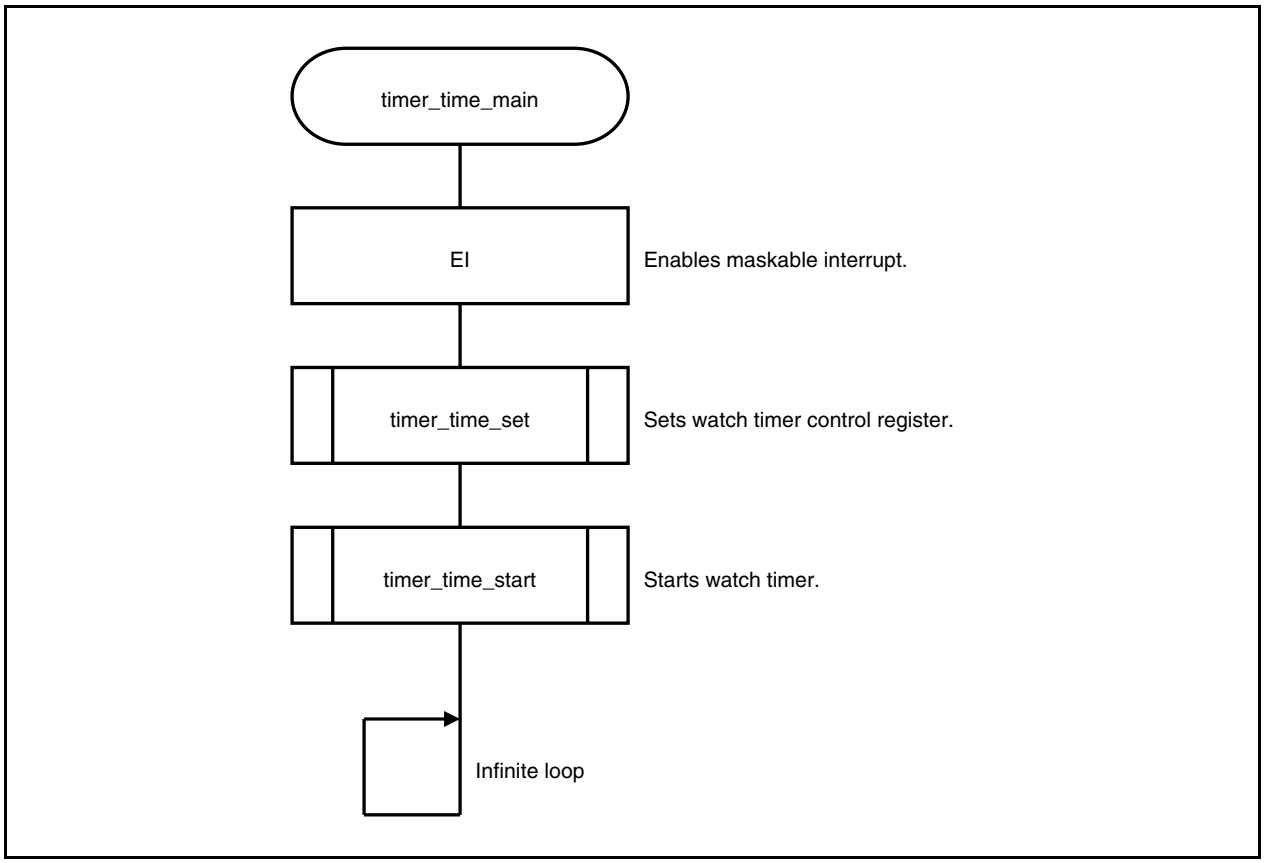
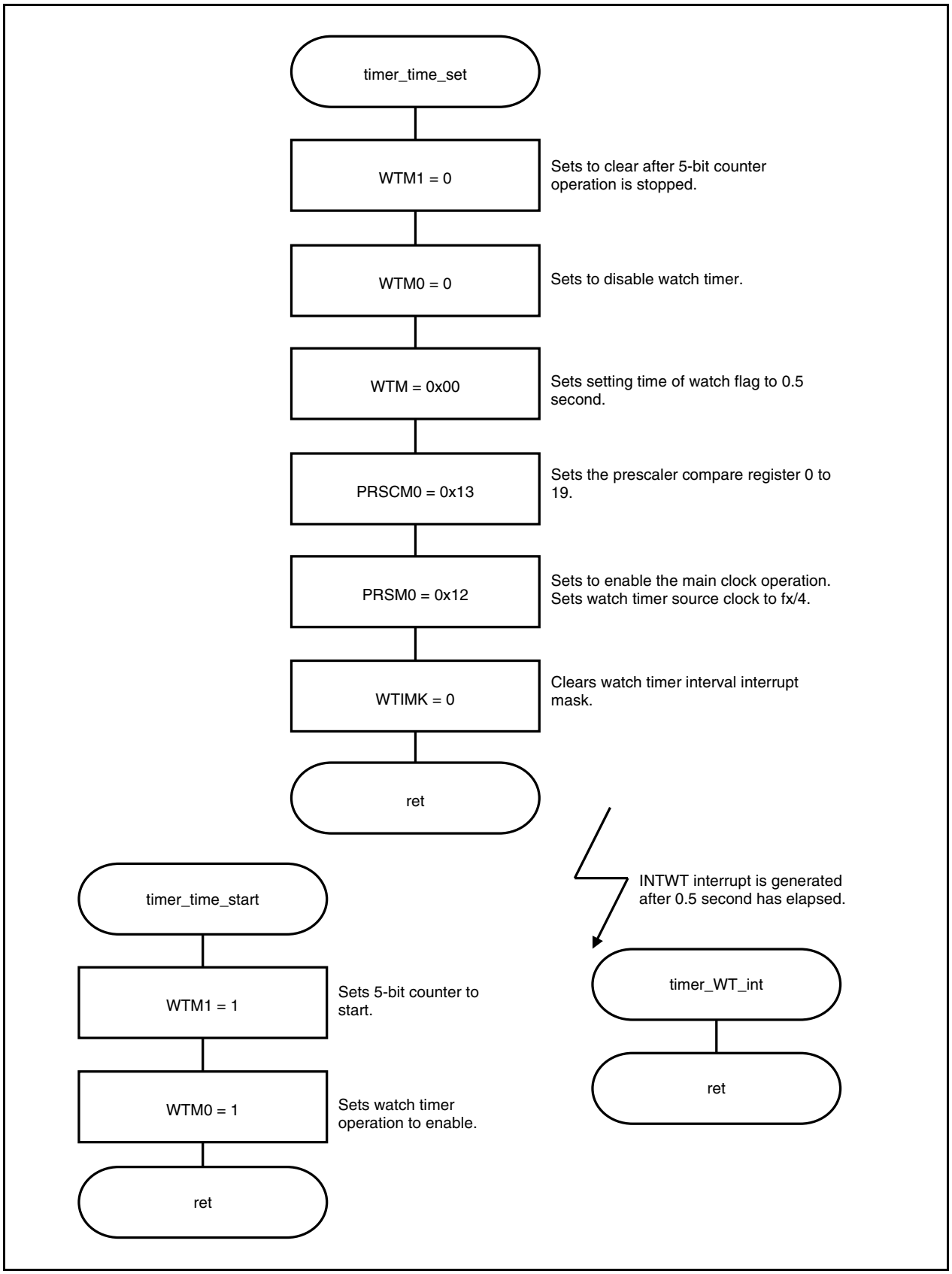


Figure 6-2. Watch Timer (2)



6.2 Interval Timer

[Function]	Operates as the interval timer generating the interrupt request signal (INTWTI) repeatedly at the interval of the count value set at the WTM.WTM4 to WTM.WTM7 bits.
[Function name]	timer_interval_main
[Argument]	None
[Processing content]	Generates the interrupt request signal (INTWTI) at the interval of set value.
[SFR used]	None
[call functions]	timer_interval_set, timer_interval_start
[Variable]	None
[Interrupt]	timer_WTI_int
[Interrupt source]	INTWTI
[File name]	timer_interval\timer_interval.c
[Caution]	None

[Function name]	timer_interval_set
[Argument]	None
[Processing content]	Sets the watch timer control register.
[SFRs used]	WTM.WTM0: 0 (Sets to clear after 5-bit counter operation is stopped.) WTM.WTM1: 0 (Sets to disable the watch timer.) WTM: 0x00 (Sets the prescaler interval time to 488 μs.) PRSCM0: 0x13 (Sets the prescaler compare register 0 to 19.) PRSM0: 0x12 (Enables the main clock operation and sets the watch timer source clock to fx/4.) WTIMK: 0 (Sets to clear the watch timer interval interrupt mask.)
[call function]	None
[Variable]	None
[File name]	timer_interval\timer_interval.c
[Caution]	None

[Function name]	timer_interval_start
[Argument]	None
[Processing content]	Starting function of interval timer
[Starting method]	Call after calling the timer_interval_set function.
[SFRs used]	WTM.WTM0: 1 (Sets to start the 5-bit counter.) WTM.WTM1: 1 (Sets to enable the operation of watch timer.)
[call function]	None
[Variable]	None
[File name]	timer_interval\timer_interval.c
[Caution]	None

Interrupt function

[Function name]	timer_WT_int
[Overview]	Defined by the user.
[Factor]	INTWTI Elapse of the interval time
[call function]	None
[Variable]	None
[File name]	timer_interval\timer_interval.c
[Caution]	None

Figure 6-3. Interval Timer (1)

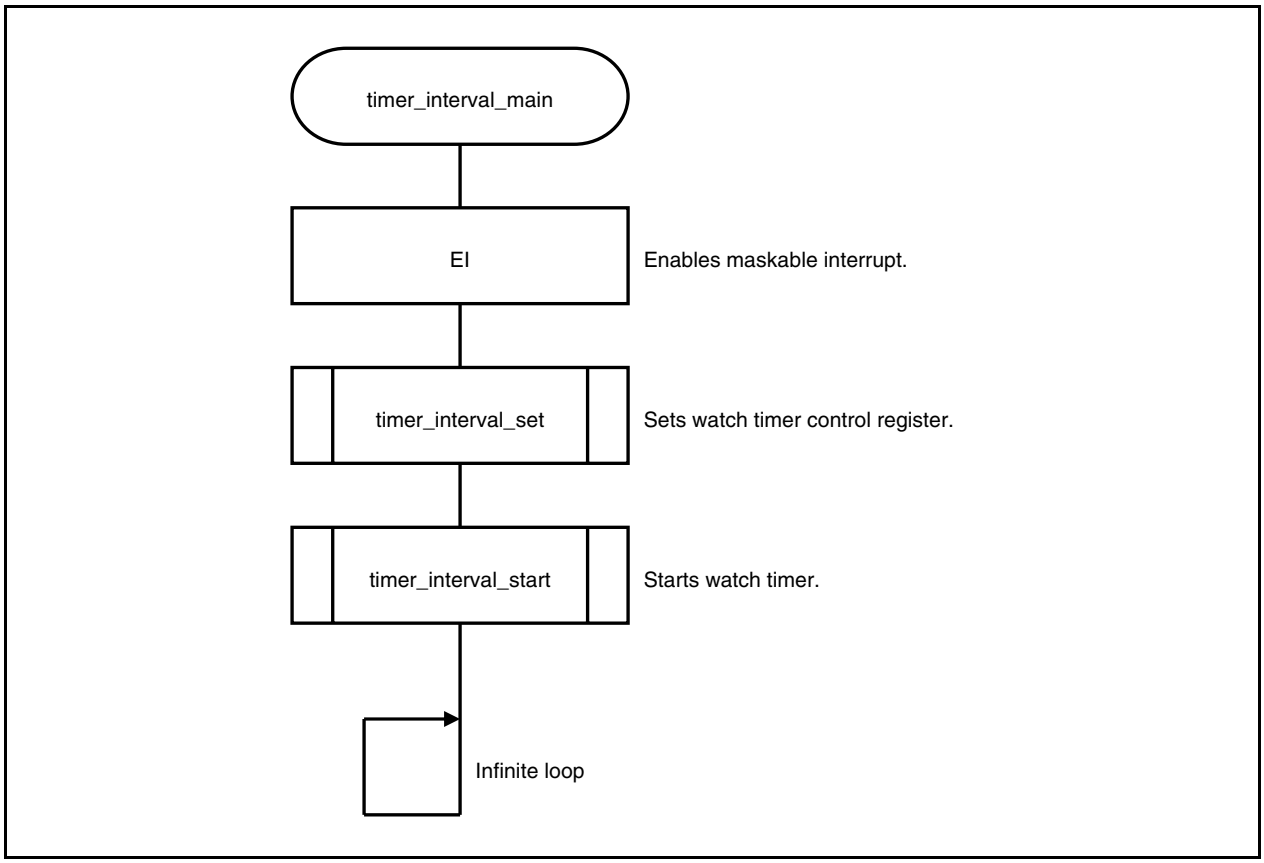
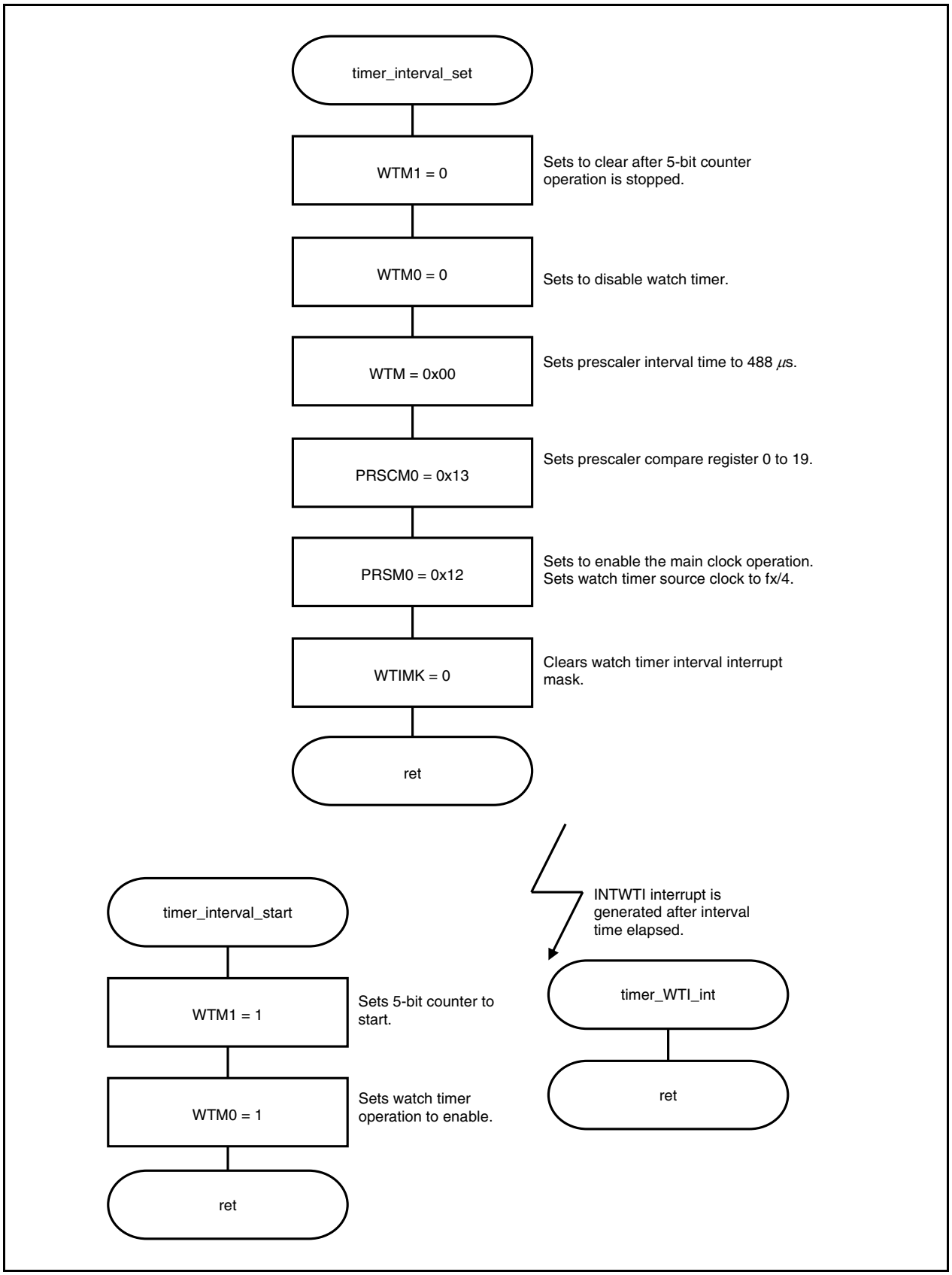


Figure 6-4. Interval Timer (2)



CHAPTER 7 WATCHDOG TIMER FUNCTION

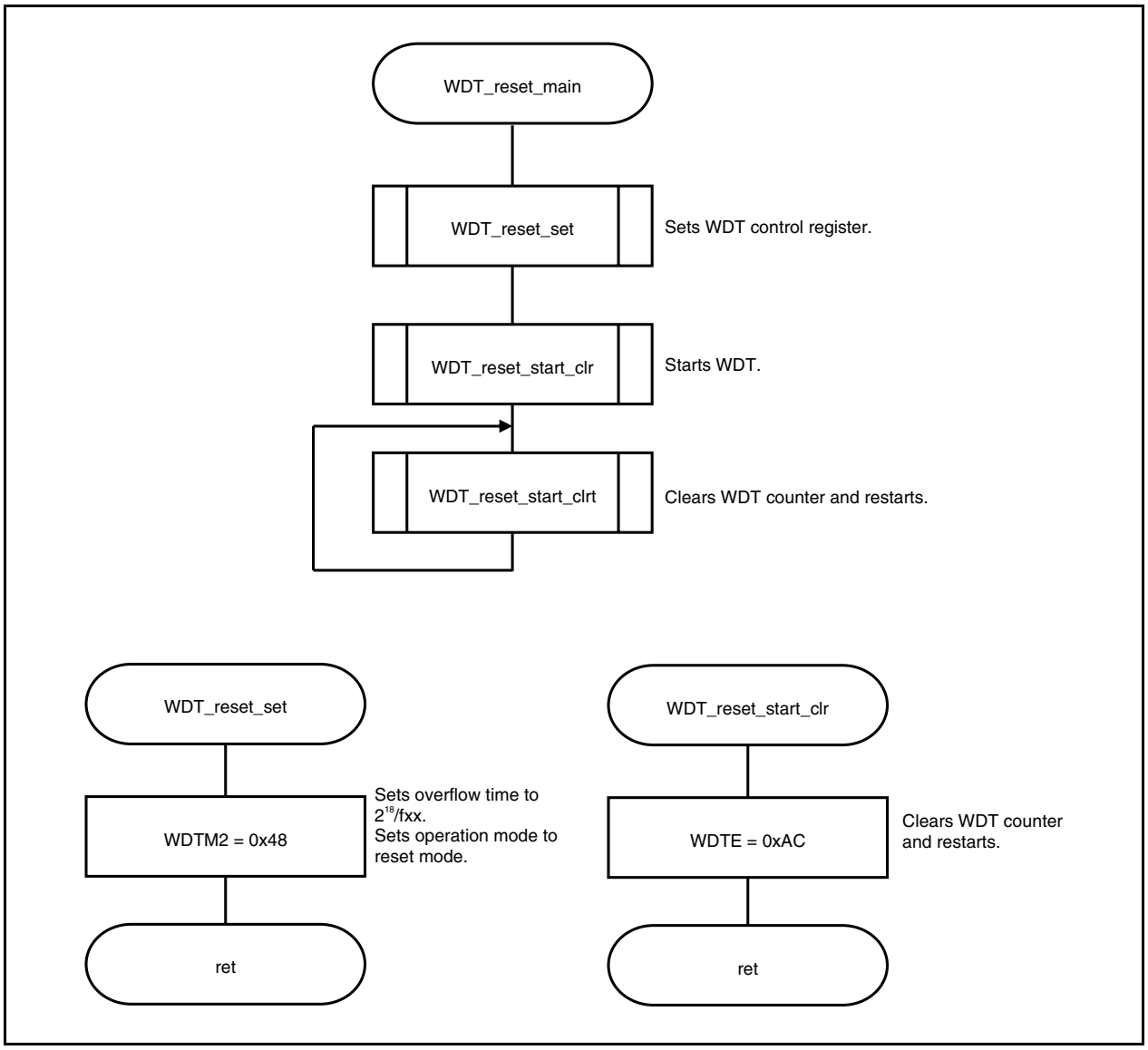
7.1 Reset Mode

[Function]	Starts automatically in reset mode after reset release. If the overflow time is set to 2^{18}/fxx and the ACH is not written to the WDTE register within the overflow detection time, the reset signal (WDT2RES) is generated.
[Function name]	WDT_reset_main
[Argument]	None
[Processing content]	If the ACH is not written to the WDTE register within the overflow time, the reset signal (WDT2RES) is generated.
[SFR used]	None
[call functions]	WDT_reset_set, WDT_reset_start_clr
[Variable]	None
[Interrupt]	None
[Interrupt source]	WDT2RES
[File name]	WDT_reset\WDT_reset.c
[Caution]	Writing to the WDTM2 register is available only by the byte access once after reset.

[Function name]	WDT_reset_set
[Argument]	None
[Processing content]	Sets the WDT control register.
[SFR used]	WDTM2: 0x48 (Sets overflow time to 2^{18}/fxx and sets operation mode to reset mode.)
[call function]	None
[Variable]	None
[File name]	WDT_reset\WDT_reset.c
[Caution]	None

[Function name]	WDT_reset_start_clr
[Argument]	None
[Processing content]	Starting function of WDT
[SFR used]	WDTE: 0xAC (Clears the WDT counter and restarts.)
[call function]	None
[Variable]	None
[File name]	WDT_reset\WDT_reset.c
[Caution]	None

Figure 7-1. Reset Mode (1)



7.2 Non-Maskable Interrupt Request Mode

[Function]	Starts automatically in reset mode after reset release. If the overflow time is set to 2^{18}/fxx and the ACH is not written to the WDTE register within the overflow detection time, the non-maskable interrupt request signal (INTWDT2) is generated.
[Function name]	WDT_nmi_main
[Argument]	None
[Processing content]	If the ACH is not written to the WDTE register within the overflow time, the reset signal (INTWDT2) is generated.
[SFR used]	None
[call functions]	WDT_nmi_set, WDT_nmi_start_clr
[Variable]	None
[Interrupt]	WDT_INTWDT2_int
[Interrupt source]	INTWDT2
[File name]	WDT_nmi\WDT_nmi.c
[Caution]	Writing to the WDTM2 register is available only by the byte access once after reset.

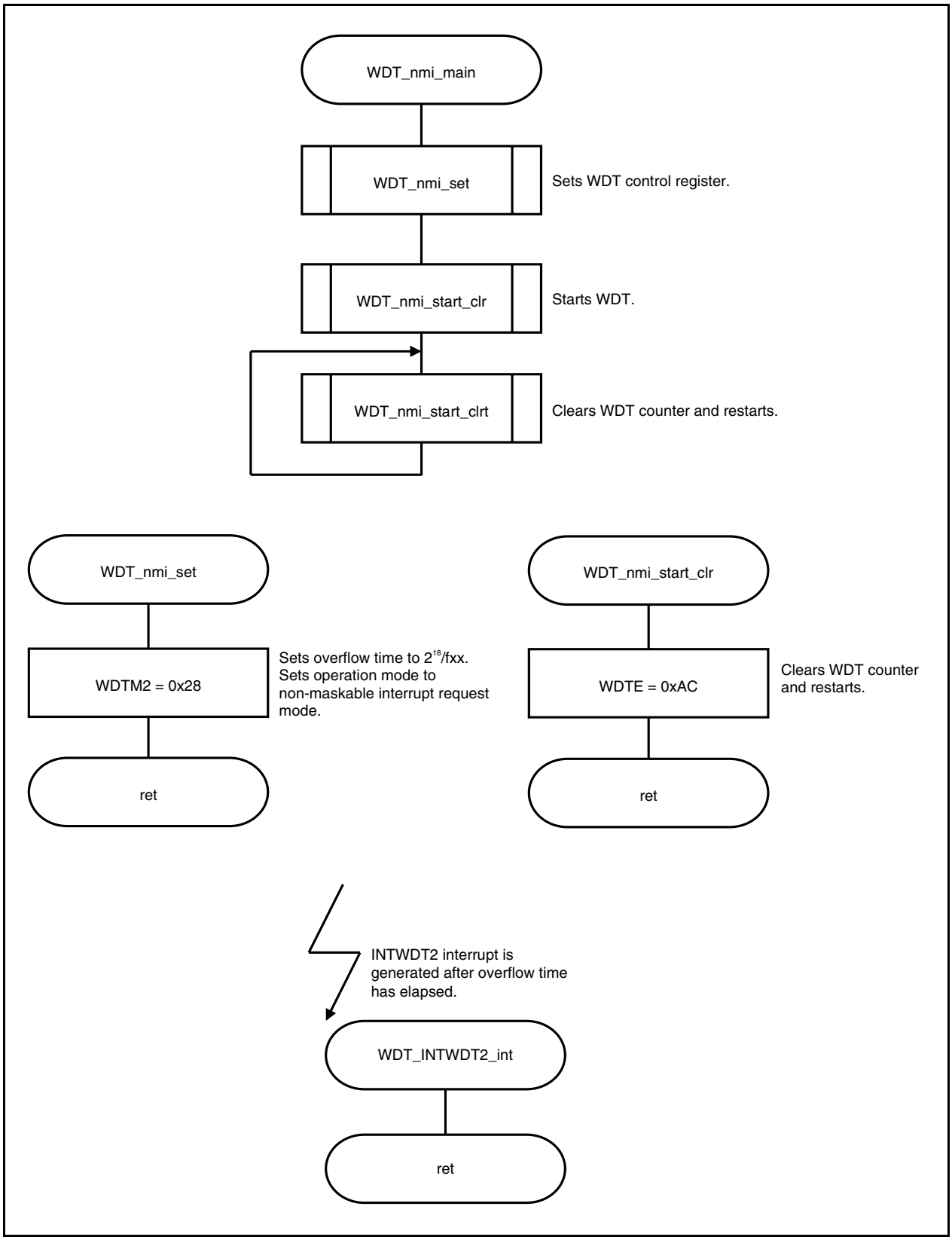
[Function name]	WDT_nmi_set
[Argument]	None
[Processing content]	Sets the WDT control register.
[SFR used]	WDTM2: 0x28 (Sets overflow time to 2^{18}/fxx and sets operation mode to reset mode.)
[call function]	None
[Variable]	None
[File name]	WDT_nmi\WDT_nmi.c
[Caution]	None

[Function name]	WDT_nmi_start_clr
[Argument]	None
[Processing content]	Starting function of WDT
[SFR used]	WDTE: 0xAC (Clears the WDT counter and restarts.)
[call function]	None
[Variable]	None
[File name]	WDT_nmi\WDT_nmi.c
[Caution]	None

Interrupt function

[Function name]	WDT_nmi_int
[Overview]	Defined by the user.
[Factor]	INTWDT2 Overflow time elapsed.
[call function]	None
[Variable]	None
[File name]	WDT_nmi\WDT_nmi.c
[Caution]	None

Figure 7-2. Non-Maskable Interrupt Request Mode



8.1 External Trigger Mode (Continuous Scan Mode)

[Function]	Performs A/D conversion by setting the A/D conversion operation start timing to external trigger mode of the hardware trigger mode.
[Function name]	ad_external_main
[Argument]	None
[Processing content]	When a trigger is input from the ADTRG pin, selects pins in the order of ANI0, ANI1, ANI2, and ANI3, continuously performs A/D conversion, and stores the A/D conversion results to buf[], buf_1[], buf_2[], and buf_3[]. Generates an A/D conversion interrupt request signal (INTAD) upon each completion of conversion operation of the specified analog input pin. Performs A/D conversion for 10 times.
[SFR used]	ADMK: 0 (Clears A/D conversion end interrupt request signal (INTAD), releases mask, sets to priority level 7.)
[call functions]	ad_trgger_port_set, ad_port_set, ad_set, ad_start, ad_stop
[Variables]	unsigned short int buf [ ]: Conversion data storing buffer unsigned short int buf_1 [ ]: Conversion data storing buffer unsigned short int buf _2[ ]: Conversion data storing buffer unsigned short int buf _3[ ] Conversion data storing buffer volatile unsigned char count: Conversion count variable
[Interrupt]	ad_int
[Interrupt source]	INTAD
[File name]	ad_external_trigger\ad_external_trigger.c
[Caution]	None

[Function name]	ad_trigger_port_set
[Processing content]	Sets port 0 to the ADTRG input pin.
[SFRs used]	PMC0: 0x08 (Sets to ADTRG input pin.) PFC0: 0x08 (Sets to ADTRG input pin.)
[call function]	None
[Variable]	None
[File name]	ad_external_trigger\ad_external_trigger.c

[Function name]	ad_port_set
[Processing content]	Sets the analog input pin.
[SFR used]	ADA0S: 0x03 (Sets analog input pins to ANI0 pin, ANI1 pin, ANI2 pin, and ANI3 pin.)
[call function]	None
[Variable]	None
[File name]	ad_external_trigger\ad_external_trigger.c

[Function name]	ad_set
[Processing content]	Sets the A/D conversion control register.
[SFRs used]	ADA0M0: 0x16 (Sets to continuous scan mode, hardware trigger mode, and sets valid edge to falling edge.) ADA0M2: 0x00 (Sets to external trigger mode) ADA0M1: 0x80 (Sets high-speed conversion mode and conversion clock number to 6.5 μs)
[call function]	None
[Variable]	None
[File name]	ad_external_trigger\ad_external_trigger.c

[Function name]	ad_start
[Processing content]	Enables A/D conversion operation.
[SFR used]	ADA0M0.ADA0CE: 1 (Enables A/D conversion operation.)
[call function]	None
[Variable]	None
[File name]	ad_external_trigger\ad_external_trigger.c

[Function name]	ad_stop
[Processing content]	Stops A/D conversion operation.
[SFR used]	ADA0M0.ADA0CE: 0 (Stops A/D conversion operation.)
[call function]	None
[Variable]	None
[File name]	ad_external_trigger\ad_external_trigger.c

Interrupt function

[Function name]	ad_int
[Processing content]	Stores A/D conversion result data to buffer.
[SFRs used]	ADA0CR0: A/D conversion result register 0 ADA0CR1: A/D conversion result register 1 ADA0CR2: A/D conversion result register 2 ADA0CR3: A/D conversion result register 3
[call function]	None
[Variables]	unsigned short int buf [ ]: Conversion data storing buffer unsigned short int buf _1[ ]: Conversion data storing buffer unsigned short int buf _2[ ]: Conversion data storing buffer unsigned short int buf _3[ ]: Conversion data storing buffer volatile unsigned char count: Conversion count variable
[File name]	ad_external_trigger\ad_external_trigger.c

Figure 8-1. External Trigger Mode (Continuous Scan Mode) (1)

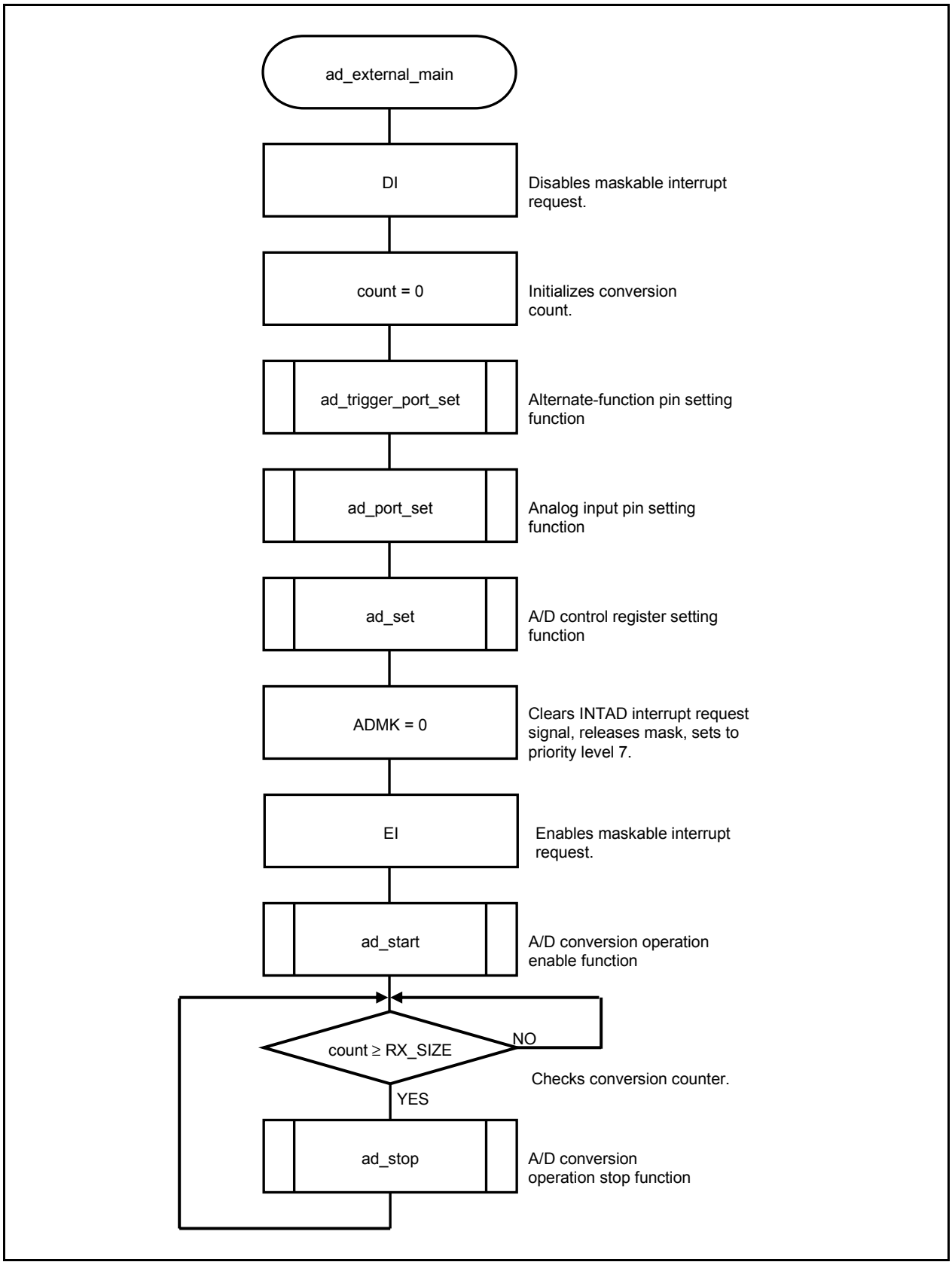


Figure 8-2. External Trigger Mode (Continuous Scan Mode) (2)

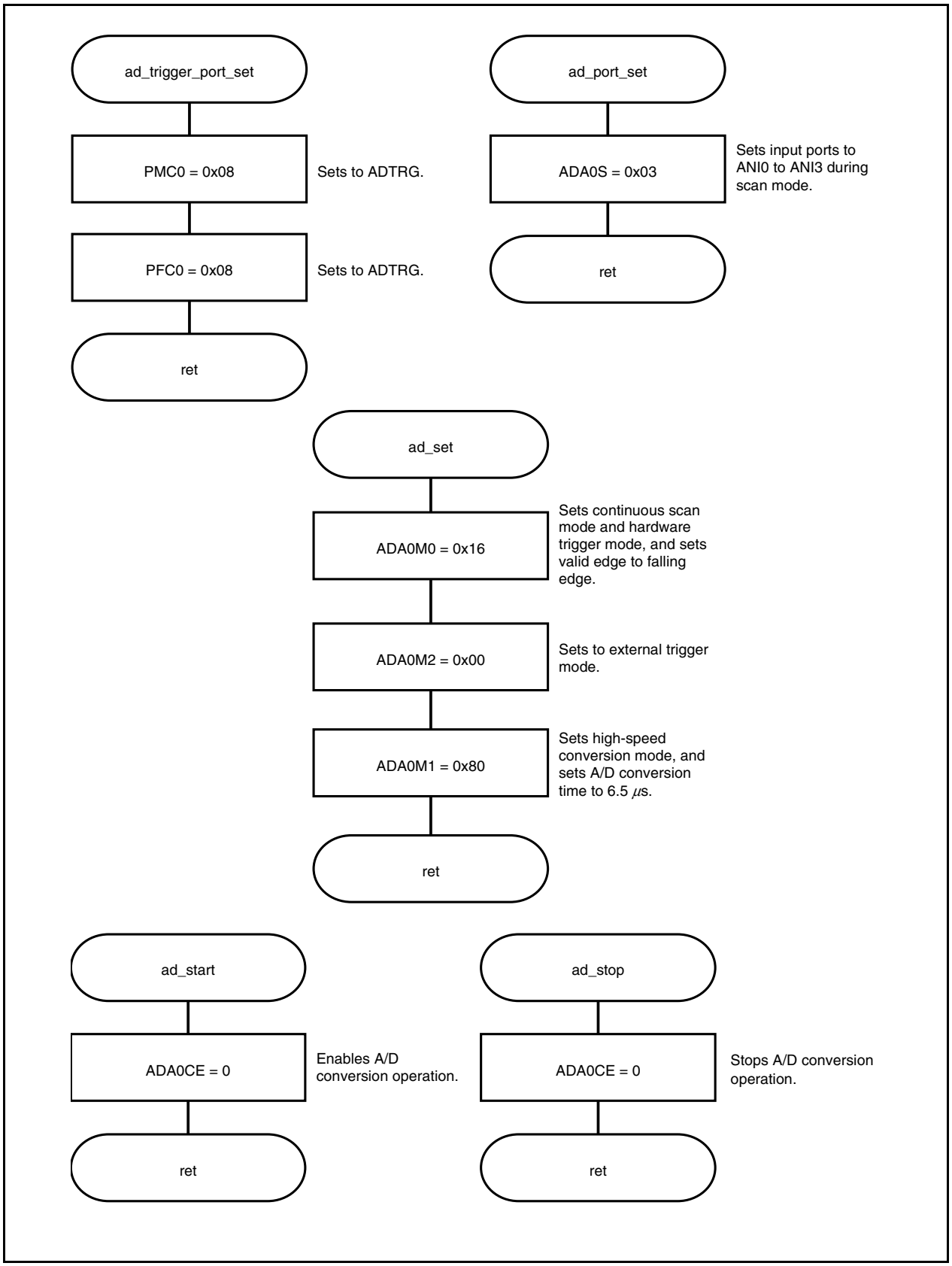
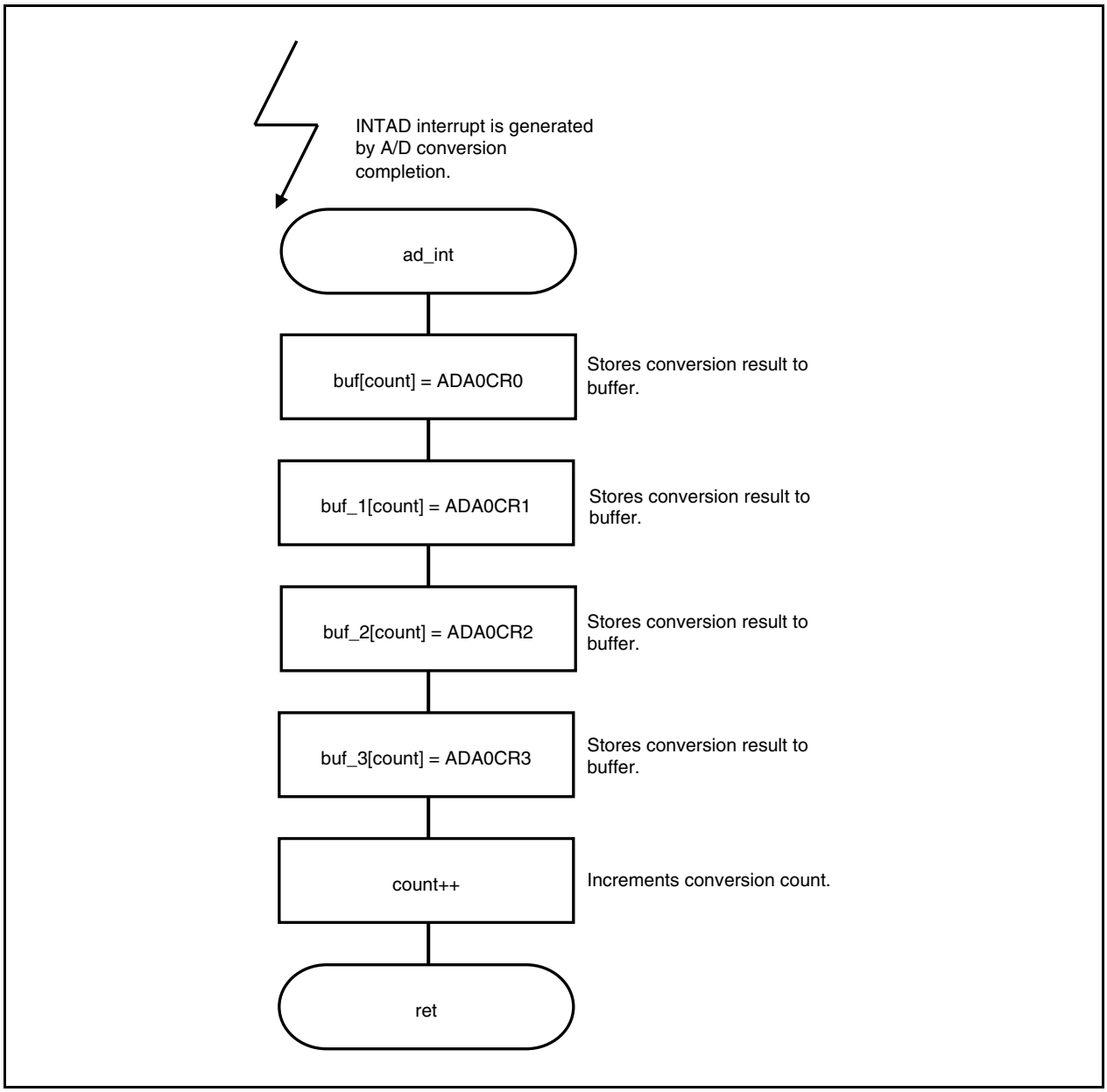


Figure 8-3. External Trigger Mode (Continuous Scan Mode) (3)



8.2 Software Trigger Mode (Continuous Select Mode)

[Function]	Performs A/D conversion by setting the A/D conversion operation start timing to software trigger mode.
[Function name]	ad_software_main
[Argument]	None
[Processing content]	Starts A/D conversion by setting the ADA0M0.ADA0CE bit to 1. Stores the A/D conversion result to buf[] by A/D converting the ANI0 pin input. An A/D conversion end interrupt request signal (INTAD) occurs upon completion of every A/D conversion. Performs A/D conversion for 10 times.
[SFR used]	ADMK: 0 (Clears A/D conversion end interrupt request signal (INTAD), releases mask, sets to priority level 7.)
[call functions]	ad_port_set, ad_set, ad_start, ad_stop
[Variables]	unsigned short int buf [ ]: Conversion data storing buffer volatile unsigned char count: Conversion count variable
[Interrupt]	ad_int
[Interrupt source]	INTAD
[File name]	ad_software_trigger\ad_software_trigger.c
[Caution]	None

[Function name]	ad_port_set
[Processing content]	Sets the analog input pin.
[SFR used]	ADA0S: 0x00 (Sets analog input pin to ANI00.)
[call function]	None
[Variable]	None
[File name]	ad_software_trigger\ad_software_trigger.c

[Function name]	ad_set
[Processing content]	Sets the A/D conversion control register.
[SFRs used]	ADA0M0: 0x00 (Sets to continuous select mode, software trigger mode, and sets valid edge to “No edge detection”.) ADA0M1: 0x00 (Normal conversion mode, sets conversion clock number to 16.25 μs.)
[call function]	None
[Variable]	None
[File name]	ad_software_trigger\ad_software_trigger.c

[Function name]	ad_start
[Processing content]	Starts A/D conversion operation.
[SFR used]	ADA0M0.ADA0CE: 1 (Enables A/D conversion operation.)
[call function]	None
[Variable]	None
[File name]	ad_software_trigger\ad_software_trigger.c

[Function name]	ad_stop
[Processing content]	Stops A/D conversion operation.
[SFR used]	ADA0M0.ADA0CE: 0 (Stops A/D conversion operation.)
[call function]	None
[Variable]	None
[File name]	ad_software_trigger\ad_software_trigger.c

Interrupt function

[Function name]	ad_int
[Processing content]	Stores A/D conversion result data to buffer.
[SFR used]	ADA0CR0: A/D conversion result register 0
[call function]	None
[Variables]	unsigned short int buf [ ]: Conversion data storing buffer volatile unsigned char count: Conversion count variable
[File name]	ad_software_trigger\ad_software_trigger.c

Figure 8-4. Software Trigger Mode (Continuous Select Mode) (1)

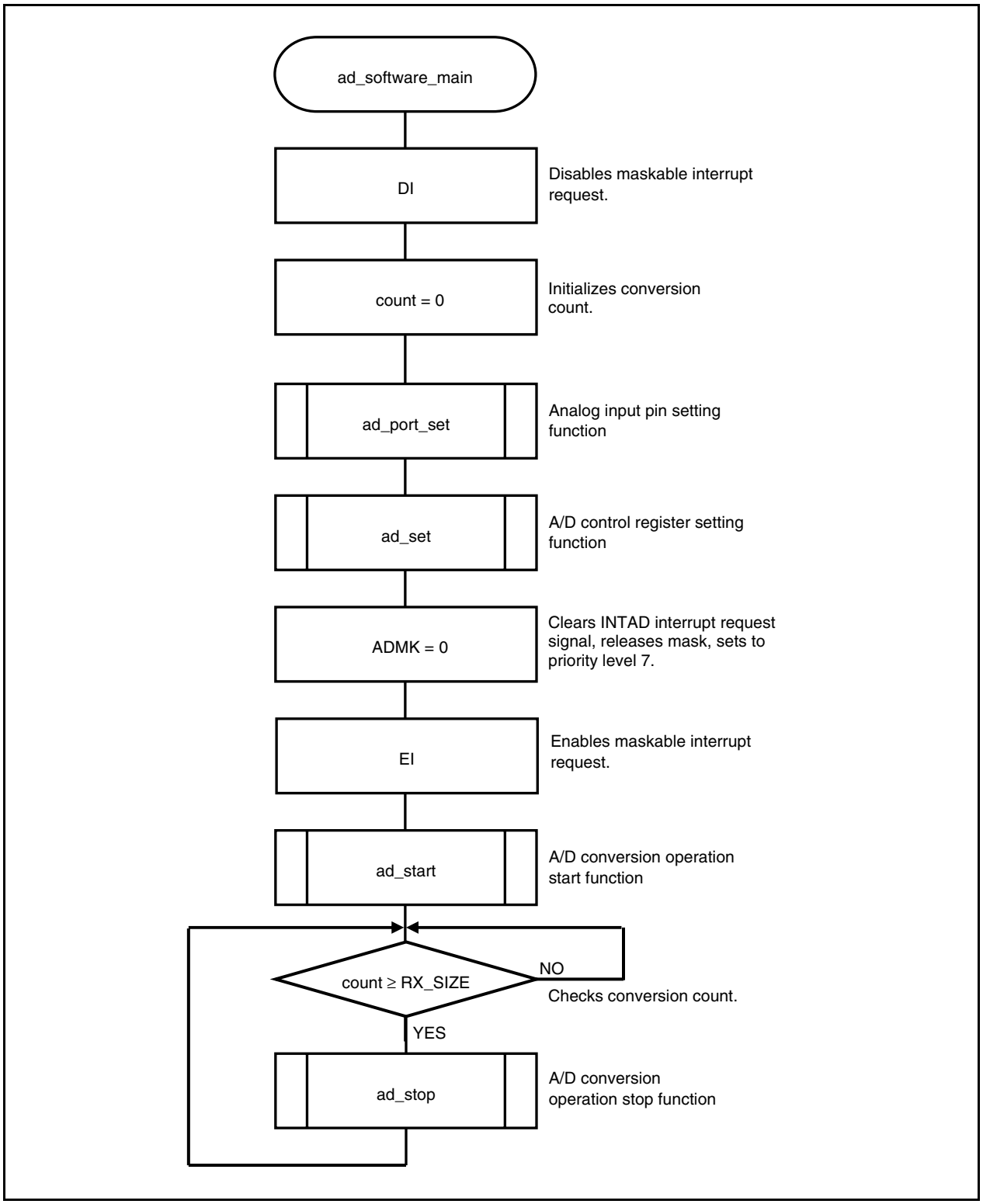
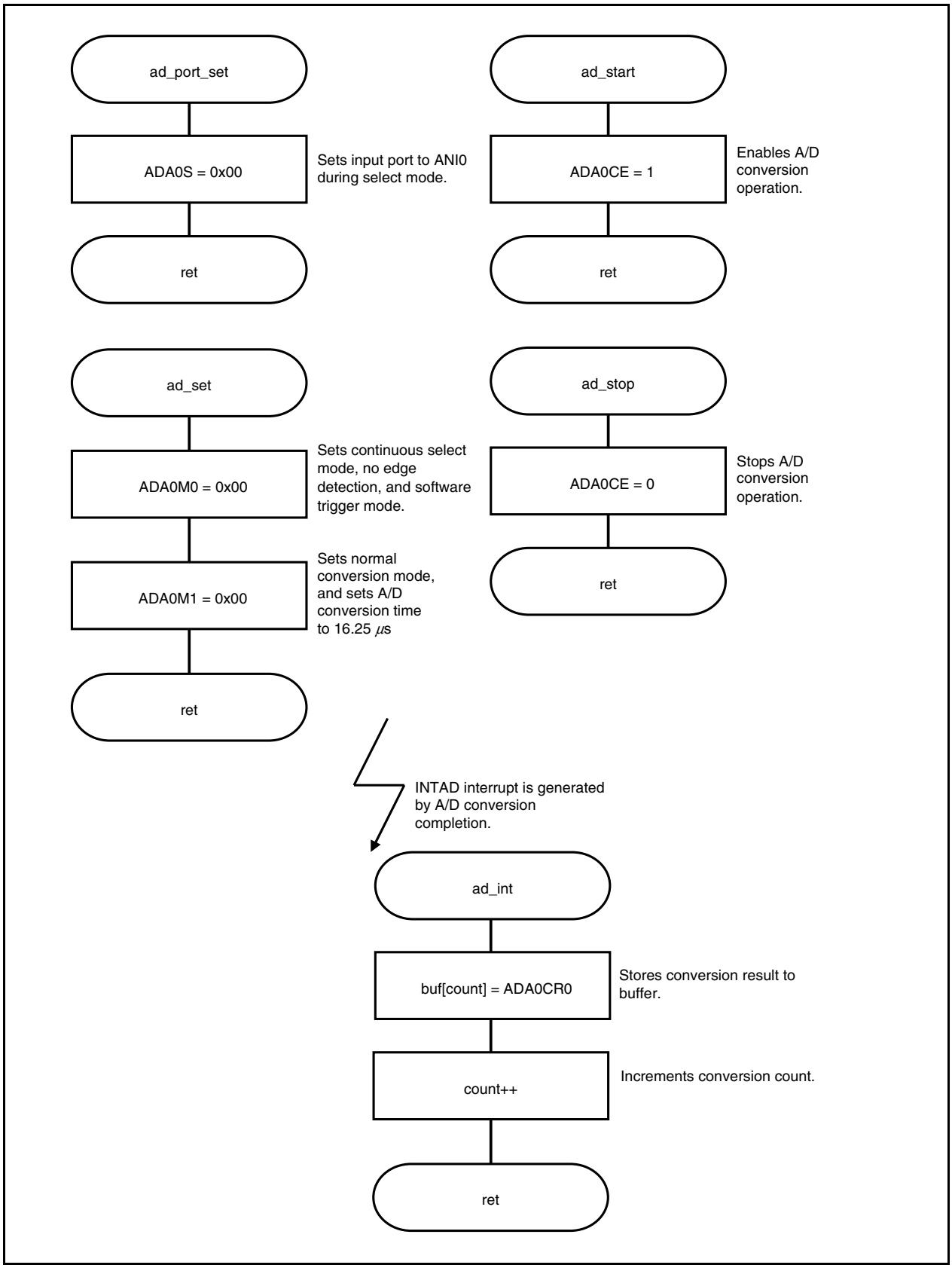


Figure 8-5. Software Trigger Mode (Continuous Select Mode) (2)



8.3 Software Trigger Mode (One-Shot Scan Mode)

[Function]	Performs A/D conversion by setting the A/D conversion operation start timing to software trigger mode.
[Function name]	ad_software1_main
[Argument]	None
[Processing content]	Starts A/D conversion by setting the ADA0M0.ADA0CE bit to 1. When a trigger is input from the ADTRG pin, selects pins in the order of ANI0, ANI1, ANI2, and ANI3, continuously performs A/D conversion, and stores the A/D conversion results to buf[], buf_1[], buf_2[], and buf_3[]. Generators an A/D conversion end interrupt request signal (INTAD) upon each completion of A/D conversion. Performs A/D conversion for 10 times.
[SFR used]	ADMK: 0 (Clears A/D conversion end interrupt request signal (INTAD), releases mask, sets to priority level 7.)
[call functions]	ad_port_set, ad_set, ad_start, ad_stop
[Variables]	unsigned short int buf [ ]: Conversion data storing buffer unsigned short int buf_1[ ]: Conversion data storing buffer unsigned short int buf_2[ ]: Conversion data storing buffer unsigned short int buf_3[ ]: Conversion data storing buffer volatile unsigned char count: Conversion count variable
[Interrupt]	ad_int
[Interrupt source]	INTAD
[File name]	ad_software1_trigger\ad_software1_trigger.c
[Caution]	None

[Function name]	ad_port_set
[Processing content]	Sets the analog input pin.
[SFR used]	ADA0S: 0x03 (Sets analog input pin to ANI0 pin, ANI1 pin, ANI2 pin, and ANI3 pin.)
[call function]	None
[Variable]	None
[File name]	ad_software1_trigger\ad_software1_trigger.c

[Function name]	ad_set
[Processing content]	Sets the A/D conversion control register.
[SFRs used]	ADA0M0: 0x30 (Sets to one-shot scan mode, software trigger mode, and sets valid edge to “No edge detection”.) ADA0M1: 0x00 (Normal conversion mode, sets conversion clock number to 16.25 μs.)
[call function]	None
[Variable]	None
[File name]	ad_software1_trigger\ad_software1_trigger.c

[Function name]	ad_start
[Processing content]	Starts A/D conversion operation.
[SFR used]	ADA0M0.ADA0CE: 1 (Enables A/D conversion operation.)
[call function]	None
[Variable]	None
[File name]	ad_software1_trigger\ad_software1_trigger.c

[Function name]	ad_stop
[Processing content]	Stops A/D conversion operation
[SFR used]	ADA0M0.ADA0CE: 0 (Stops A/D conversion operation.)
[call function]	None
[Variable]	None
[File name]	ad_software1_trigger\ad_software1_trigger.c

Interrupt function

[Function name]	ad_int	
[Processing content]	Stores A/D conversion result data to buffer.	
[SFRs used]	ADA0CR0	A/D conversion result register 0
	ADA0CR1	A/D conversion result register 1
	ADA0CR2	A/D conversion result register 2
	ADA0CR3	A/D conversion result register 3
[call function]	None	
[Variables]	unsigned short int buf [ ]:	Conversion data storing buffer
	unsigned short int buf_1[ ]:	Conversion data storing buffer
	unsigned short int buf_2[ ]:	Conversion data storing buffer
	unsigned short int buf_3[ ]:	Conversion data storing buffer
	volatile unsigned char count:	Conversion count variable
[File name]	ad_software1_trigger\ad_software1_trigger.c	

Figure 8-6. Software Trigger Mode (One-Shot Scan Mode) (1)

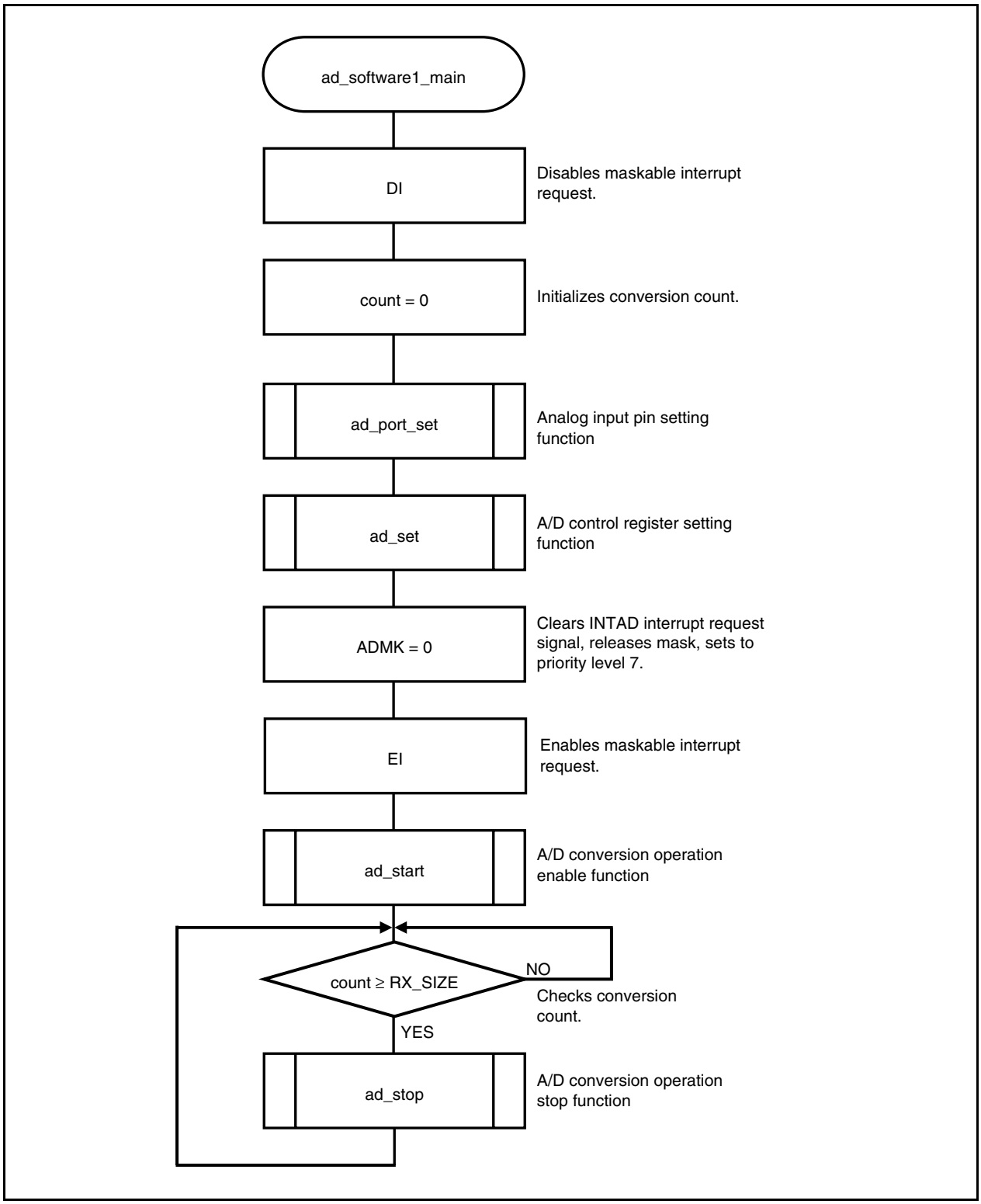


Figure 8-7. Software Trigger Mode (One-Shot Scan Mode) (2)

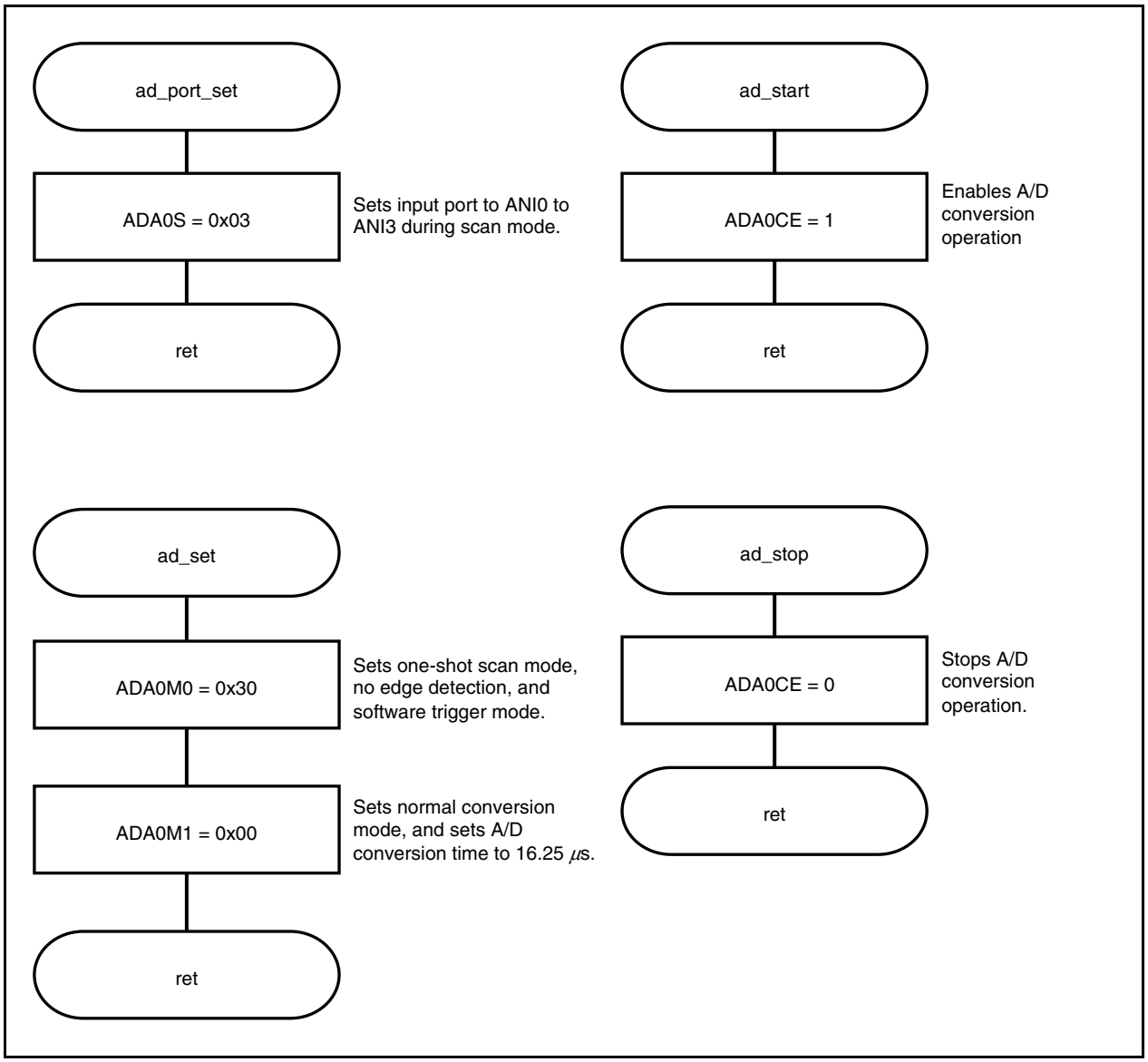
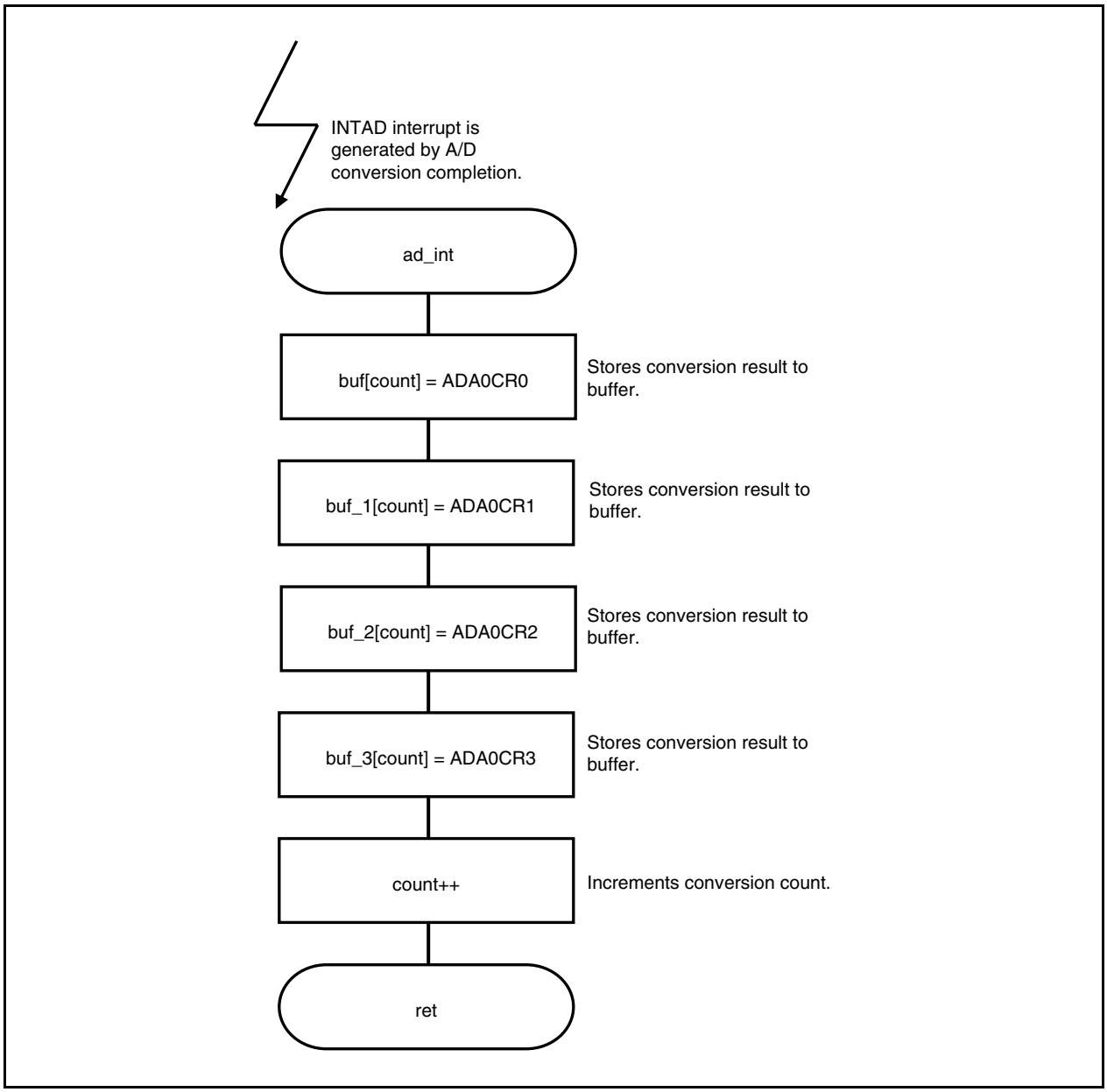


Figure 8-8. Software Trigger Mode (One-Shot Scan Mode) (3)



8.4 Timer Trigger Mode (One-Shot Select Mode)

[Function]	Performs A/D conversion by setting the A/D conversion operation start timing to timer trigger mode 0 of the hardware trigger mode.
[Function name]	ad_timer_main
[Argument]	None
[Processing content]	Starts A/D conversion when the compare match interrupt request signal (INTTP2CC0) is input. Stores the A/D conversion result to buf[] by A/D converting the ANI00 pin. An A/D conversion end interrupt request signal (INTAD) occurs upon completion of every A/D conversion. Performs A/D conversion for 10 times.
[SFR used]	ADMK: 0 (Clears A/D conversion end interrupt request signal (INTAD), releases mask, sets to priority level 7.)
[call functions]	Ad_port_set, ad_set, ad_start, ad_timer_trigger, ad_stop
[Variables]	unsigned short int buf [ ]: Conversion data storing buffer volatile unsigned char count: Conversion count variable
[Interrupt]	ad_int
[Interrupt source]	INTAD
[File name]	ad_timer_trigger\ad_timer_trigger.c
[Caution]	For generating the timer compare match interrupt request signal (INTTP2CC0), refer to **CHAPTER 3 16-BIT TIMER/EVENT COUNTER P (TMP)**.

[Function name]	ad_port_set
[Processing content]	Sets the analog input pin.
[SFR used]	ADA0S: 0x00 (Sets analog input pin to ANI0 pin.)
[call function]	None
[Variable]	None
[File name]	ad_timer_trigger\ad_timer_trigger.c

[Function name]	ad_set
[Processing content]	Sets the A/D conversion control register.
[SFRs used]	ADA0M0: 0x22 (Sets to one-shot select mode, hardware trigger mode, and sets valid edge to “No edge detection”.) ADA0M2: 0x01 (Sets timer trigger mode 0.) ADA0M1: 0x80 (Sets high-speed conversion mode and sets conversion clock number to 6.5 μs.)
[call function]	None
[Variable]	None
[File name]	ad_timer_trigger\ad_timer_trigger.c

[Function name]	ad_start
[Processing content]	Enables A/D conversion operation.
[SFR used]	ADA0M0.ADA0CE: 1 (Enables A/D conversion operation.)
[call function]	None
[Variable]	None
[File name]	ad_timer_trigger\ad_timer_trigger.c

[Function name]	ad_stop
[Processing content]	Stops A/D conversion operation.
[SFR used]	ADA0M0.ADA0CE: 0 (Stops A/D conversion operation)
[call function]	None
[Variable]	None
[File name]	ad_timer_trigger\ad_timer_trigger.c

Interrupt function

[Function name]	ad_int
[Processing content]	Stores A/D conversion result data to buffer.
[SFR used]	ADA0CR0: A/D conversion result register 0
[call function]	None
[Variables]	unsigned short int buf [ ]: Conversion data storing buffer volatile unsigned char count: Conversion count variable
[File name]	ad_timer_trigger\ad_timer_trigger.c

Figure 8-9. Timer Trigger Mode (One-Shot Select Mode) (1)

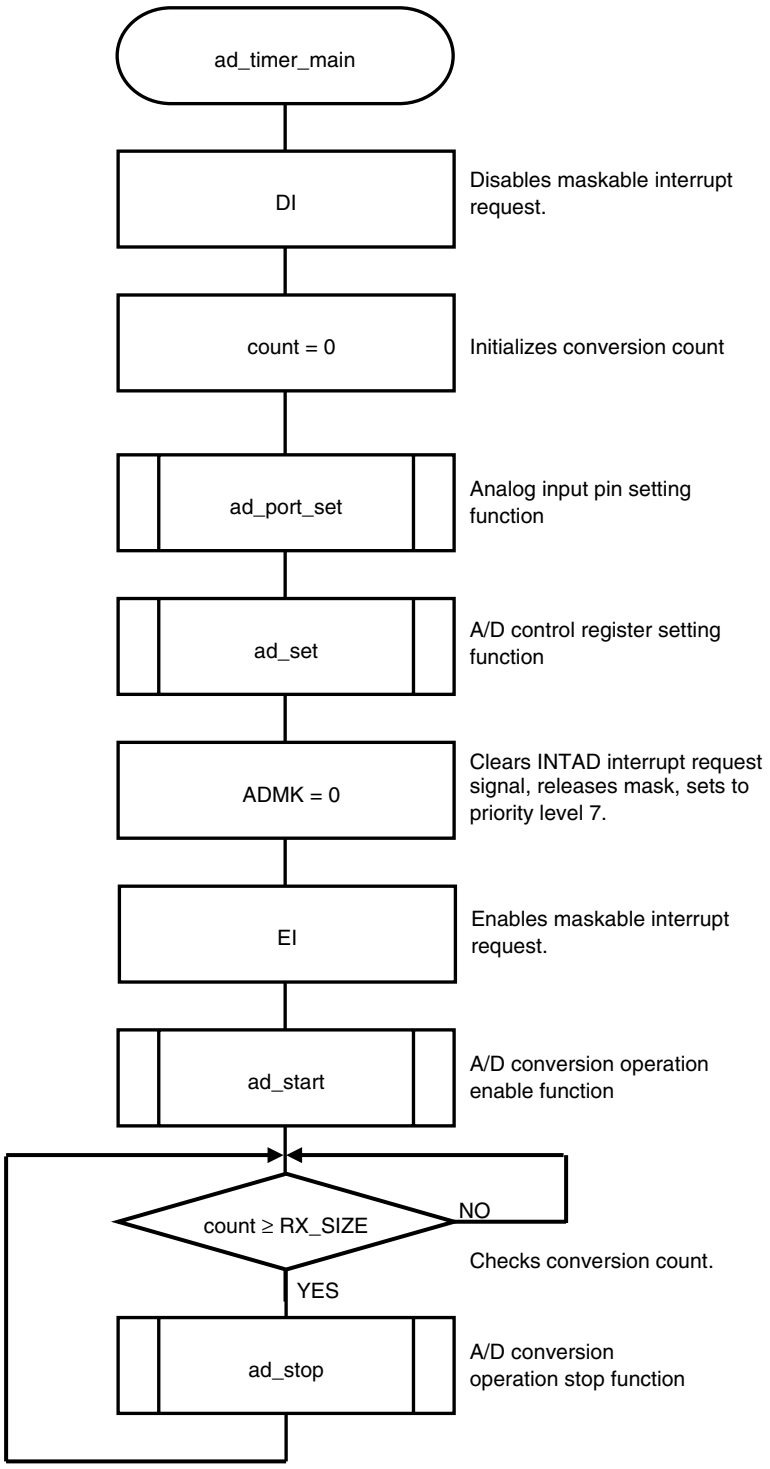
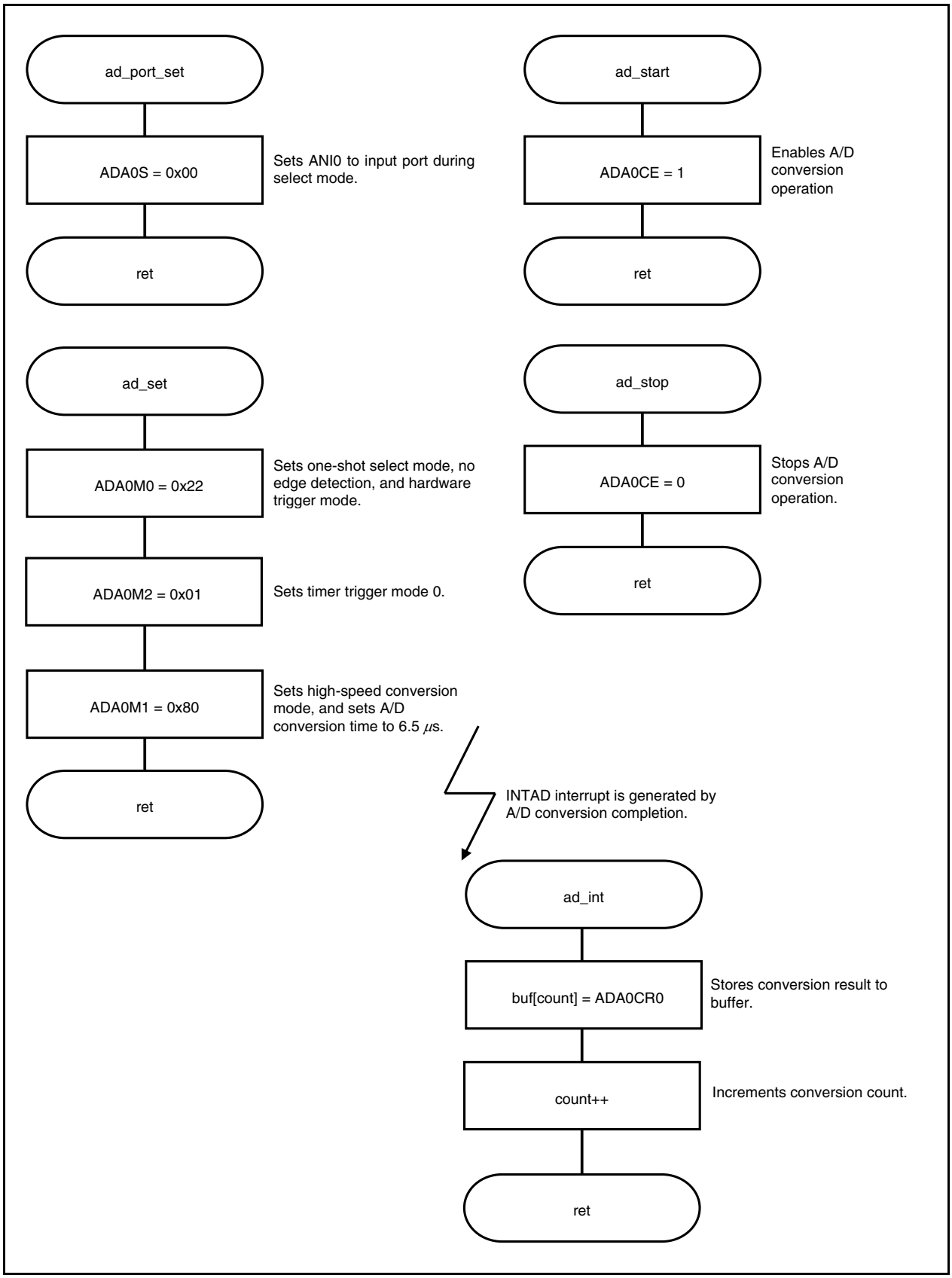


Figure 8-10. Timer Trigger Mode (One-Shot Select Mode) (2)



CHAPTER 9 D/A CONVERTER

9.1 Normal Mode

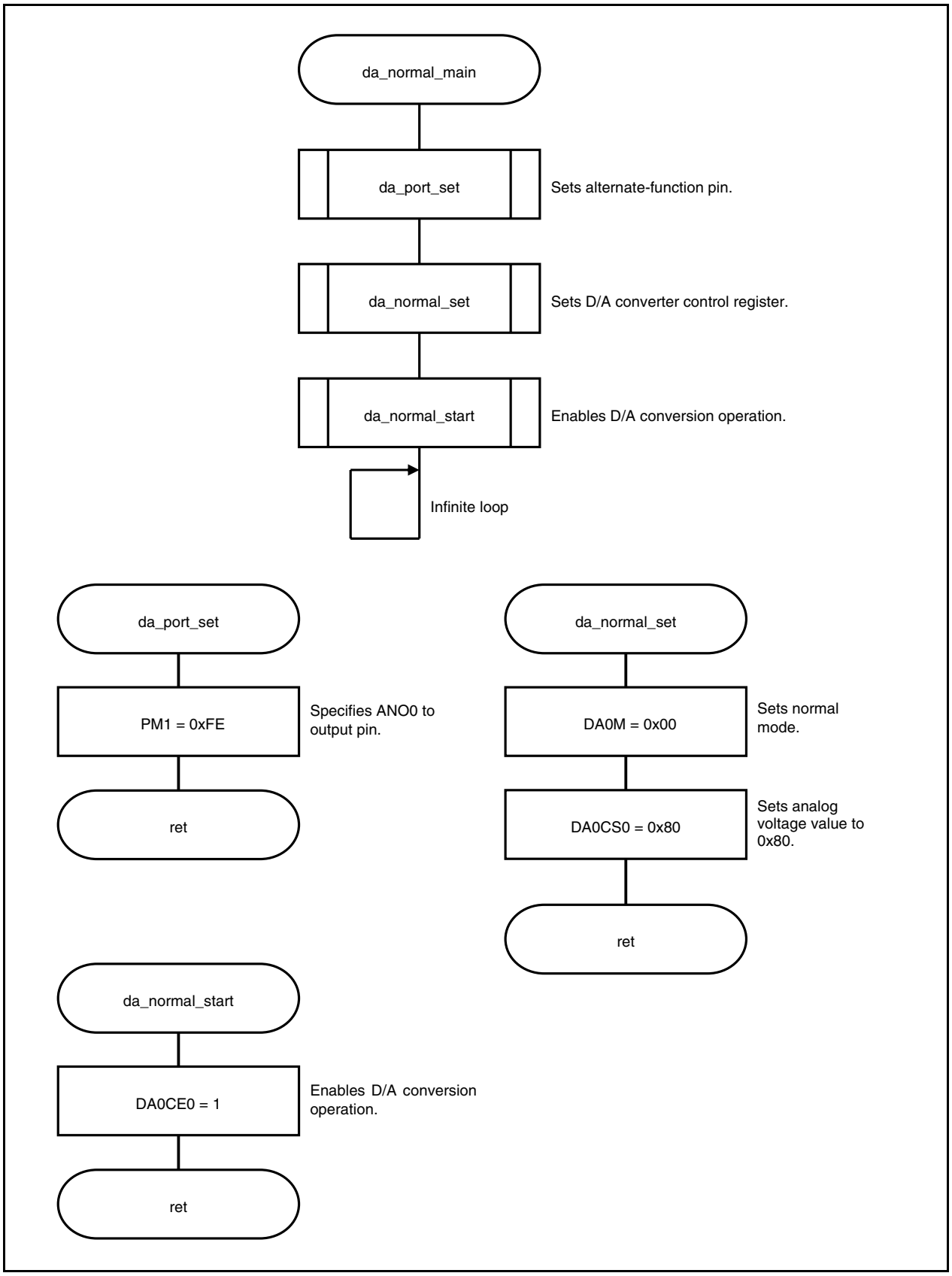
[Function]	Performs D/A conversion, using a write operation to the DA0CS0 register as a starting trigger.
[Function name]	da_normal_main
[Argument]	None
[Processing content] and	Performs D/A conversion, using a write operation to the DA0CS0 register as a trigger outputs to the ANO0 pin the value set to the DA0CS0 register.
[SFR used]	None
[call functions]	da_port_set, da_normal_set, da_normal_start,
[Variable]	None
[Interrupt]	None
[Interrupt source]	None
[File name]	da_normal\da_normal.c
[Caution]	None

[Function name]	da_port_set
[Processing content]	Sets port 1 to output pin.
[SFR used]	PM1: 0xFE (Specify ANO0 to output pin)
[call function]	None
[Variable]	None
[File name]	da_normal\da_normal.c

[Function name]	da_normal_set
[Processing content]	Sets analog voltage value to be output to the ANO0 pin.
[SFRs used]	DA0CS0: 0x80 (Sets analog voltage value to be output to the ANO0 pin to 0x80.) DA0M: 0x00 (Sets normal mode.)
[call function]	None
[Variable]	None
[File name]	da_normal\da_normal.c

[Function name]	da_normal_start
[Processing content]	Starts D/A conversion operation.
[SFR used]	DA0CE0: 1(Enables D/A conversion operation.)
[call function]	None
[Variable]	None
[File name]	da_normal\da_normal.c

Figure 9-1. Normal Mode



9.2 Real-Time Output Mode

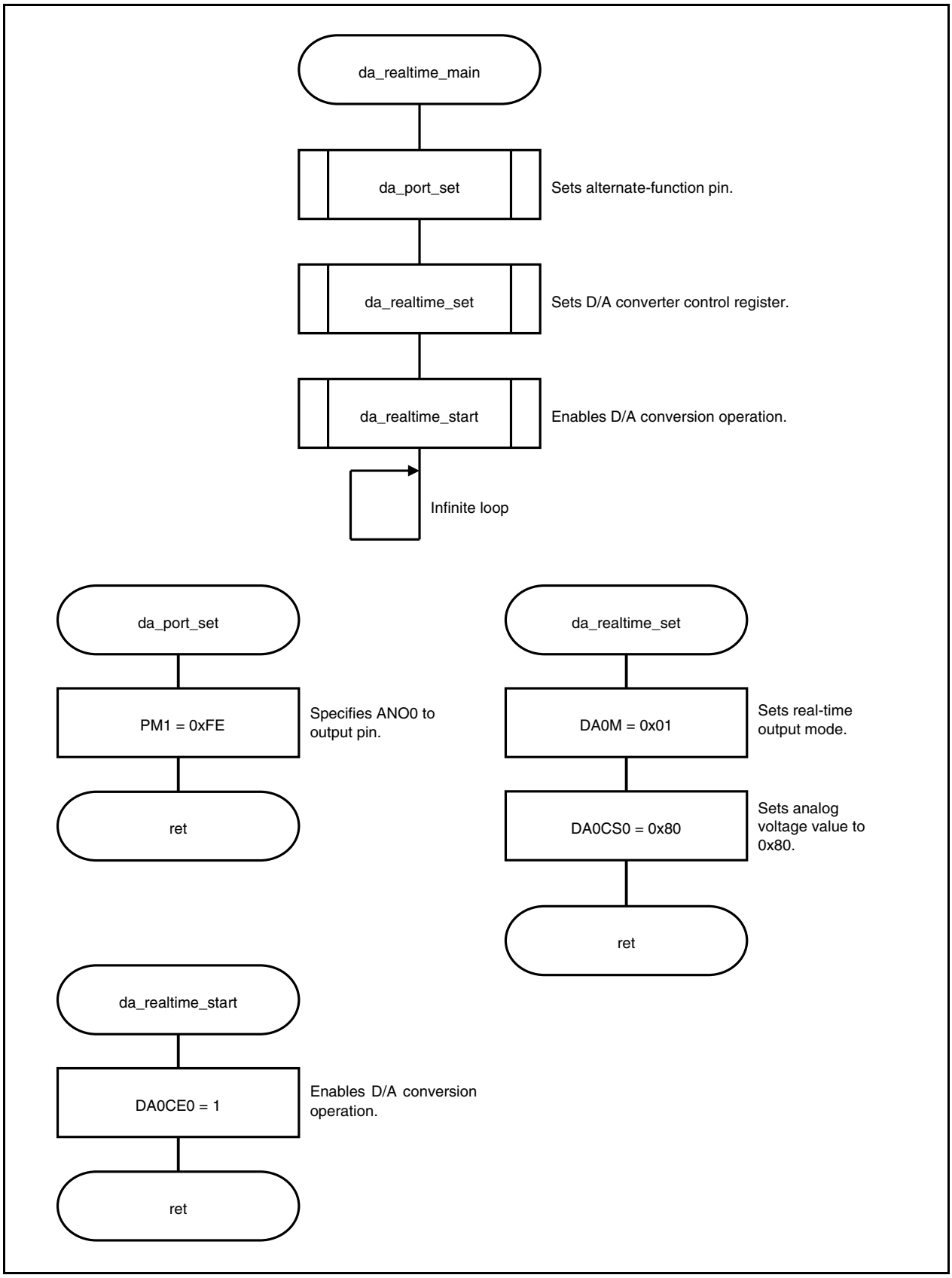
[Function]	Performs D/A conversion, using the TMP2 interrupt request signal (INTTP2CC0) as a starting trigger.
[Function name]	da_realtime_main
[Argument]	None
[Processing content] trigger and	Performs D/A conversion, using the TMP2 interrupt request signal (INTTP2CC0) as a outputs the value set to the DA0CS0 register to the ANO0 pin.
[SFR used]	None
[call functions]	da_port_set, da_realtime_set, da_realtime_start,
[Variable]	None
[Interrupt]	None
[Interrupt source]	None
[File name]	da_realtime\da_realtime.c
[Caution]	For generating the timer compare match interrupt request signal (INTTP2CC0), refer to **CHAPTER 3 16-BIT TIMER/EVENT COUNTER P (TMP)**.

[Function name]	da_port_set
[Processing content]	Sets port 1 to output pin.
[SFR used]	PM1: 0xFE (Specifies ANO0 to output pin.)
[call function]	None
[Variable]	None
[File name]	da_realtime\da_realtime.c

[Function name]	da_realtime_set
[Processing content]	Sets analog voltage value to be output to the ANO0 pin.
[SFRs used]	DA0CS0: 0x80 (Sets analog voltage value to be output to the ANO0 pin to 0x80.) DA0M: 0x00 (Sets normal mode.)
[call function]	None
[Variable]	None
[File name]	da_realtime\da_realtime.c

[Function name]	da_realtime_start
[Processing content]	Starts D/A conversion operation.
[SFR used]	DA0CE0: 1(Enables D/A conversion operation.)
[call function]	None
[Variable]	None
[File name]	da_realtime\da_realtime.c

Figure 9-2. Real-Time Output Mode



CHAPTER 10 ASYNCHRONOUS SERIAL INTERFACE A (UARTA)

10.1 Asynchronous Serial Interface A (UARTAn) (n = 0 to 2)

[Function]	Performs continuous UARTA0 transmission/reception.
[Function name]	uarta_main
[Argument]	None
[Processing content]	Performs transmission/reception for ten times using UARTA0. Stores received data in buf_rx[], and transmitted data in buf_tx[].
[SFRs used]	UA0TMK: 0 (Clears UARTA0 transmission enable interrupt request signal (INTUA0T), releases mask, sets to priority level 7.) UA0RMK: 0 (Clears UARTA0 reception end interrupt request signal (INTUA0R), releases mask, sets to priority level 7.)
[call functions]	uarta_port_set, uarta_set, uarta_start, uarta_receive_end, uarta_send_end, uarta_end
[Variables]	unsigned char buf_tx[ ]: Transmit data storing buffer unsigned char buf_rx[ ]: Receive data storing buffer volatile unsigned char count_tx: Transmission count variable volatile unsigned char count_rx: Reception count variable unsigned char count: Transfer data generating variable
[Interrupts]	uarta_int_send, uarta_int_receive
[Interrupt sources]	INTUA0T, INTUA0R
[File name]	UARTA\uarta.c
[Caution]	None

[Function name]	uarta_port_set
[Processing content]	Sets alternate-function pin to UARTA0 I/O pin.
[SFR used]	PMC3: 0x03 (Sets TXDA0 output and RXDA0 input.)
[call function]	None
[Variable]	None
[File name]	UARTA\uarta.c
[Caution]	None

[Function name]	uarta_set
[Processing content]	Sets the UARTA0 control register. Sets baud rate to 9,600 bps.
[SFRs used]	UA0CTL1: 0x01 (Sets baud rate to 9,600 bps.) UA0CTL2: 0x82 (Sets baud rate to 9,600 bps.) UA0OPT0: 0x14 (Sets to transfer data normal output and transfer data normal input.) UA0CTL0: 0x9A (Enables UARTA0 operation, sets to LSB first, and sets odd parity to output data, character length of 8 bits, and stop bit length of 1 bit.)
[call function]	None
[Variable]	None
[File name]	UARTA\uarta.c
[Caution]	None

[Function name]	uarta_start
[Processing content]	Enables transmission/reception and writes data to the UA0TX register.
[SFRs used]	UA0CTL0.UA0TXE: 1 (Enables transmission operation.) UA0CTL0.UA0RXE: 1 (Enables reception operation.) UA0TX Transmit data register
[call function]	None
[Variable]	unsigned char buf_tx[ ]: Transmit data storing buffer
[File name]	UARTA\uarta.
[Caution]	Set UA0RXE and UA0TXE bits to 1 after setting UA0CTL0.UA0PWR bit to 1.

[Function name]	uarta_receive_end
[Processing content]	Disables reception operation.
[SFR used]	UA0CTL0.UA0RXE: 0 (Disables reception operation.)
[call function]	None
[Variable]	None
[File name]	UARTA\uarta.c
[Caution]	None

[Function name]	uarta_send_end
[Processing content]	Disables transmission operation.
[SFR used]	UA0CTL0.UA0TXE: 0 (Disables transmission operation.)
[call function]	None
[Variable]	None
[File name]	UARTA\uarta.c
[Caution]	None

[Function name]	uarta_end
[Processing content]	Disables operation of UARTA0.
[SFR used]	UA0CTL0.UA0PWR: 0 (Disables operation of UARTA0.)
[call function]	None
[Variable]	None
[File name]	UARTA\uarta.c
[Caution]	When stopping the operation, set the UA0RXE and UA0TXE bits to 0 after setting the UA0CTL.UA0PWR bit to 0.

Interrupt functions

[Function name]	uarta_int_send	
[Processing content]	Writes transmit data to the transmit data register. Stops transmission operation if number of transmissions and number of counts match.	
[SFR used]	UA0TX Transmit data register	
[call function]	None	
[Variables]	unsigned char buf_tx[ ]: volatile unsigned char count_tx:	Transmit data storing buffer Transmission count variable
[File name]	UARTA\uarta.c	
[Caution]	None	

[Function name]	uarta_int_recive	
[Processing content]	Stores the receive data of the receive data register to buffer. Stops reception operation if number of receptions and number of counts match.	
[SFR used]	UA0RX Receive data register	
[call function]	None	
[Variable]	unsigned char buf_rx[]: volatile unsigned char count_rx:	Receive data storing buffer Reception count variable
[File name]	UARTA\uarta.c	
[Caution]	None	

Figure 10-1. Asynchronous Serial Interface A (UARTAn) (1)

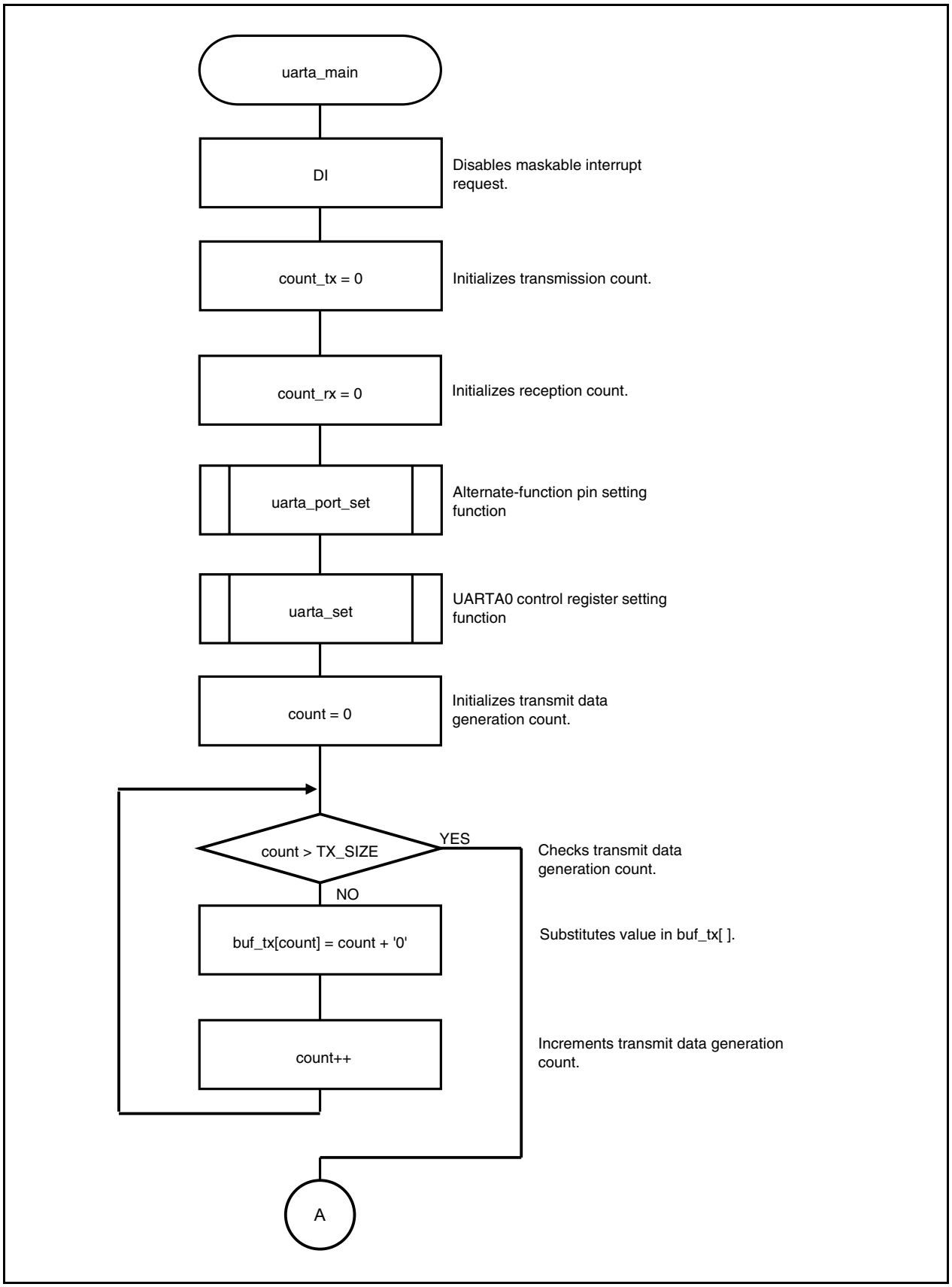


Figure 10-2. Asynchronous Serial Interface A (UARTAn) (2)

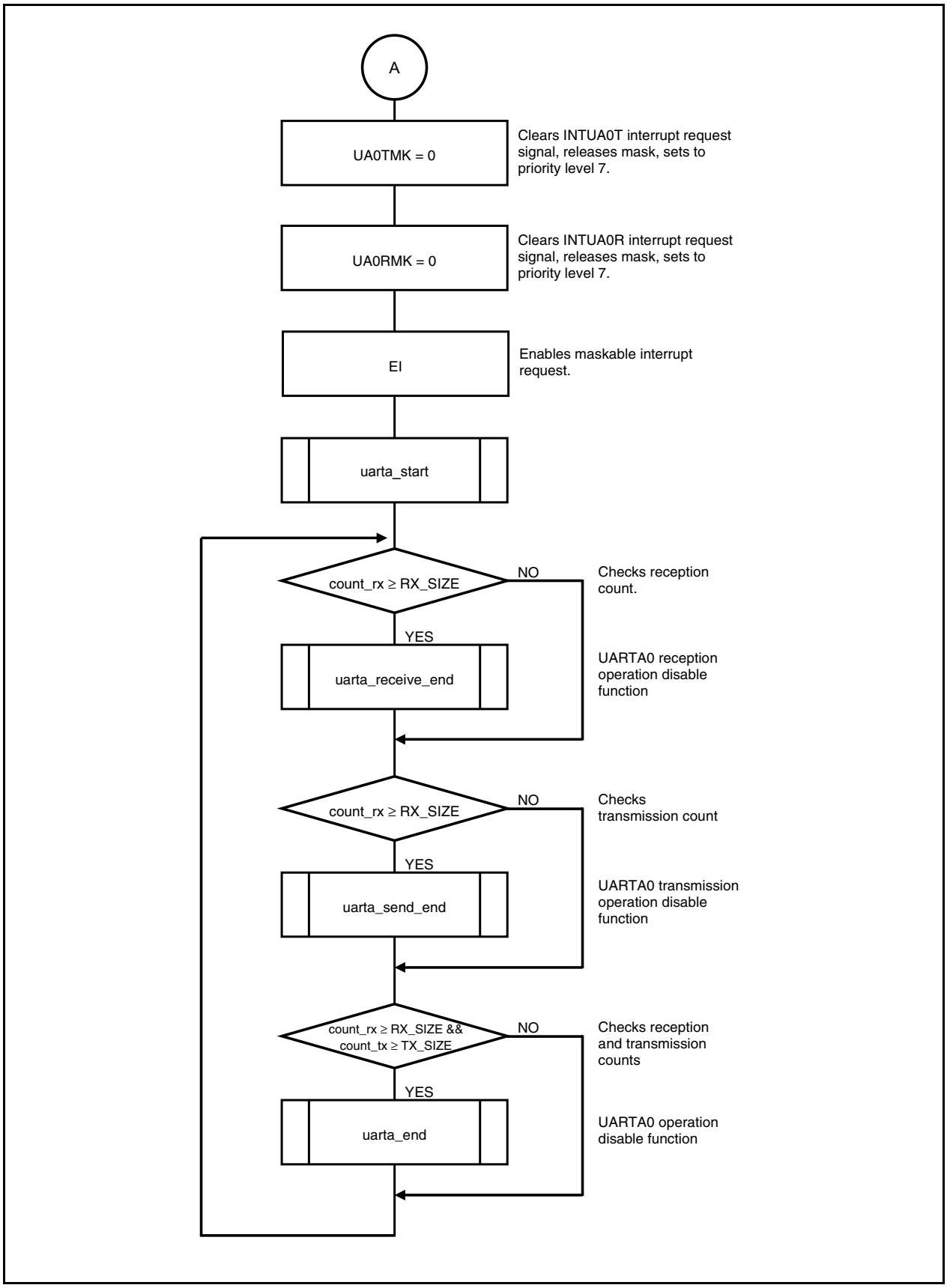


Figure 10-3. Asynchronous Serial Interface A (UARTAn) (3)

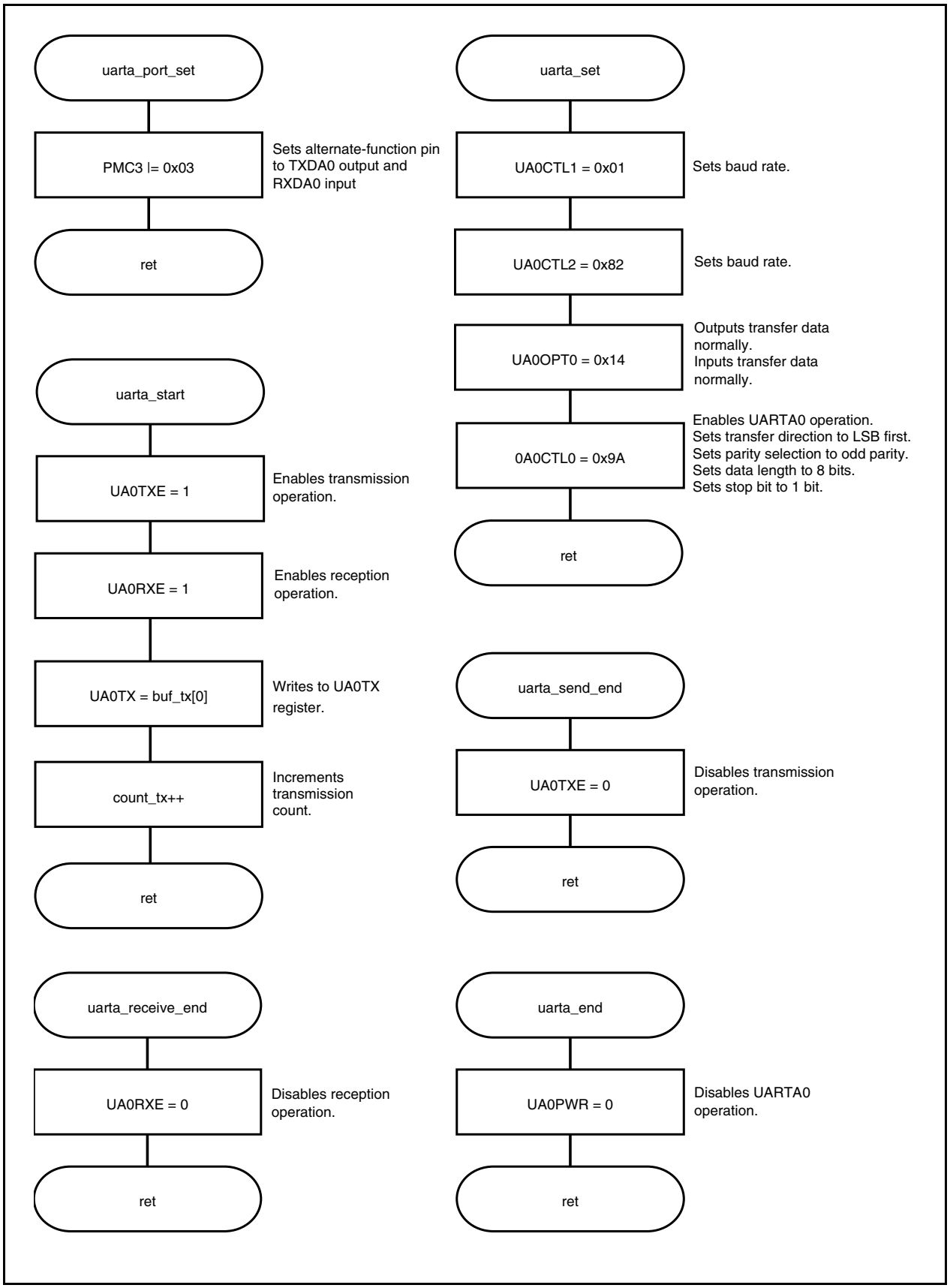
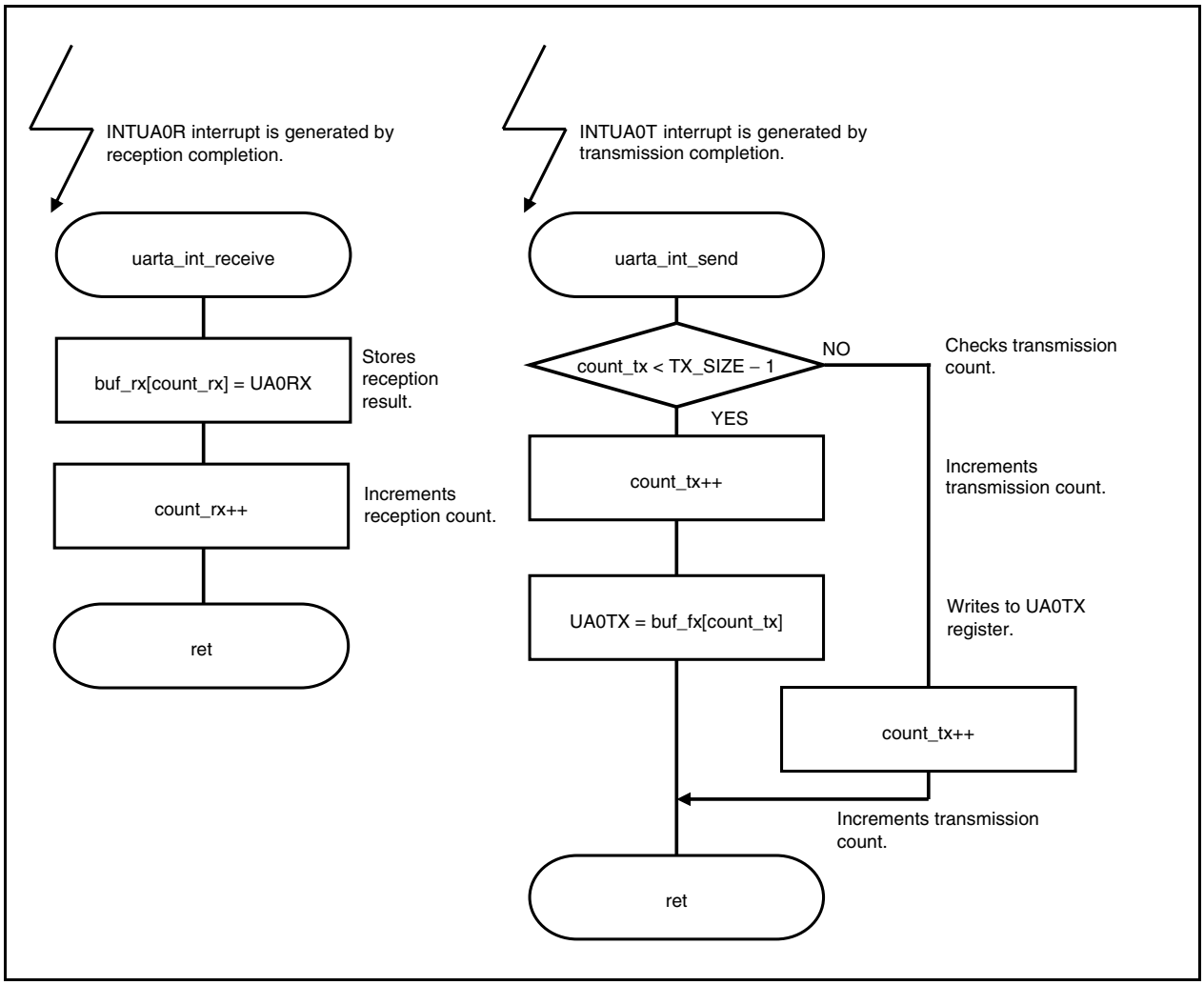


Figure 10-4. Asynchronous Serial Interface A (UARTAn) (4)



11.1 Continuous Transfer Mode (Master Mode, Transmission/Reception Mode)

[Function]	Sets communication mode to master mode and transfer direction mode to MSB first, and performs data reception for ten times in continuous transfer mode. Validates communication start trigger and sets communication clock to fxx/64, and transfer data length to 8 bits.
[Function name]	csib1_main
[Argument]	None
[Processing content]	Sets transmission count (count_tx) to initial value 0. Sets reception count (count_rx) to initial value 0 and starts transmission/reception after calling each setting function.
[SFRs used]	CB0RMK: 0 (Enables CSIB0 reception end interrupt (INTCB0R) process.) CB0TMK: 0 (Enables CSIB0 transmission end interrupt (INTCB0T) process.) CB0STR.CB0TSF: Communication status flag
[call functions]	csib_port, csib_set, csib_start, csib_end
[Variables]	unsigned char buf_tx[ ]: Transmit data storing buffer unsigned char buf_rx[ ]: Receive data storing buffer volatile unsigned char count_tx: Transmission count variable volatile unsigned char count_rx: Reception count variable unsigned char count: Transmit data generation variable
[Interrupts]	csib_int_send, csib_int_receive
[Interrupt sources]	INTCB0R, INTCB0T
[File name]	csib1\csib1.c
[Caution]	None

[Function name]	csib_port
[Processing content]	Sets port 4 as CSIB0 I/O pin.
[SFR used]	PMC4: 0x07 (Sets $\overline{\text{SCKB0}}$ I/O, SOB0 output and SIB0 input.)
[call function]	None
[Variable]	None
[File name]	csib1\csib1.c
[Caution]	None

[Function name]	csib_set
[Processing content]	Sets the CSIB0 control register.
[SFRs used]	CB0CTL1: 0x05 (Sets to communication type 1 and sets communication clock to fxx/64.) CB0CTL2: 0x00 (Sets transfer data length to 8 bits.) CB0CTL0: 0x63 (Enables CSIB0 transmission operation and reception operation, sets to MSB first and continuous transfer mode, and validates communication start trigger.)
[call function]	None
[Variable]	None
[File name]	csib1\csib1.c
[Caution]	The CB0TXE and CB0RXE bits of the CB0CTL0 register are rewritable only when the CB0PWR bit is 0. However, the CB0PWR bit can be set to 1 simultaneously.

[Function name]	csib_start
[Processing content]	Enables CSIB0 operation and writes to the transmit data register.
[SFRs used]	CB0CTL0.CB0PWR: 1 (Enables CSIB0 operation.) CB0TX Transmit data register
[call function]	None
[Variables]	unsigned char buf_tx[ ]: Transmit data storing buffer volatile char count_tx: Transmission count variable
[File name]	csib1\csib1.c
[Caution]	None

[Function name]	csib_end
[Processing content]	Disables CSIB0 operation and transmission/reception operation.
[SFRs used]	CB0CTL0.CB0PWR: 0 (Disables CSIB0 operation.) CB0CTL0.CB0RXE: 0 (Disables CSIB0 reception operation.) CB0CTL0.CB0TXE: 0 (Disables CSIB0 transmission operation.)
[call function]	None
[Variable]	None
[File name]	csib1\csib1.c
[Caution]	None

Interrupt functions

[Function name]	csib_int_send	
[Processing content]	Sets new data for transmitting next data.	
[SFR used]	CB0TX Transmit data register.	
[call function]	None	
[Variables]	unsigned char buf_tx[ ]:	Transmit data storing buffer
	volatile unsigned char count_tx:	Transmission count variable
[File name]	csib1\csib1.c	
[Caution]	None	

[Function name]	csib_int_receive	
[Processing content]	Stores receive data to buffer.	
[SFR used]	CB0RX Receive data register	
[call function]	None	
[Variables]	unsigned char buf_rx[ ]:	Receive data storing buffer
	volatile unsigned char count_rx:	Reception count variable
[File name]	csib1\csib1.c	
[Caution]	None	

Figure 11-1. Continuous Transfer Mode (Master Mode, Transmission/Reception Mode) (1)

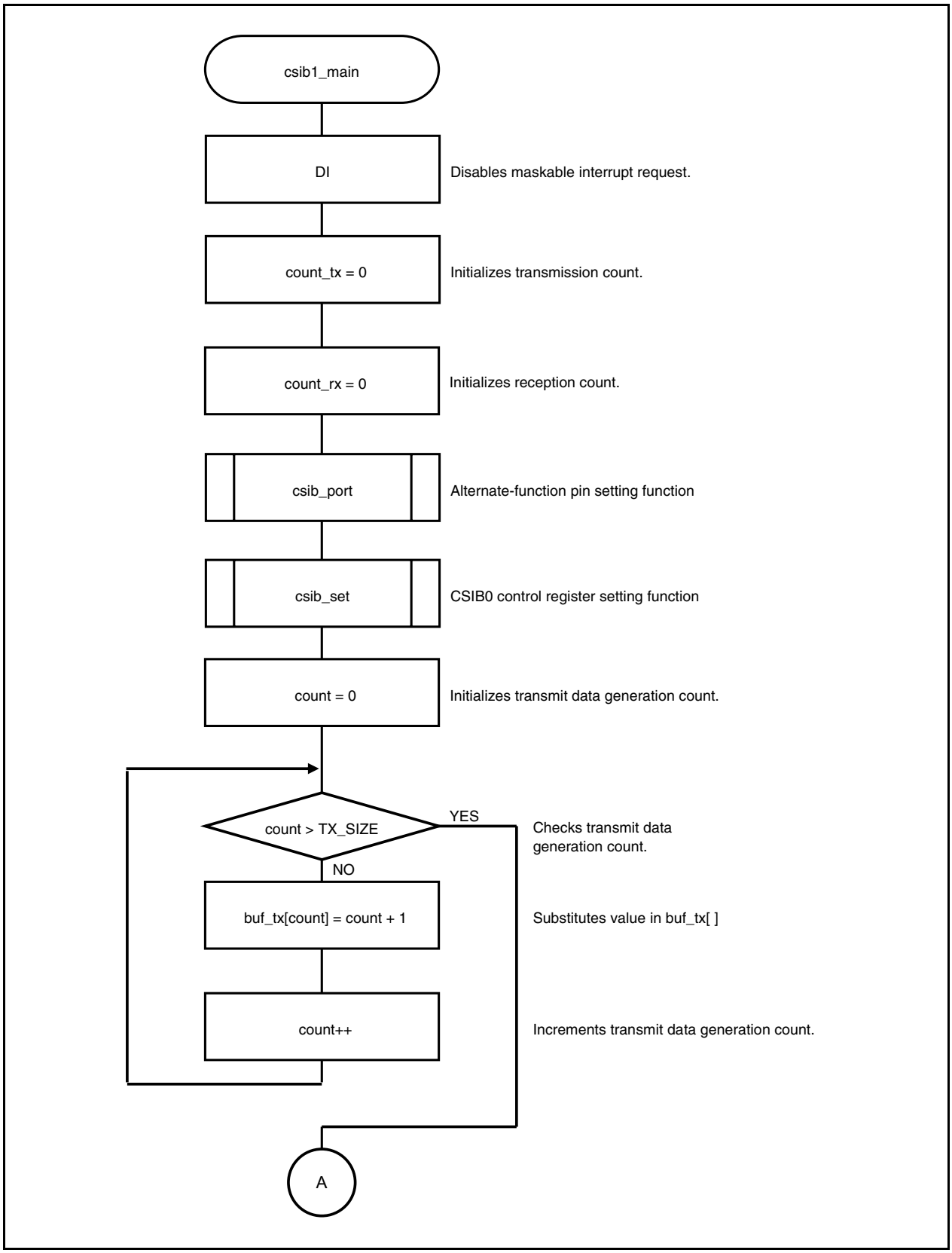


Figure 11-2. Continuous Transfer Mode (Master Mode, Transmission/Reception Mode) (2)

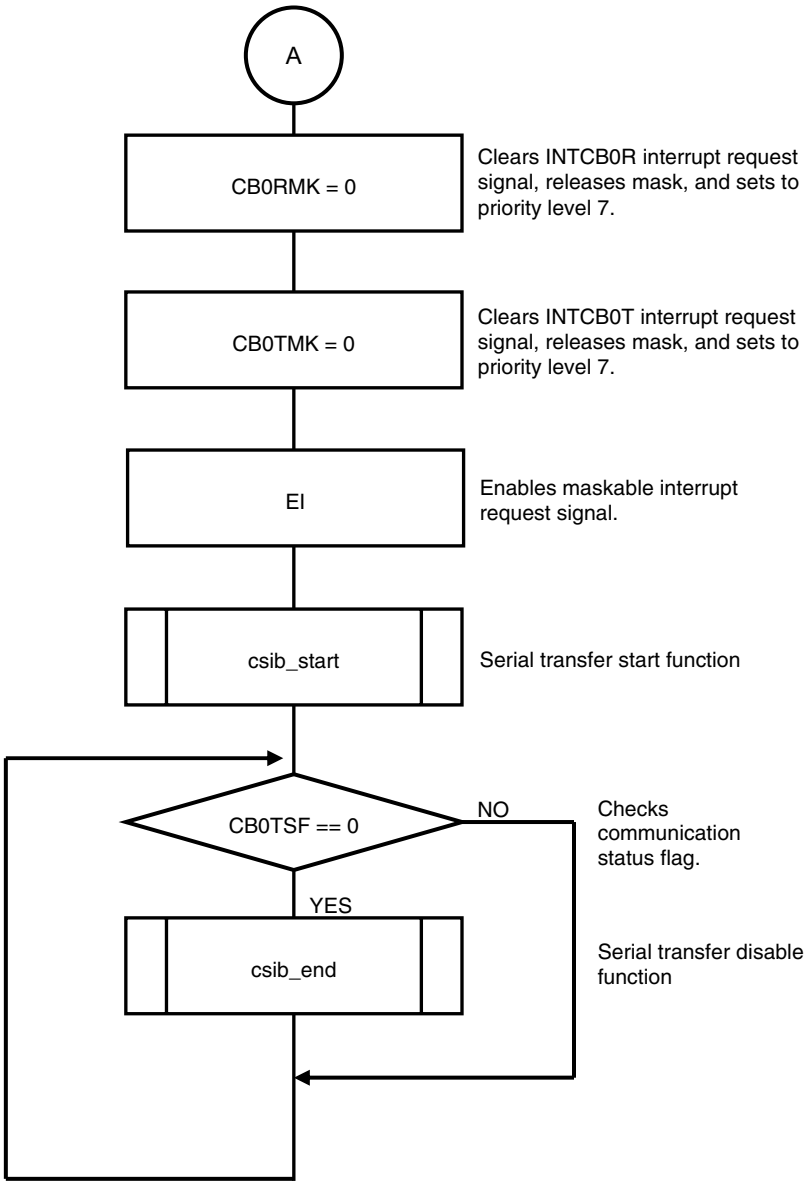


Figure 11-3. Continuous Transfer Mode (Master Mode, Transmission/Reception Mode) (3)

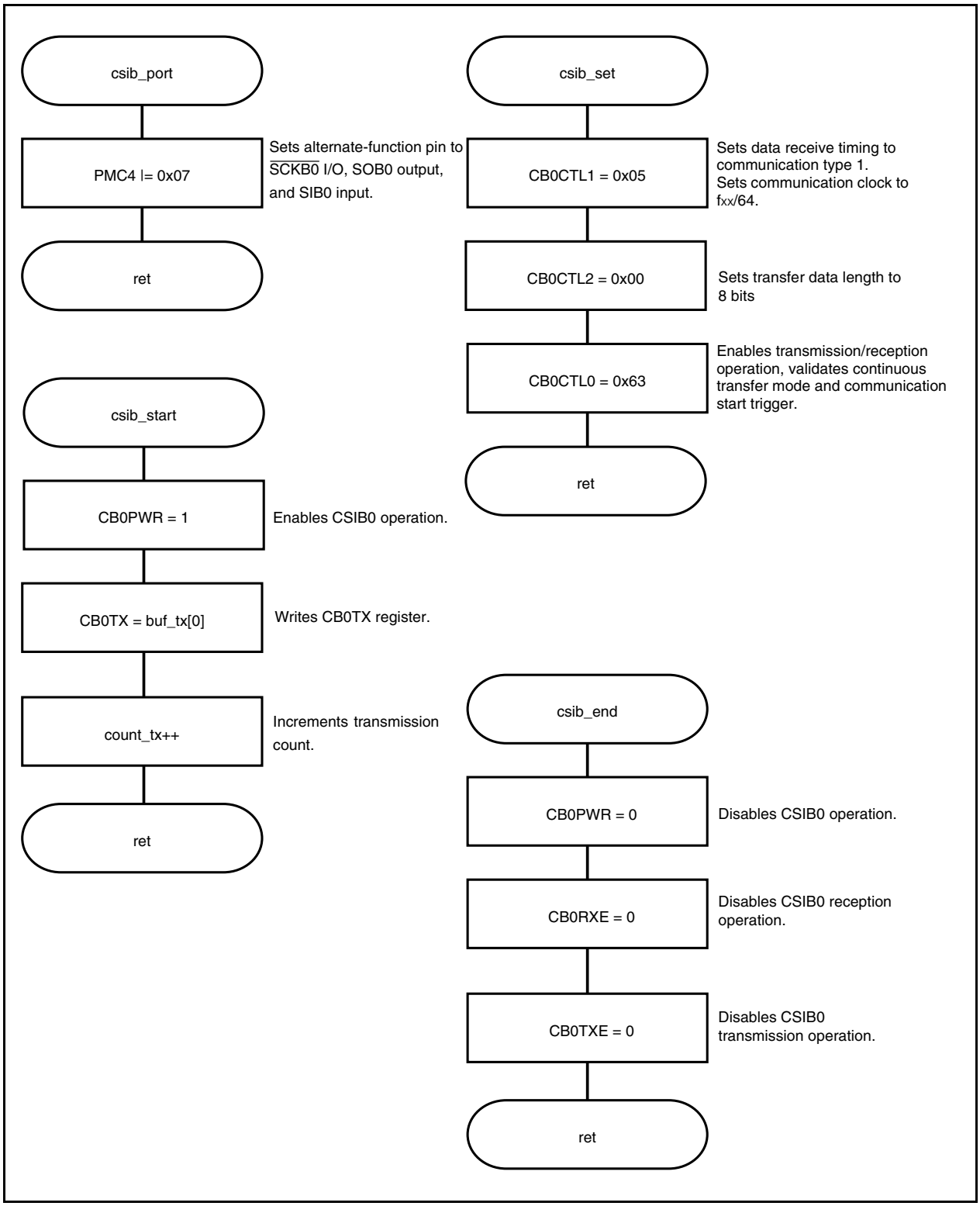
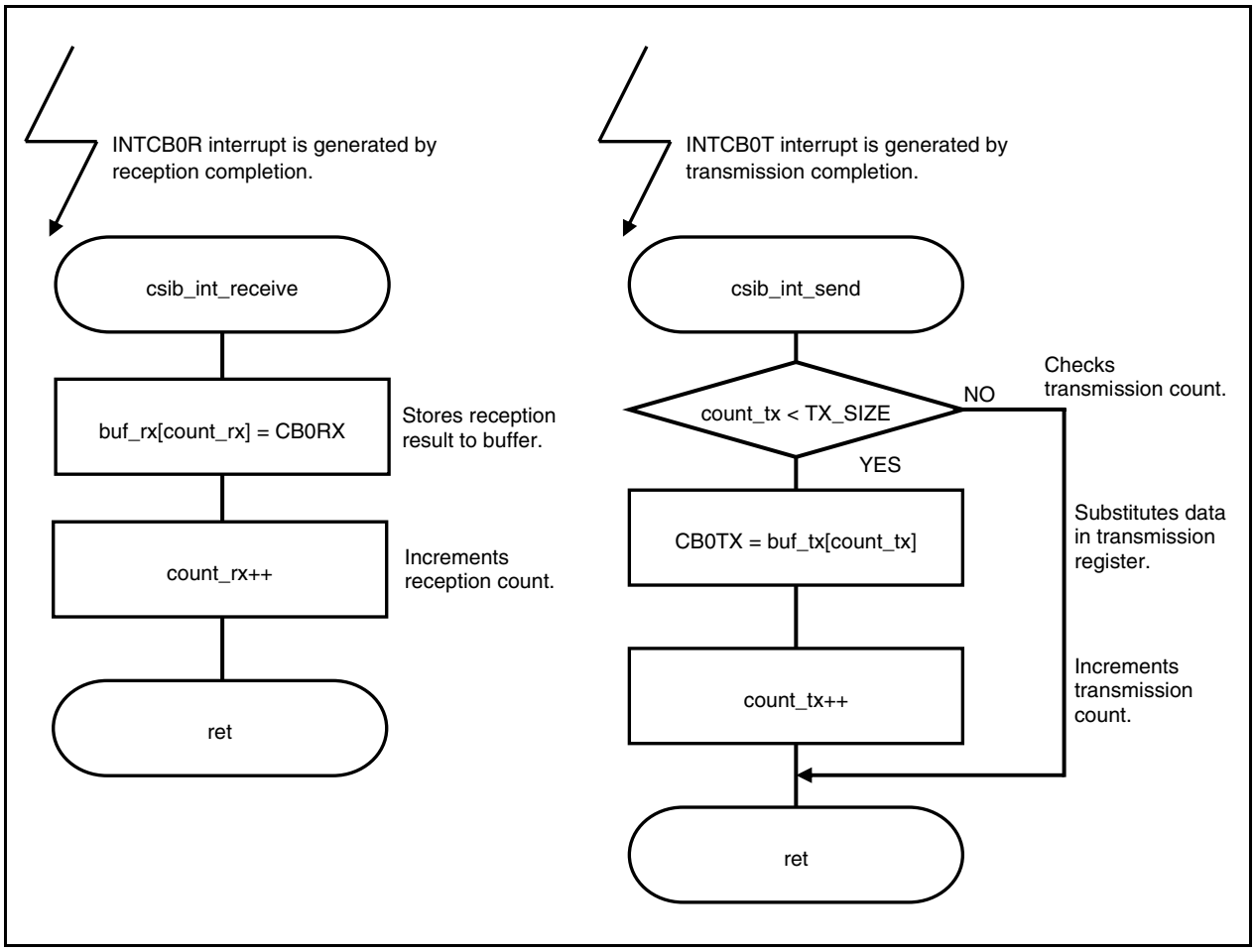


Figure 11-4. Continuous Transfer Mode (Master Mode, Transmission/Reception Mode) (4)



11.2 Continuous Transfer Mode (Slave Mode, Transmission/Reception Mode)

[Function]	Sets communication mode to slave mode and transfer direction mode to MSB first, and performs data reception for ten times in continuous transfer mode. Validates communication start trigger and sets communication clock to the external clock, and transfer data length to 8 bits.
[Function name]	csib2_main
[Argument]	None
[Processing content]	Sets transmission count (count_tx) to initial value 0. Sets reception count (count_rx) to initial value 0 and starts reception after calling each setting function.
[SFRs used]	CB0RMK: 0 (Enables CSIB0 reception end interrupt (INTCB0R) process.) CB0TMK: 0 (Enables CSIB0 transmission end interrupt (INTCB0T) process.) CB0STR.CB0TSF Communication status flag
[call functions]	csib_port, csib_set, csib_start, csib_end
[Variables]	unsigned char buf_tx[ ]: Transmit data storing buffer unsigned char buf_rx[ ]: Receive data storing buffer volatile unsigned char count_tx: Transmission count variable volatile unsigned char count_rx: Reception count variable unsigned char count: Transmit data generation variable
[Interrupts]	csib_int_send, csib_int_receive
[Interrupt sources]	INTCB0T, INTCB0R
[File name]	csib2\csib2.c
[Caution]	None

[Function name]	csib_port
[Argument]	None
[Processing content]	Sets port 4 as CSIB0 I/O pin.
[SFR used]	PMC4: 0x07 (Sets $\overline{\text{SCKB0}}$ I/O, SOB0 output and SIB0 input.)
[call function]	None
[Variable]	None
[File name]	csib2\csib2.c
[Caution]	None

[Function name]	csib_set
[Processing content]	Sets the CSIB0 control register.
[SFRs used]	CB0CTL1: 0x07 (Sets to communication type 1 and sets communication clock to the external clock.) CB0CTL2: 0x00 (Sets transfer data length to 8 bits.) CB0CTL0: 0x63 (Enables CSIB0 transmission operation and reception operation, sets to MSB first and continuous transfer mode, and validates communication start trigger.)
[call function]	None
[Variable]	None
[File name]	csib2\csib2.c
[Caution]	CB0TXE and CB0RXE bits of the CB0CTL0 register are rewritable only when the CB0PWR bit is 0. However, the CB0PWR bit can be set to 1 simultaneously.

[Function name]	csib_start
[Processing content]	Enables CSIB0 operation and writes to the transmit data register.
[SFRs used]	CB0CTL0.CB0PWR: 1 (Enables CSIB0 operation.) CB0TX Transmit data register
[call function]	None
[Variables]	unsigned char buf_tx[ ]: Transmit data storing buffer volatile unsigned char count_tx: Transmission count variable
[File name]	csib2\csib2.c
[Caution]	None

[Function name]	csib_end
[Processing content]	Disables CSIB0 operation and transmission/reception operation.
[SFR used]	CB0CTL0.CB0PWR: 0 (Disables CSIB0 operation.) CB0CTL0.CS0RXE: 0 (Disables CSIB0 reception operation.) CB0CTL0.CB0TXE: 0 (Disables CSIB0 transmission operation.)
[call function]	None
[Variable]	None
[File name]	csib2\csib2.c
[Caution]	None

Interrupt functions

[Function name]	csib_int_send	
[Processing content]	Sets new data for transmitting next data.	
[SFR used]	CB0TX Transmit data register.	
[call function]	None	
[Variables]	unsigned char buf_tx[ ]:	Transmit data storing buffer
	volatile unsigned char count_tx:	Transmission count variable
[File name]	csib2\csib2.c	
[Caution]	None	

[Function name]	csib_int_receive	
[Processing content]	Stores receive data to buffer.	
[SFR used]	CB0RX Receive data register	
[call function]	None	
[Variables]	unsigned char buf_rx[ ]:	Receive data storing buffer
	volatile unsigned char count_rx:	Reception count variable
[File name]	csib2\csib2.c	
[Caution]	None	

Figure 11-5. Continuous Transfer Mode (Slave Mode, Transmission/Reception Mode) (1)

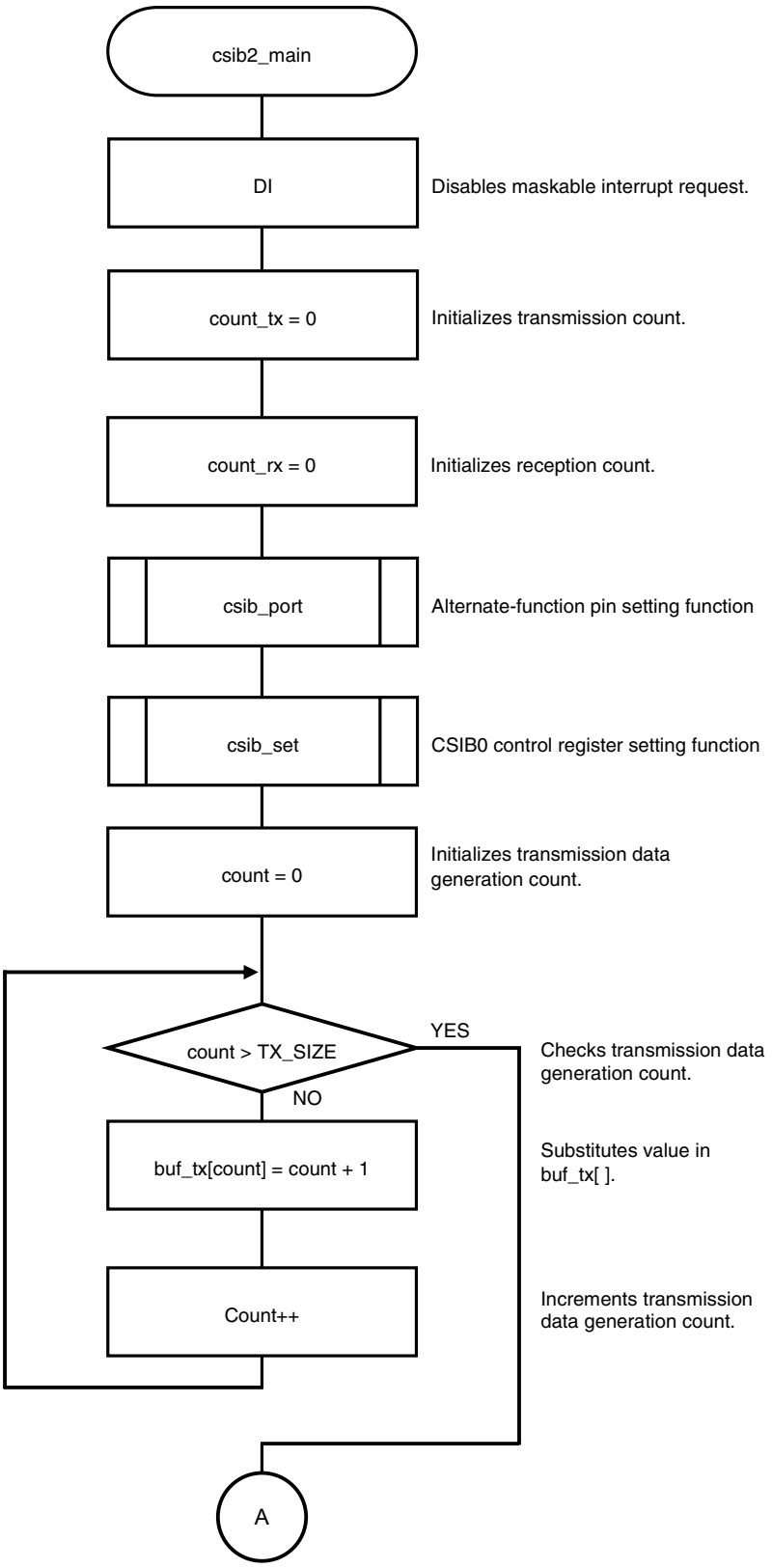


Figure 11-6. Continuous Transfer Mode (Slave Mode, Transmission/Reception Mode) (2)

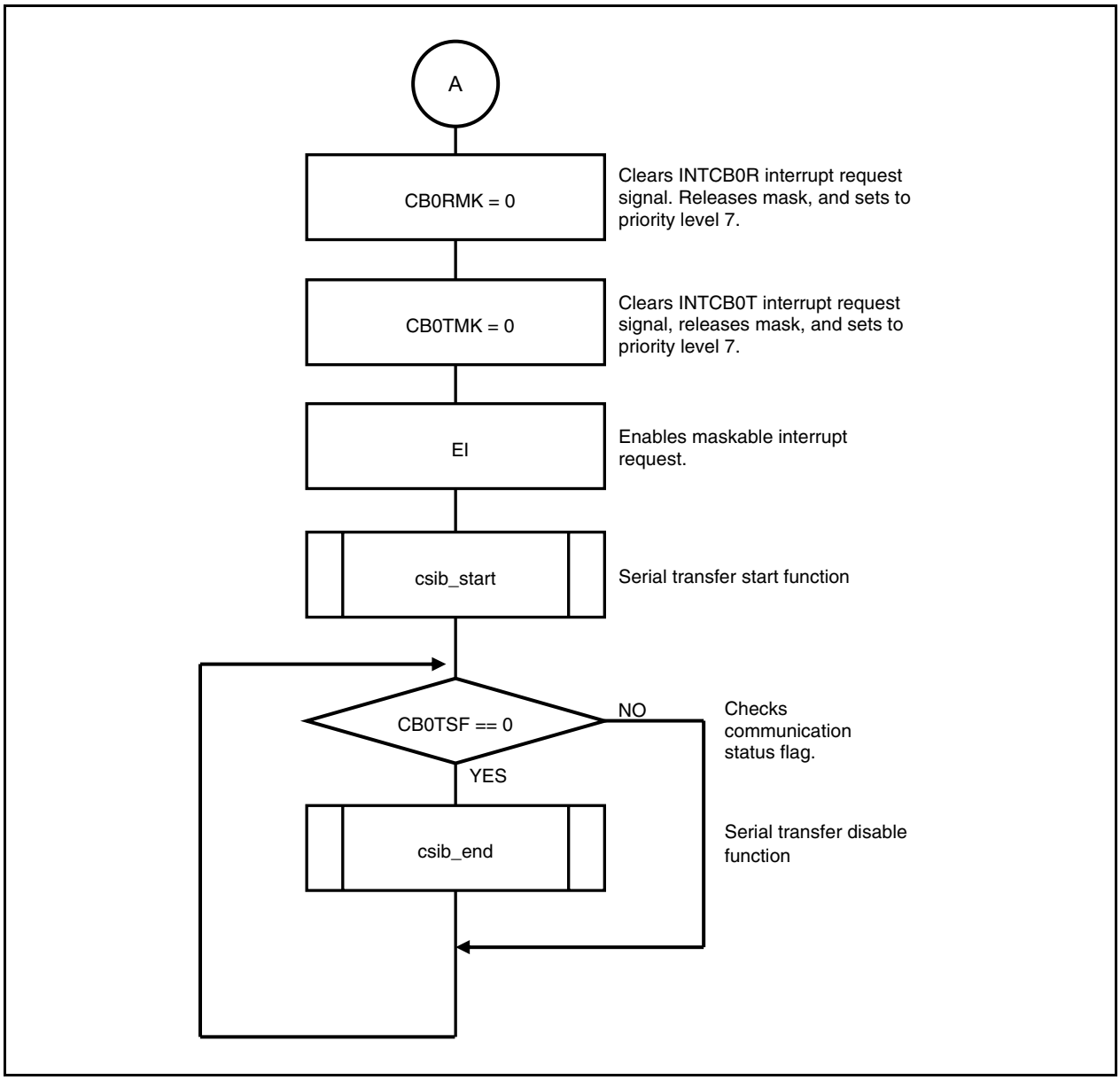


Figure 11-7. Continuous Transfer Mode (Slave Mode, Transmission/Reception Mode) (3)

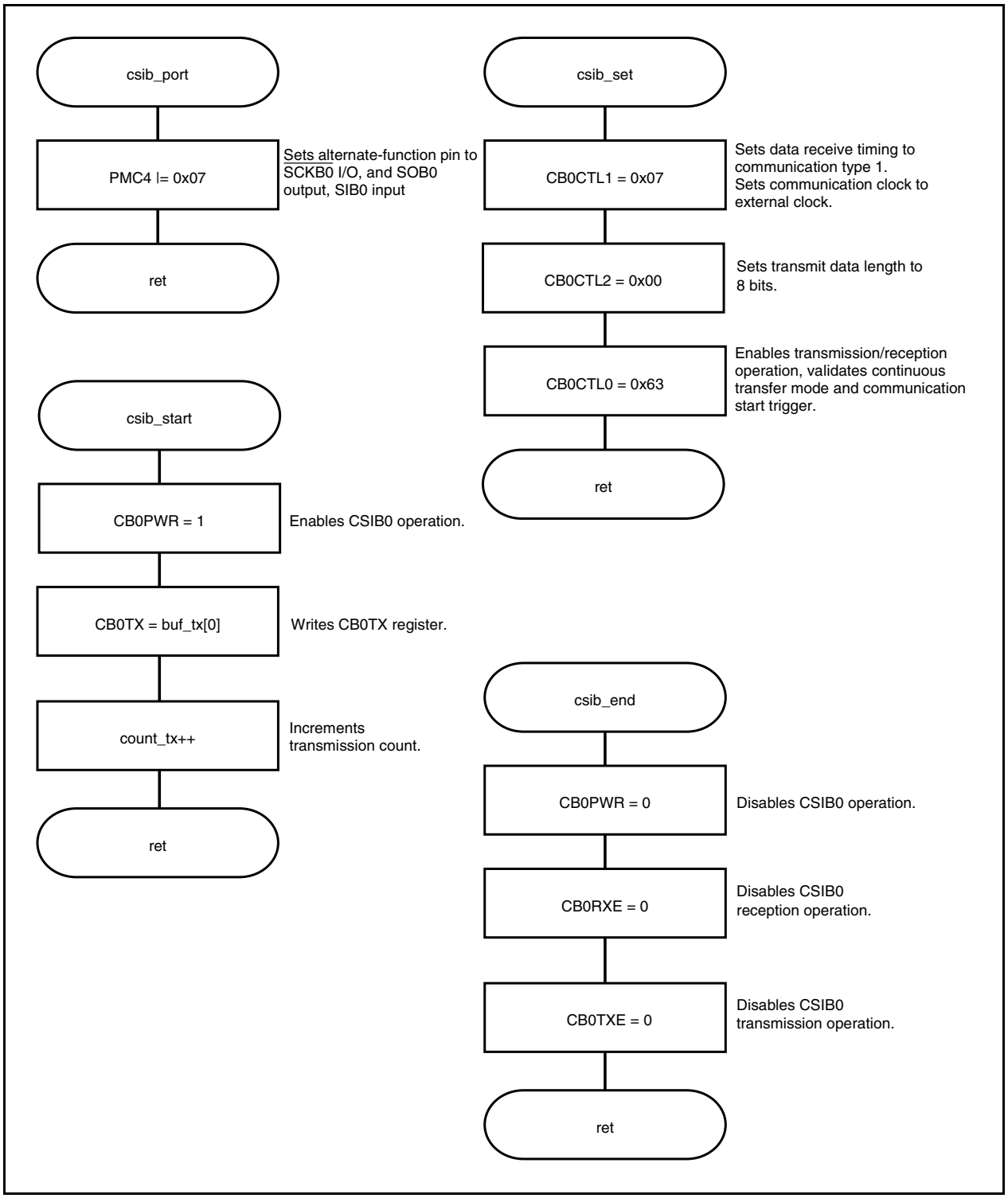
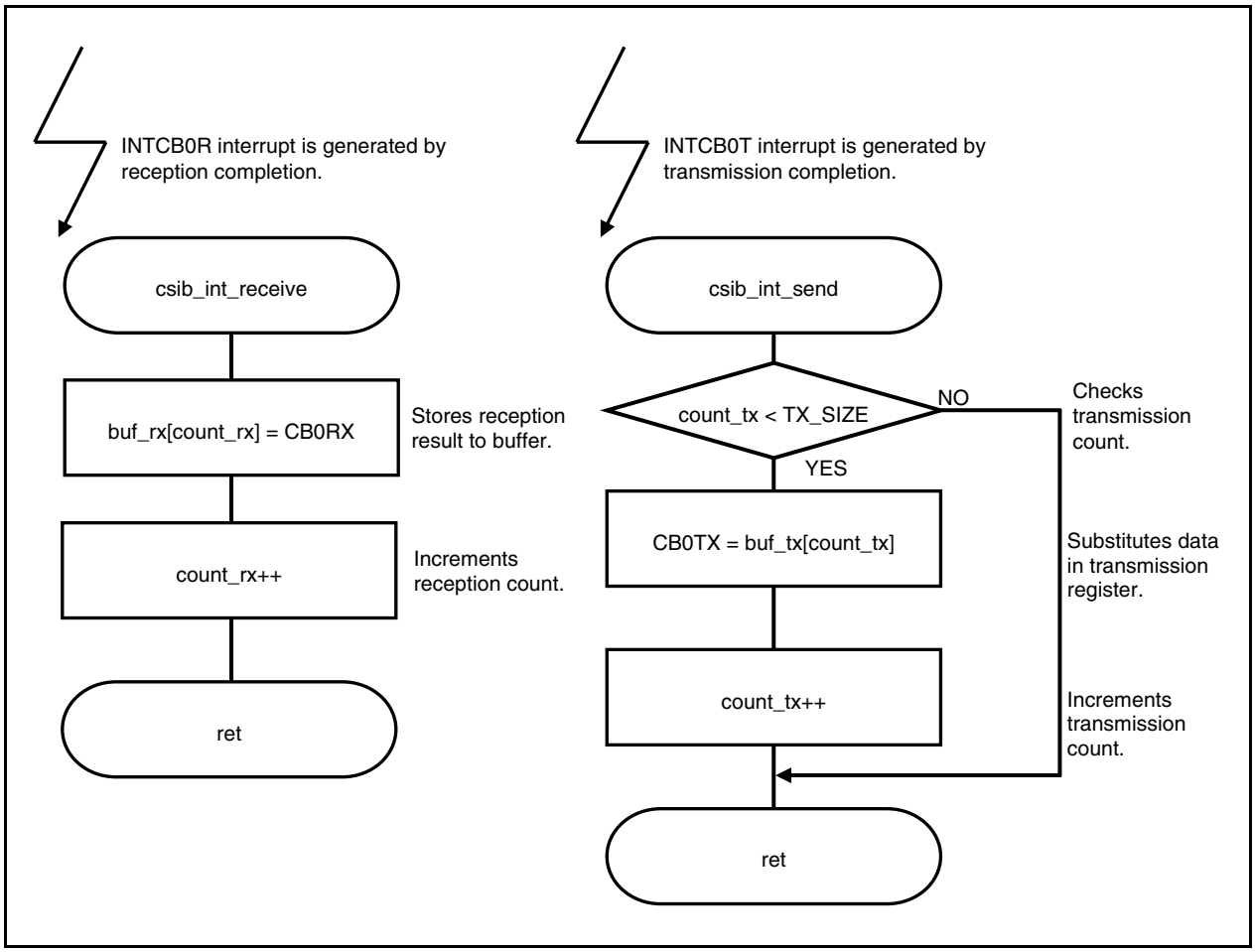


Figure 11-8. Continuous Transfer Mode (Slave Mode, Transmission/Reception Mode) (4)



11.3 Single Transfer Mode (Master Mode, Transmission Mode)

[Function]	Sets communication mode to master mode and transfer direction mode to MSB first, and performs data transmission for ten times in single transfer mode. Validates communication start trigger and sets communication clock to fxx/64, and transfer data length to 8 bits.
[Function name]	csib3_main
[Argument]	None
[Processing content]	Sets transmission count (count_tx) to initial value 0. Starts transmission after calling each setting function.
[SFR used]	CB0RMK: 0 (Clears CSIB0 reception end interrupt request signal (INTCB0R), releases mask, and sets to priority level 7.)
[call functions]	csib_port, csib_set, csib_start, csib_end
[Variables]	unsigned char buf_tx[ ]: Transmit data storing buffer volatile unsigned char count_tx: Transmission count variable unsigned char count: Transmit data generation variable
[Interrupt]	csib_int_send
[Interrupt source]	INTCB0R
[File name]	csib3\csib3.c
[Caution]	None

[Function name]	csib_port
[Processing content]	Sets port 4 as CSIB0 I/O pin.
[SFR used]	PMC4: 0x06 (Sets $\overline{\text{SCKB0}}$ I/O and SOB0 output.)
[call function]	None
[Variable]	None
[File name]	csib3\csib3.c
[Caution]	None

[Function name]	csib_set
[Processing content]	Sets the CSIB0 control register.
[SFRs used]	CB0CTL1: 0x05 (Sets to communication type 1 and sets communication clock to f_{XX}/64.) CB0CTL2: 0x00 (Sets transfer data length to 8 bits.) CB0CTL0: 0x41 (Enables CSIB0 transmission operation, sets to MSB first and single transfer mode, and validates communication start trigger.)
[call function]	None
[Variable]	None
[File name]	csib3\csib3.c
[Caution]	CB0TXE and CB0RXE bits of the CB0CTL0 register are rewritable only when the CB0PWR bit is 0. However, the CB0PWR bit can be set to 1 simultaneously.

[Function name]	csib_start
[Processing content]	Enables CSIB0 operation and writes to the transmit data register.
[SFRs used]	CB0CTL0.CB0PWR: 1 (Enables CSIB0 operation.) CB0TX Transmit data register
[call function]	None
[Variables]	unsigned char buf_tx[ ]: Transmit data storing buffer volatile unsigned char count_tx: Transmission count variable
[File name]	csib3\csib3.c
[Caution]	None

[Function name]	csib_end
[Processing content]	Disables CSIB0 operation and transmission operation.
[SFRs used]	CB0CTL0.CB0PWR: 0 (Disables CSIB0 operation.) CB0CTL0.CB0TXE: 0 (Disables CSIB0 transmission operation.)
[call function]	None
[Variable]	None
[File name]	csib3\csib3.c
[Caution]	None

Interrupt function

[Function name]	csib_int_send
[Processing content]	Sets new data for transmitting next data.
[SFR used]	CB0TX Transmit data register.
[call function]	None
[Variables]	unsigned char buf_tx[ ]: Transmit data storing buffer volatile unsigned char count_tx: Transmission count variable
[File name]	csib3\csib3.c
[Caution]	None

Figure 11-9. Single Transfer Mode (Slave Mode, Transmission Mode) (1)

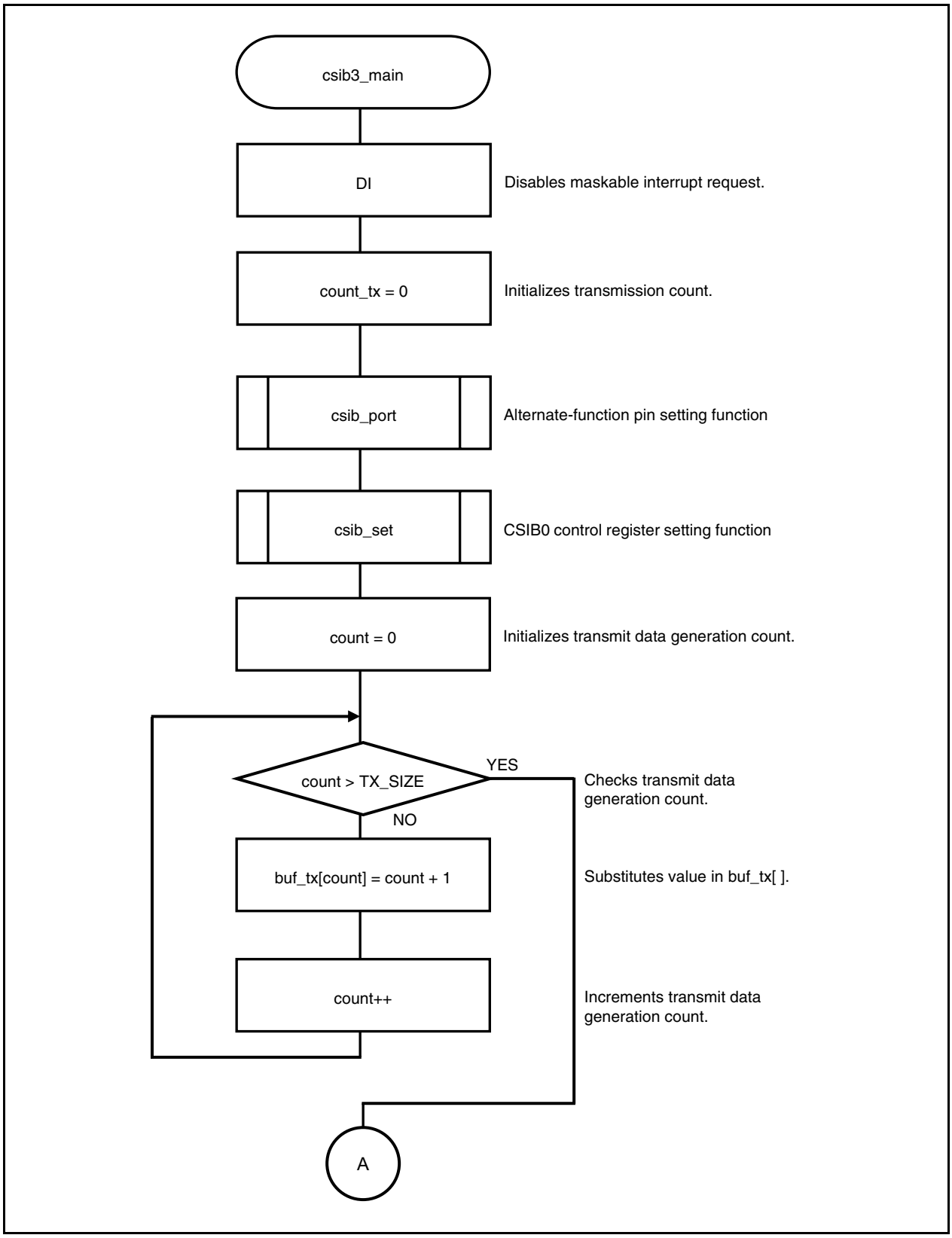


Figure 11-10. Single Transfer Mode (Slave Mode, Transmission Mode) (2)

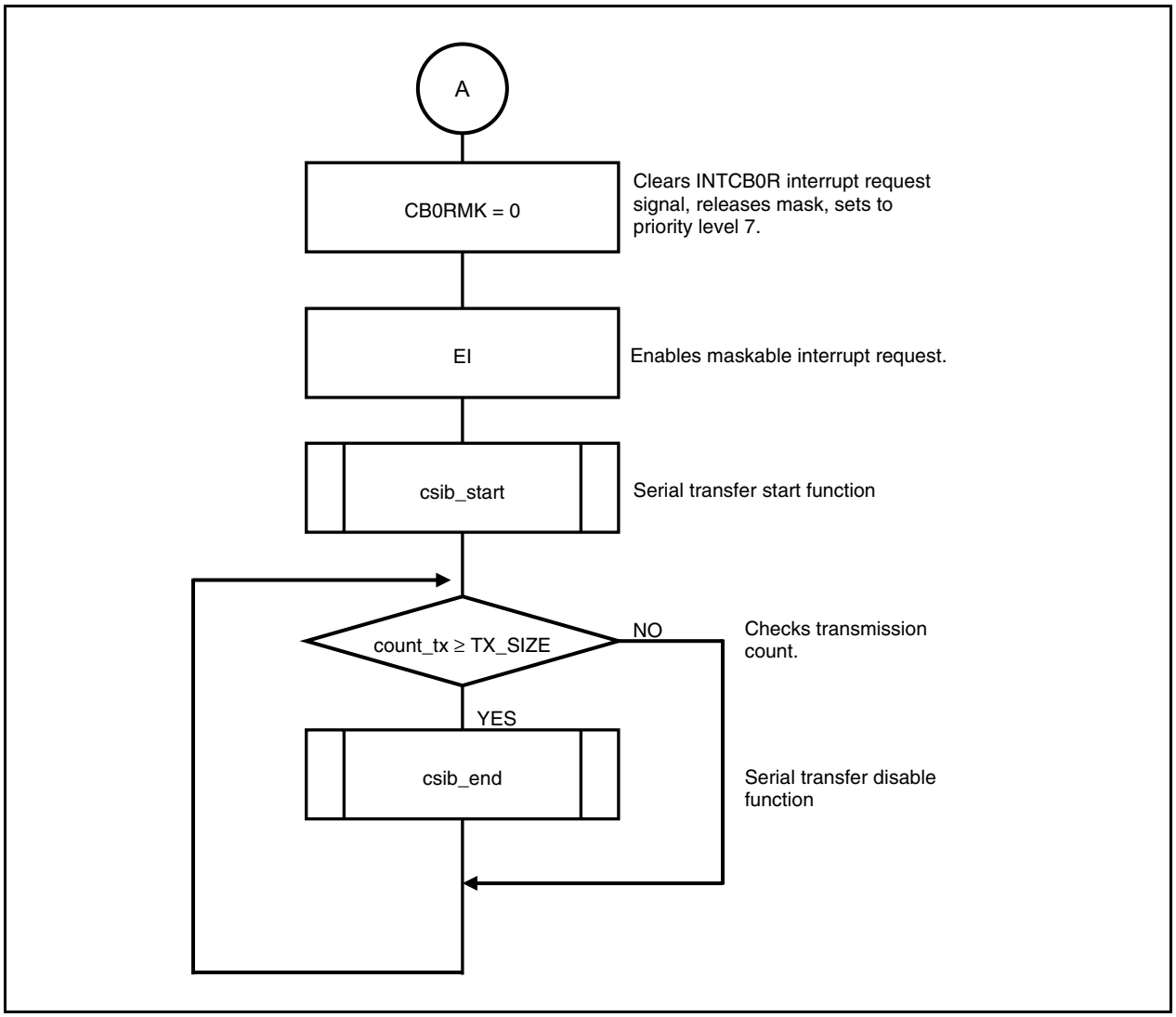


Figure 11-11. Single Transfer Mode (Slave Mode, Transmission Mode) (3)

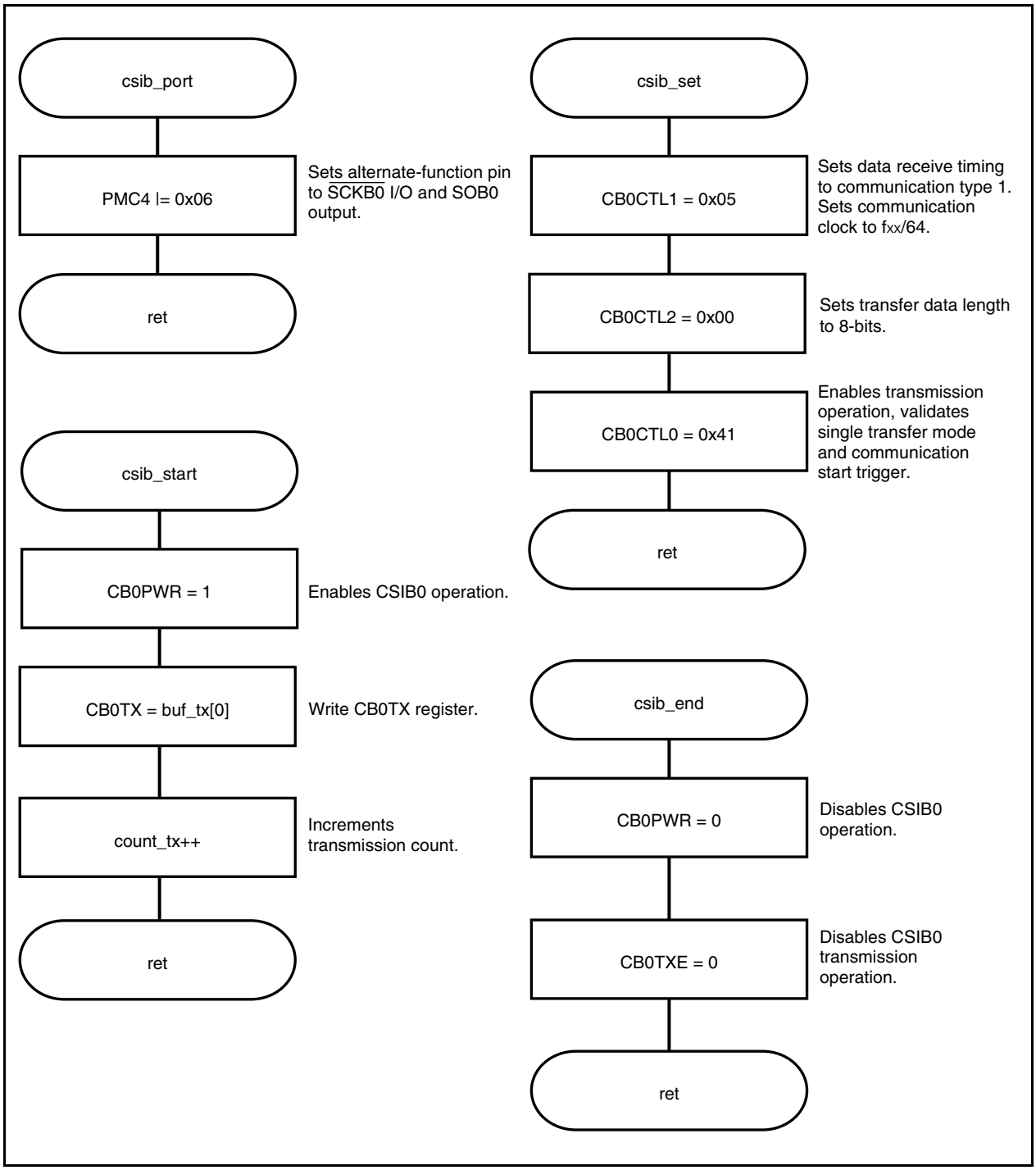
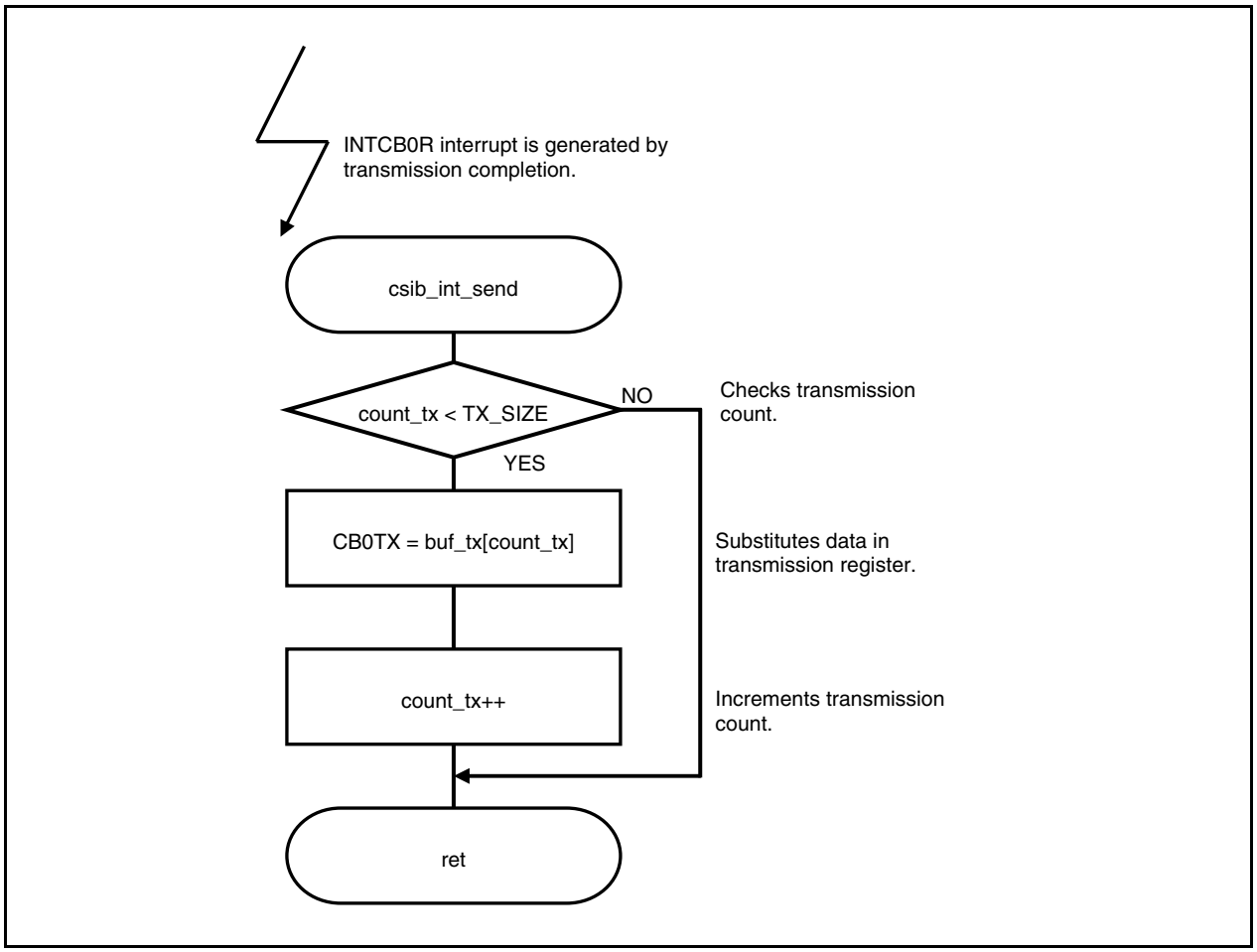


Figure 11-12. Single Transfer Mode (Slave Mode, Transmission Mode) (4)



11.4 Single Transfer Mode (Slave Mode, Reception Mode)

[Function]	Sets communication mode to slave mode and transfer direction mode to MSB first, and performs data reception for ten times in single transfer mode. Validates communication start trigger and sets communication clock to external clock, and transfer data length to 8 bits.
[Function name]	csib4_main
[Argument]	None
[Processing content]	Sets transmission count (count_tx) to initial value 0. Starts reception after calling each setting function.
[SFR used]	CB0RMK: 0 (Clears CSIB0 reception end interrupt request signal (INTCB0R), releases mask, and sets to priority level 7.)
[call functions]	csib_port, csib_set, csib_start, csib_end
[Variables]	unsigned char buf_rx[ ]: Receive data storing buffer volatile unsigned char count_rx: Reception count variable
[Interrupt]	csib_int_receive
[Interrupt source]	INTCB0R
[File name]	csib4\csib4.c
[Caution]	None

[Function name]	csib_port
[Processing content]	Sets port 4 as CSIB0 I/O pin.
[SFR used]	PMC4: 0x05 (Sets $\overline{\text{SCKB0}}$ I/O and SIB0 output.)
[call function]	None
[Variable]	None
[File name]	csib4\csib4.c
[Caution]	None

[Function name]	csib_set
[Processing content]	Sets the CSIB0 control register.
[SFRs used]	CB0CTL1: 0x07 (Sets to communication type 1 and sets communication clock to external clock.) CB0CTL2: 0x00 (Sets transfer data length to 8 bits.) CB0CTL0: 0x21 (Enables CSIB reception operation, sets to MSB first and single transfer mode, and validates communication start trigger.)
[call function]	None
[Variable]	None
[File name]	csib4\csib4.c
[Caution]	CB0TXE and CB0RXE bits of the CB0CTL0 register are rewritable only when the CB0PWR bit is 0. However, the CB0PWR bit can be set to 1 simultaneously.

[Function name]	csib_start
[Processing content]	Enables CSIBO operation and performs dummy read on the receive data register.
[SFRs used]	CB0CTL0.CB0PWR: 1 (Enables CSIB0 operation.) CB0RX Receive data register
[call function]	None
[Variable]	unsigned char buf_rx[ ]: Receive data storing buffer
[File name]	csib4\csib4.c
[Caution]	None

[Function name]	csib_end
[Processing content]	Disables CSIB0 operation and reception operation.
[SFRs used]	CB0CTL0.CB0PWR: 0 (Disables CSIB0 operation.) CB0CTL0.CB0RXE: 0 (Disables CSIB0 reception operation.)
[call function]	None
[Variable]	None
[File name]	csib4\csib4.c
[Caution]	None

Interrupt function

[Function name]	csib_int_receive
[Processing content]	Stores receive data to buffer
[SFRs used]	CB0RX Receive data register CB0CTL0.CB0SCE: 0 (Disables communication start trigger.)
[call function]	None
[Variables]	unsigned char buf_rx[ ]: Receive data storing buffer volatile unsigned char count_rx: Reception count variable
[File name]	csib4\csib4.c
[Caution]	None

Figure 11-13. Single Transfer Mode (Slave Mode, Reception Mode) (1)

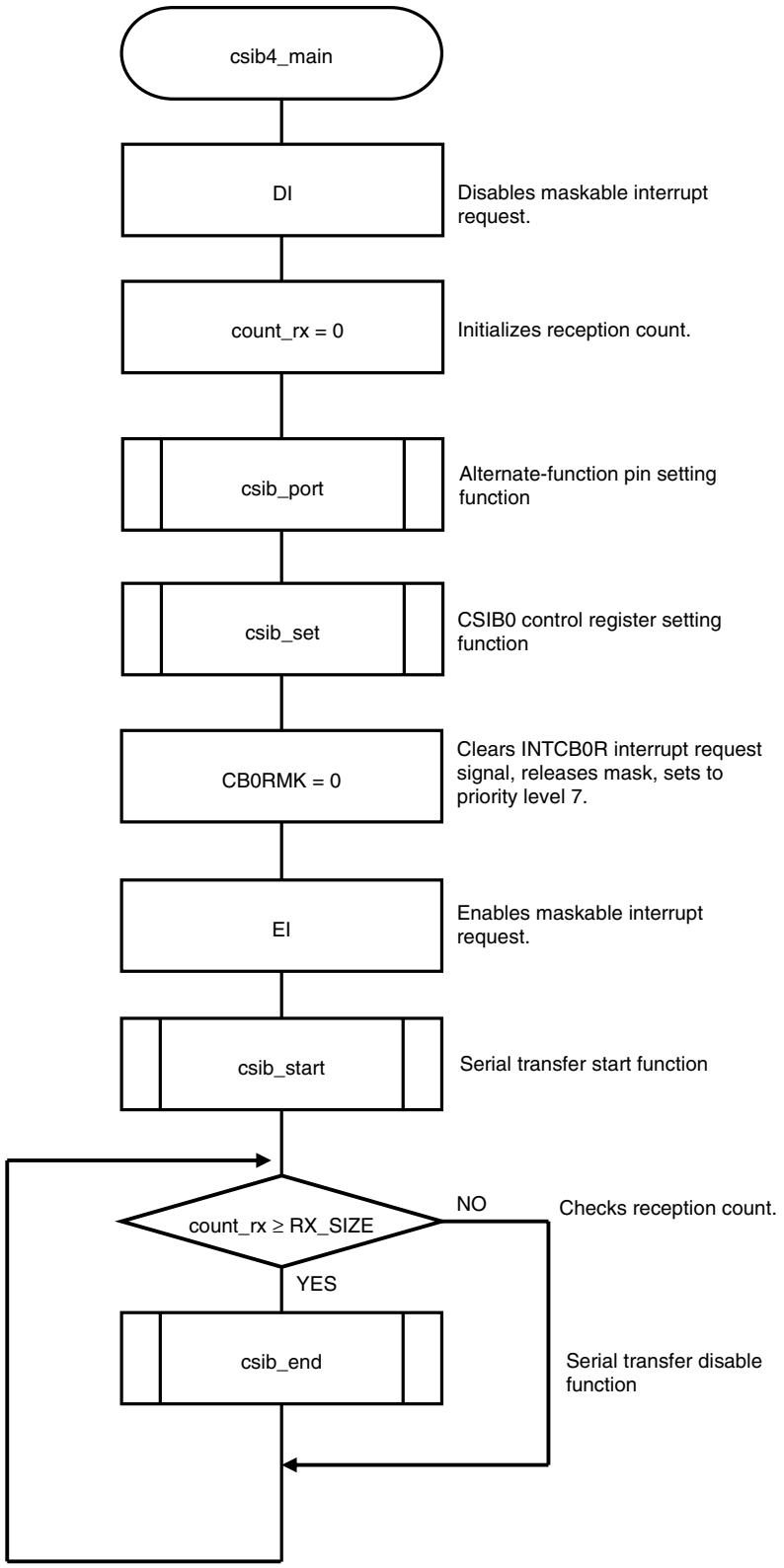


Figure 11-14. Single Transfer Mode (Slave Mode, Reception Mode) (2)

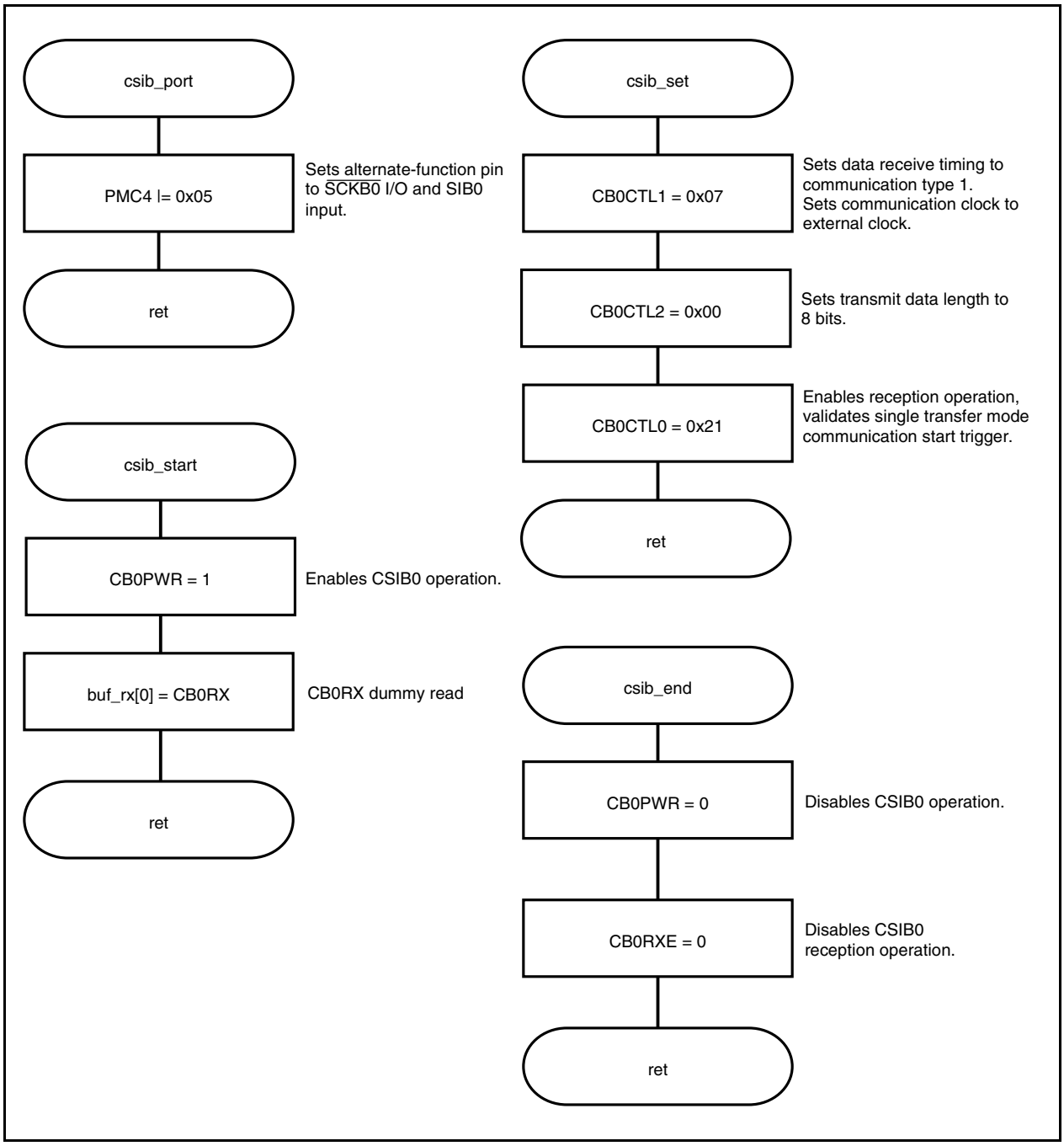
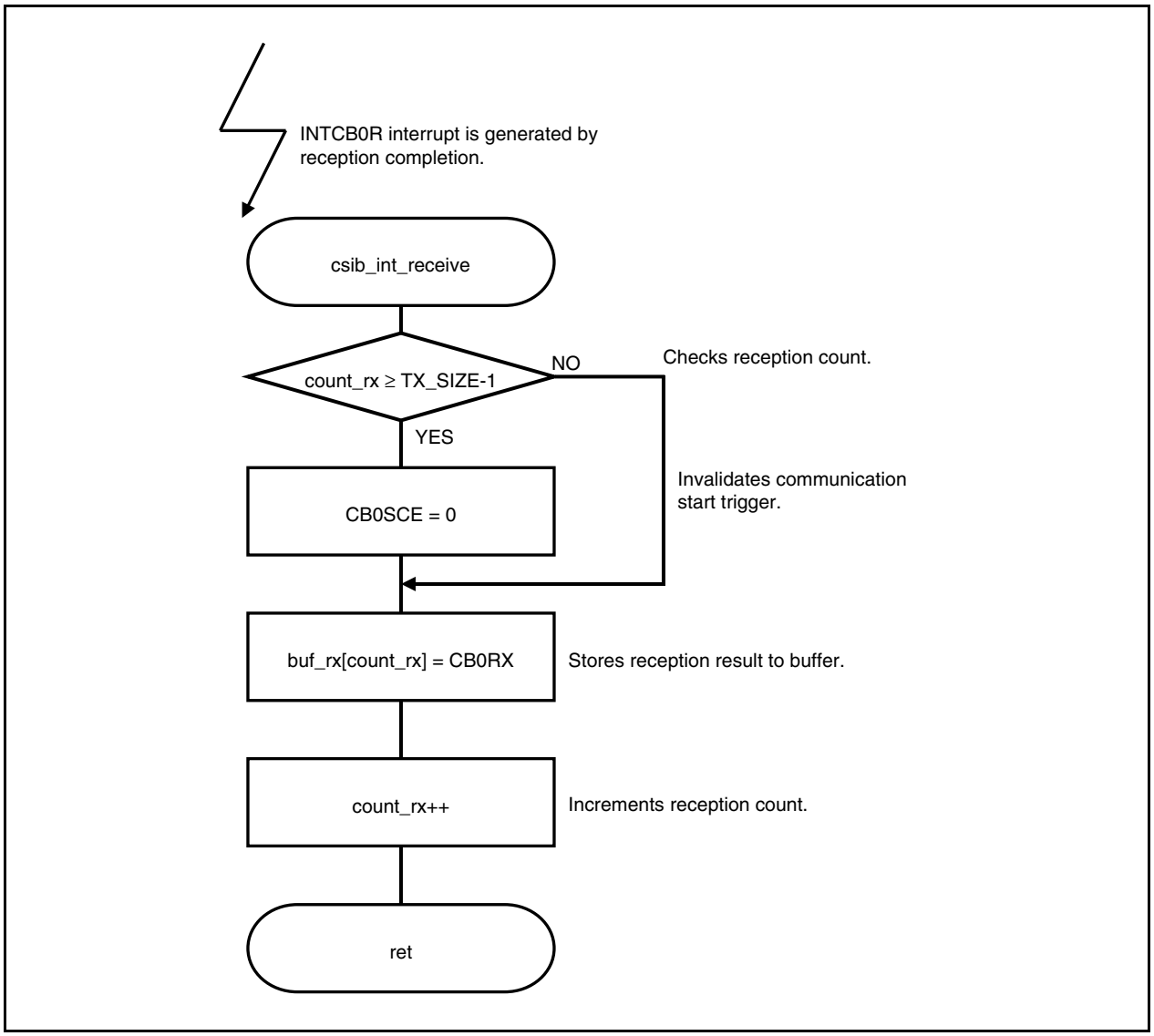


Figure 11-15. Single Transfer Mode (Slave Mode, Reception Mode) (3)



12.1 External Interrupt

[Function]	Implements an external interrupt by using INTP3.
[Function name]	ex_int_main
[Argument]	None
[Processing content]	• Performs each interrupt setting by calling the initial setting function. • Enables interrupt.
[SFR used]	None
[call functions]	ex_int_init, ex_int_end
[Variable]	unsigned char flag
[Interrupt]	ex_int_p3_handler
[Interrupt source]	INTP3
[File name]	ex_intrrupt\ex_interrupt.c
[Caution]	None

[Function name]	ex_int_init
[Processing content]	Performs initial setting of external interrupt.
[SFR used]	None
[call functions]	ex_int_p3_init, ex_int_interrupt_init
[Variable]	None
[File name]	ex_intrrupt\ex_interrupt.c
[Caution]	None

[Function name]	ex_int_p3_init
[Processing content]	Performs settings of edge detection by setting the P06 pin to the external maskable interrupt request input.
[SFRs used]	PMC0: 0x40 (Sets P06 pin to external maskable interrupt request input (INTP3).) INTR0: 0x40 (Sets valid edge to rising edge.) INTF0: 0x00 (Sets valid edge to rising edge.)
[call function]	None
[Variable]	None
[File name]	ex_intrrupt\ex_interrupt.c
[Caution]	None

[Function name]	ex_int_interrupt_init
[Processing content]	Performs initial setting of interrupt.
[SFRs used]	PIC3: 0x45 (Sets priority level of INTP3 interrupt request signal to level 5.) PIC3. PIF3: 0 (Clears INTP3 interrupt request flag.) PIC3.PMK3: 0 (Enables INTP3 interrupt servicing.)
[call function]	None
[Variable]	None
[File name]	ex_intrrupt\ex_interrupt.c
[Caution]	None

[Function name]	ex_in_end
[Processing content]	Sets pin to port mode and edge specification to “No edge detection” after external interrupt occurrence.
[SFRs used]	INTR0: 0x00 (Sets to “No edge detection”.) INTF0: 0x00 (Sets to “No edge detection”.) PMC0: 0x00 (Sets P06 pin to port mode.) PIC3.PMK3: 1 (Disables INTP3 interrupt servicing.)
[call function]	None
[Variable]	None
[File name]	ex_intrrupt\ex_interrupt.c
[Caution]	None

Interrupt function

[Function name]	ex_int_p3_handler
[Processing content]	Checks occurrence of external interrupt. There is no specific processing content.
[SFR used]	None
[call function]	None
[Variable]	unsigned char flag
[File name]	ex_intrrupt\ex_interrupt.c
[Caution]	None

Figure 12-1. External Interrupt (1)

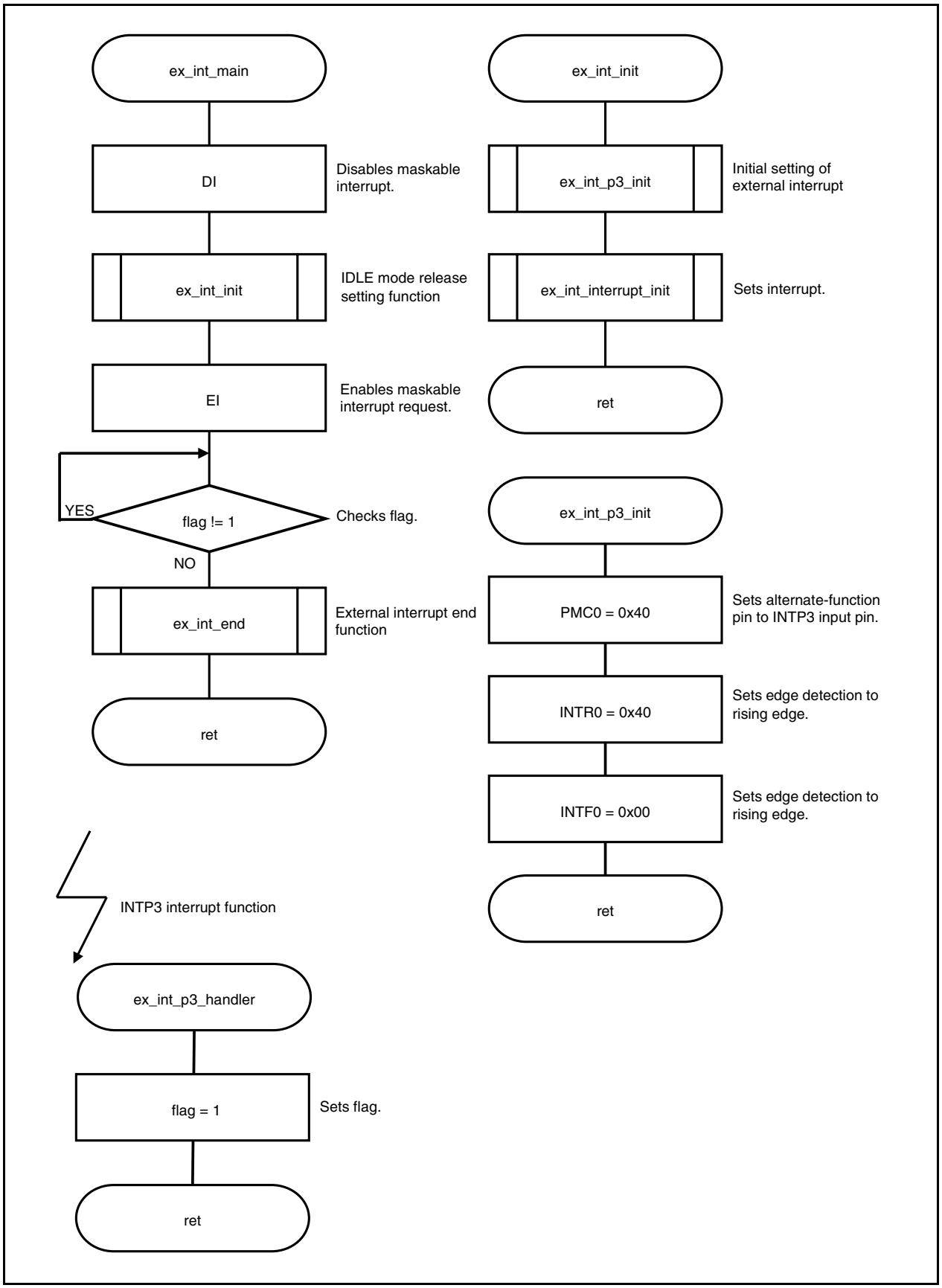
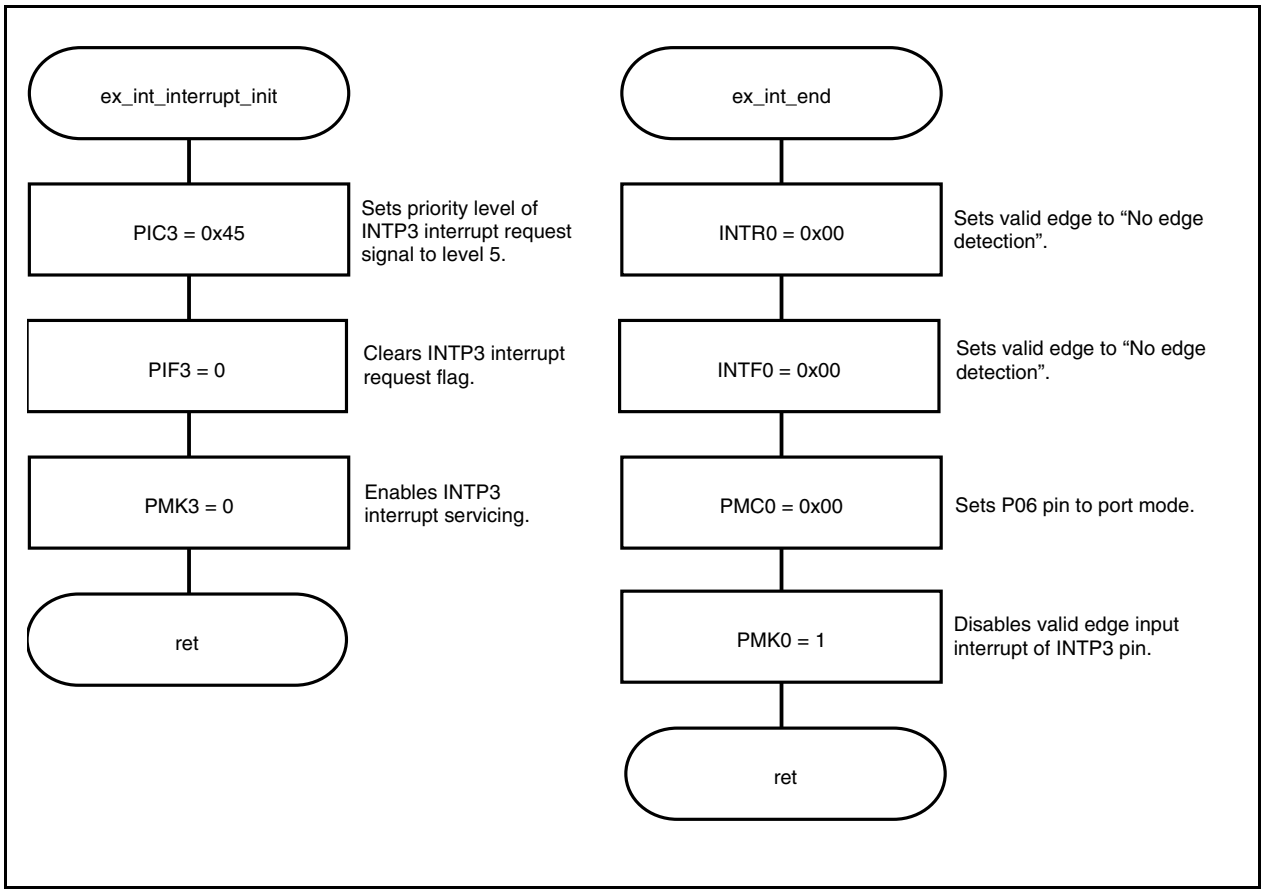


Figure 12-2. External Interrupt (2)



CHAPTER 13 STANDBY FUNCTION

13.1 HALT Mode

[Function]	Standby function (HALT mode)
[Function name]	halt_main
[Argument]	None
[Processing content]	Shifts from normal mode to HALT mode. HALT mode is released by external interrupt request signal.
[Starting method]	None
[SFR used]	None
[call functions]	halt_init, halt_mode
[Variable]	None
[Interrupt]	external_int
[Interrupt source]	INTP3
[File name]	halt\halt.c
[Caution]	None

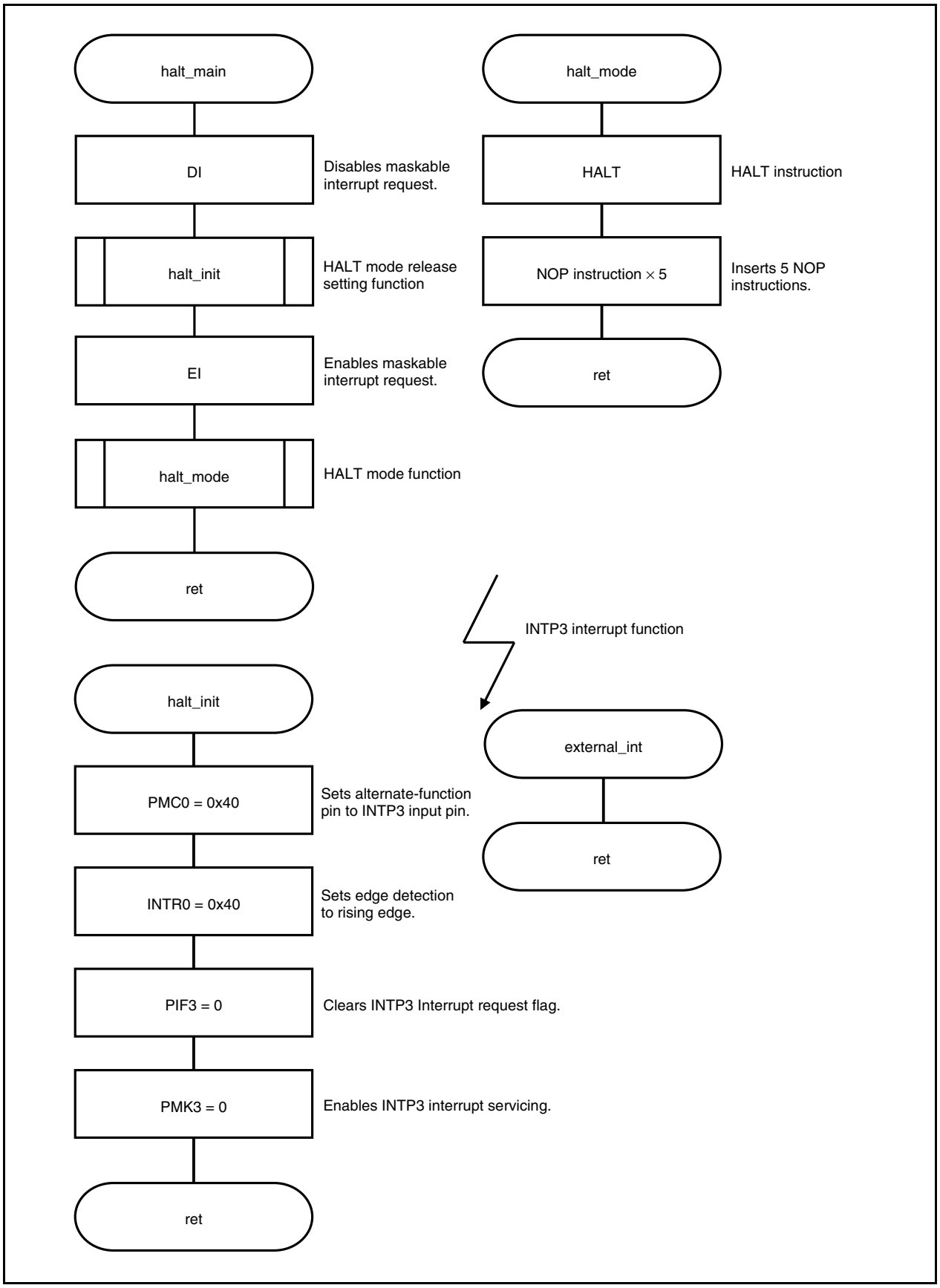
[Function name]	halt_init
[Processing content]	Sets external interrupt request signal (INTP3) to release HALT mode.
[SFRs used]	PMC0: 0x40 (Sets an alternate-function pin.) INTR0: 0x40 (Sets a valid edge to rising edge.) PIC3.PIF3: 0 (Clears INTP3 interrupt request flag.) PIC3.PMK3: 0 (Enables INTP3 interrupt servicing.)
[call function]	None
[Variable]	None
[File name]	halt\halt.c
[Caution]	None

[Function name]	halt_mode
[Processing content]	Executes HALT instruction.
[SFR used]	None
[call function]	None
[Variable]	None
[File name]	halt\halt.c
[Caution]	Insert five or more NOP instructions after the HALT instruction.

Interrupt function

[Function name]	external_int
[Processing content]	Checks occurrence of external interrupt. There is no specific processing content.
[SFR used]	None
[call function]	None
[Variable]	None
[File name]	halt\halt.c
[Caution]	None

Figure 13-1. HALT Mode (1)



13.2 IDLE Mode

[Function]	Standby function (IDLE mode)
[Function name]	idle_main
[Argument]	None
[Processing content]	Shifts from normal operation mode to IDLE mode. IDLE mode is released by external interrupt request signal.
[Starting method]	None
[SFR used]	None
[call functions]	idle_init, idle_mode
[Variable]	None
[Interrupt]	external_int
[Interrupt source]	INTP3
[File name]	idle\idle.c
[Caution]	None

[Function name]	idle_init	
[Processing content]	Sets external interrupt request signal (INTP3) to release IDLE mode.	
[SFRs used]	PMC0:	0x40 (Alternate-function pin setting)
	INTR0:	0x40 (Sets a valid edge to rising edge.)
	PIC3.PIF3:	0 (Clears INTP3 interrupt request flag.)
	PIC3.PMK3:	0 (Enables INTP3 interrupt servicing.)
[call function]	None	
[Variable]	None	
[File name]	idle\idle.c	
[Caution]	None	

[Function name]	idle_mode
[Processing content]	Disables all DMA transfer and shifts to IDLE mode.
[SFRs used]	DCHC0.E00: 0 (Disables DMA0 transfer.) DCHC1.E11: 0 (Disables DMA1 transfer.) DCHC2.E22: 0 (Disables DMA2 transfer.) DCHC3.E33: 0 (Disables DMA3 transfer.) PSMR: 0x00 (Sets to IDLE mode.) PRCMD: 0x02 (Used for a write to the command register and special register.) PSC: 0x02 (Enables standby mode release, sets standby mode.)
[call function]	None
[Variable]	None
[File name]	idle\idle.c
[Caution]	• DMA transfer is disabled in this sample program because DMA transfer must be disabled before executing the data transfer to the specific registers. • PCC and PSC registers are special registers. Data can be written to these registers only in combination of specific sequences.

Interrupt function

[Function name]	external_int
[Processing content]	Checks occurrence of external interrupt. There is no specific processing content.
[call function]	None
[SFR used]	None
[Variable]	None
[File name]	idle\idle.c
[Caution]	None

Figure 13-2. IDLE Mode (1)

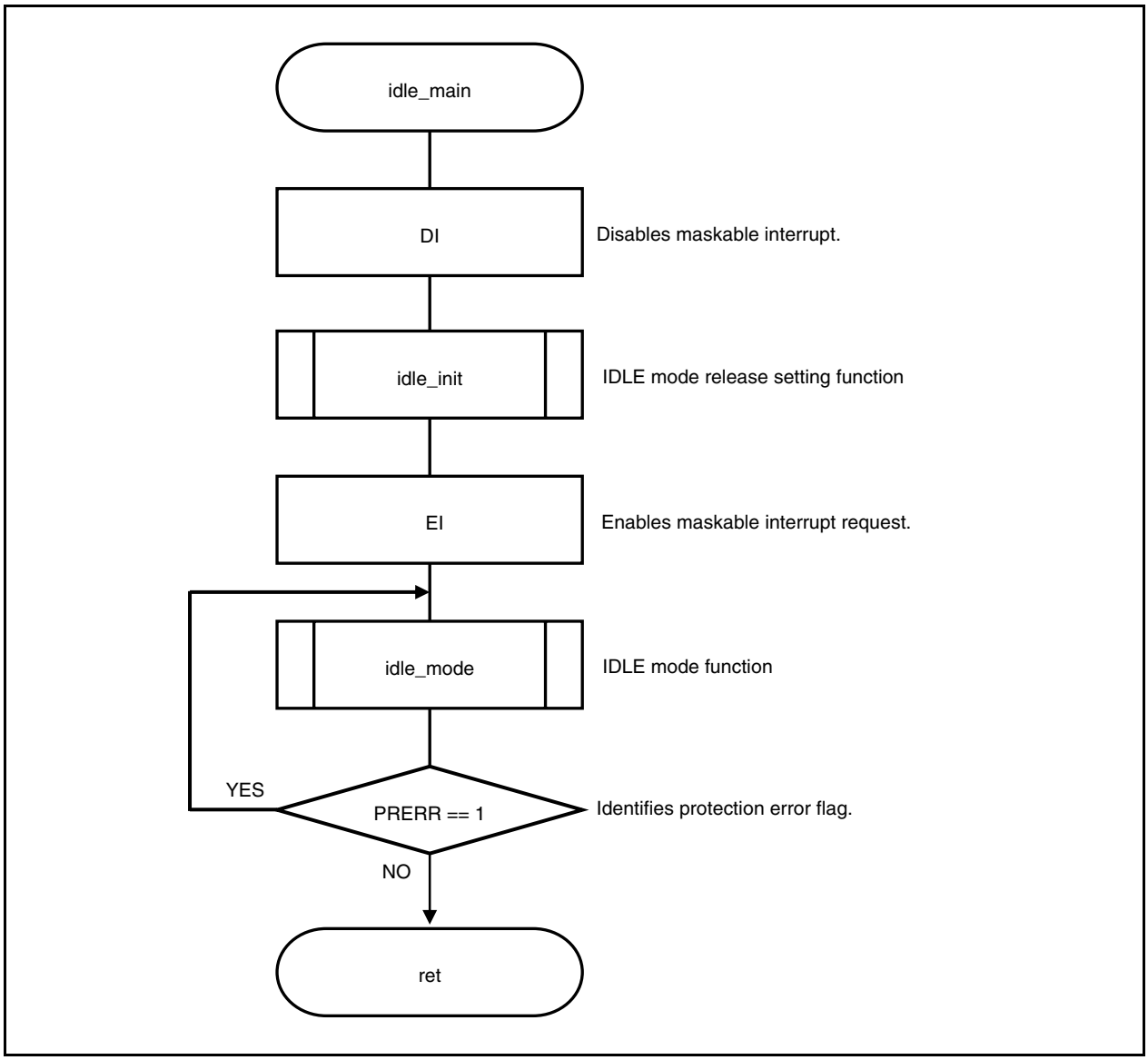


Figure 13-3. IDLE Mode (2)

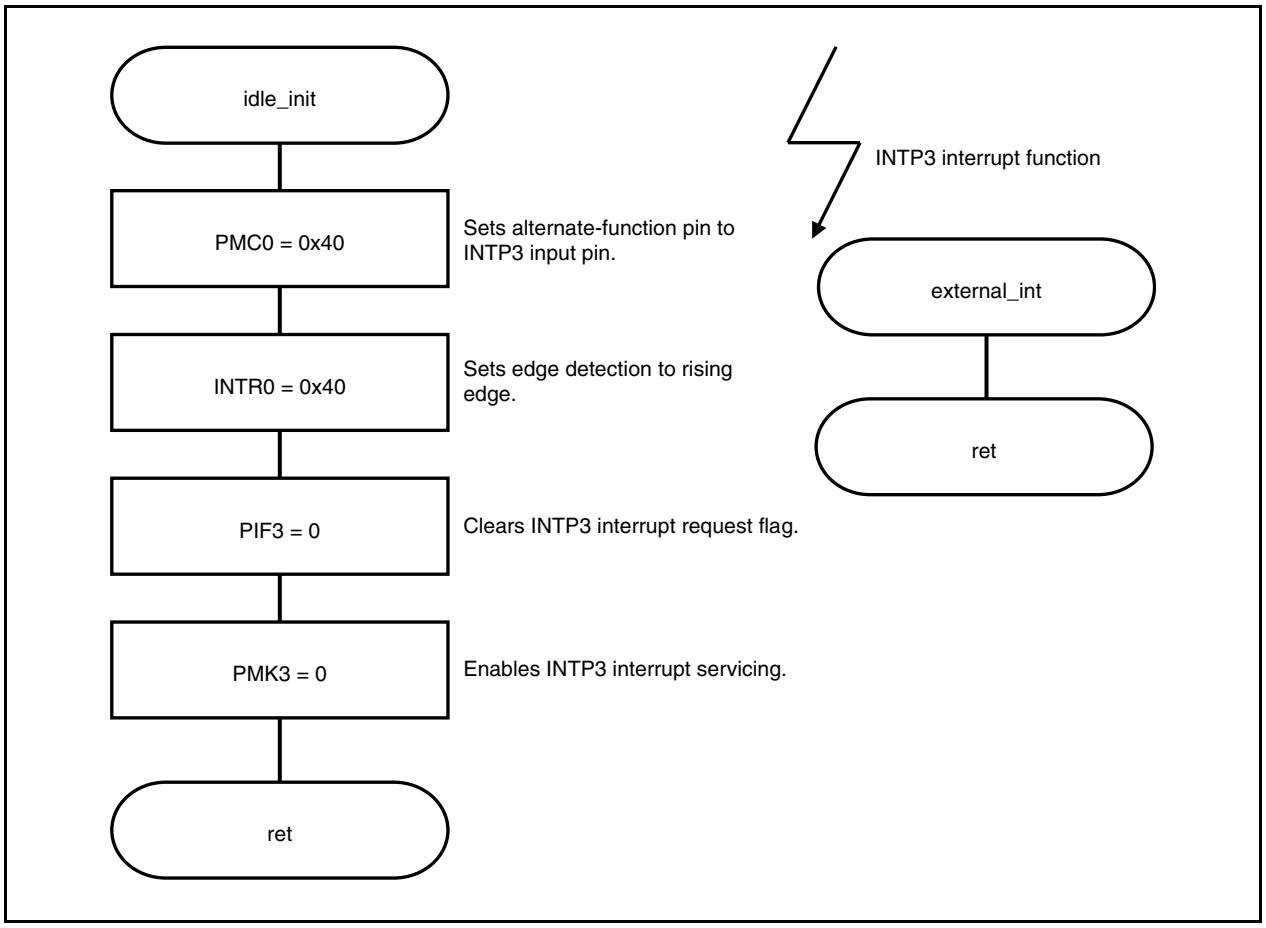
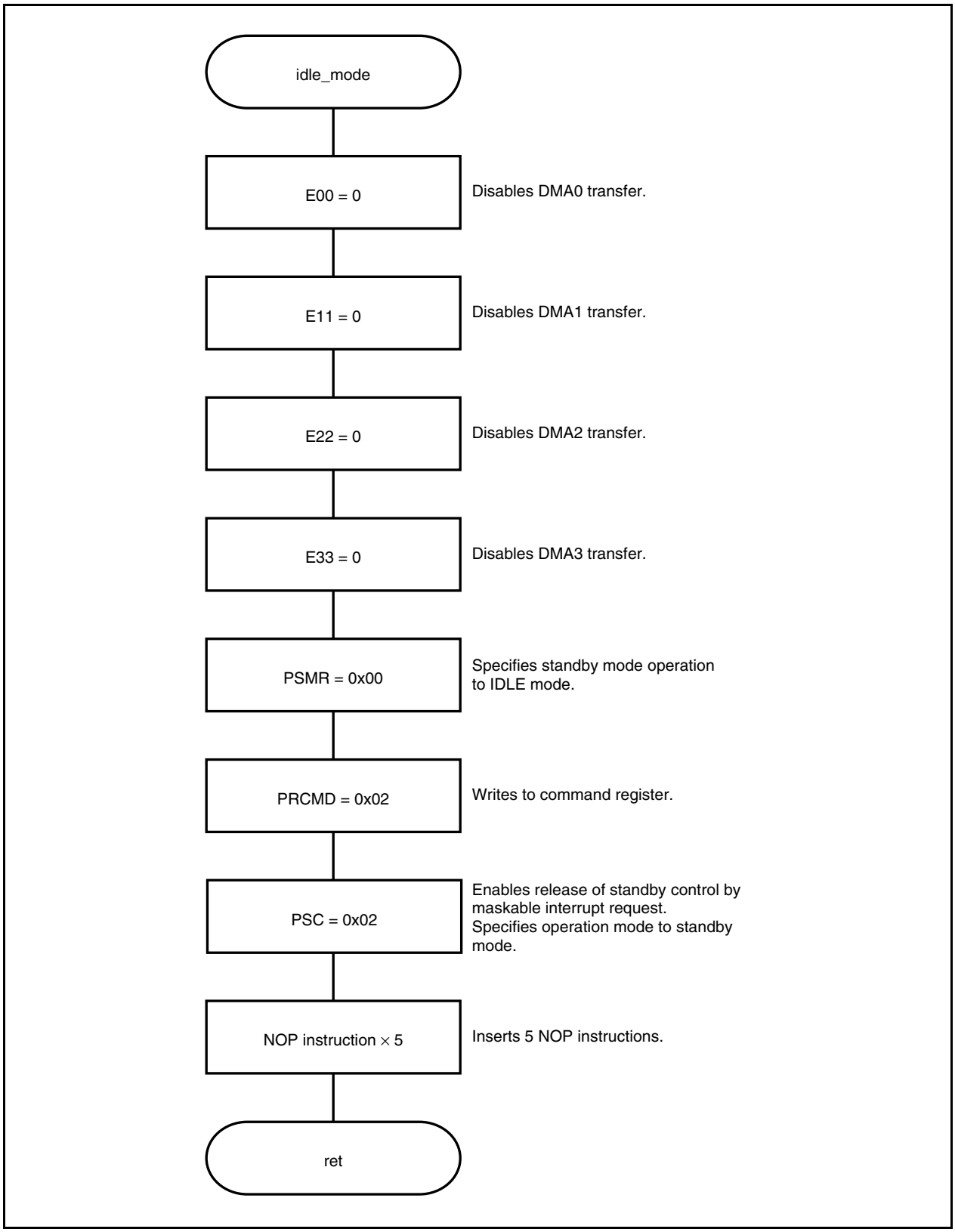


Figure 13-4. IDLE Mode (3)



13.3 STOP Mode

[Function]	Standby function (STOP mode)
[Function name]	stop_main
[Argument]	None
[Processing content]	Shifts from normal operation mode to STOP mode. STOP mode is released by external interrupt request signal.
[Starting method]	None
[SFR used]	None
[call functions]	stop_init, stop_mode
[Variable]	None
[Interrupt]	None
[Interrupt source]	None
[File name]	stop\stop.c
[Caution]	None

[Function name]	stop_init	
[Processing content]	Sets external interrupt request signal (INTP3) to release STOP mode.	
[SFRs used]	OSTS:	0x06 (Oscillation stabilization time 13.107 ms)
	PMC0:	0x40 (Alternate-function pin setting)
	INTR0:	0x40 (Sets a valid edge to rising edge.)
	PIC3.PIF3:	0 (Clears INTP3 interrupt request flag.)
	PIC3.PMK3:	0 (Enables INTP3 interrupt servicing.)
[call function]	None	
[Variable]	None	
[File name]	stop\stop.c	
[Caution]	None	

[Function name]	stop_mode
[Processing content]	Disables all DMA transfers and executes STOP mode.
[SFRs used]	DCHC0.E00: 0 (Disables DMA0 transfer.) DCHC1.E11: 0 (Disables DMA1 transfer.) DCHC2.E22: 0 (Disables DMA2 transfer.) DCHC3.E33: 0 (Disables DMA3 transfer.) PSMR: 0x01 (Sets to STOP mode.) PRCMD 0x02 (Used for a write to command register and specific register.) PSC: 0x02 (Enables standby mode release, sets standby mode.)
[call function]	None
[Variable]	None
[File name]	stop\stop.c
[Caution]	• DMA transfer is disabled in this sample program because DMA transfer must be disabled before executing the data transfer to the specific registers. • PSC register is a special register. Data can be written to this register only in combination of specific sequences. • Sets the PSMR.PSM0 and PSC.STB bits sequentially to set to the STOP mode.

Interrupt function

[Function name]	external_int
[Processing content]	Checks occurrence of external interrupt. There is no specific processing content.
[SFR used]	None
[call function]	None
[Variable]	None
[File name]	stop\stop.c
[Caution]	None

Figure 13-5. STOP Mode (1)

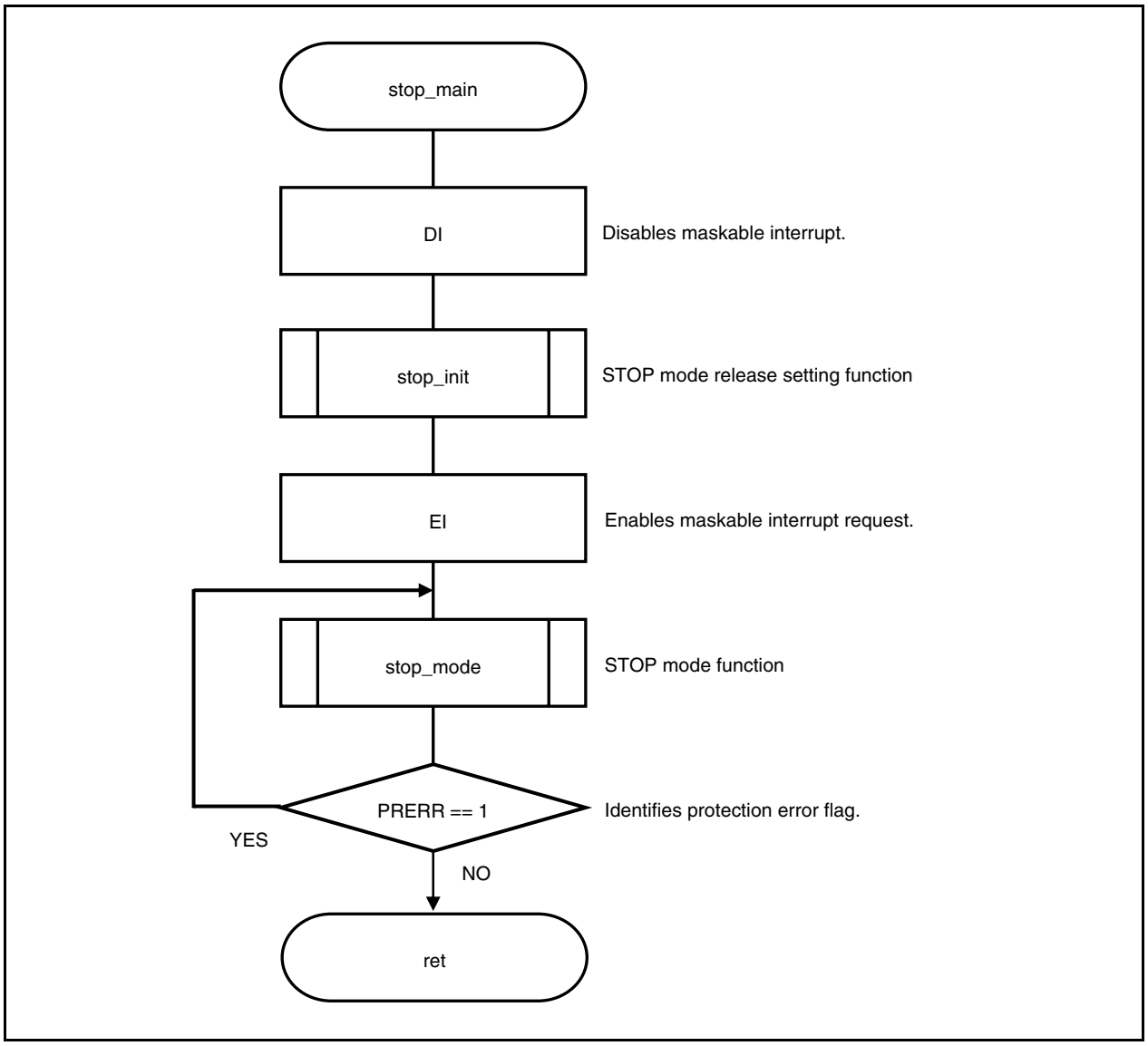
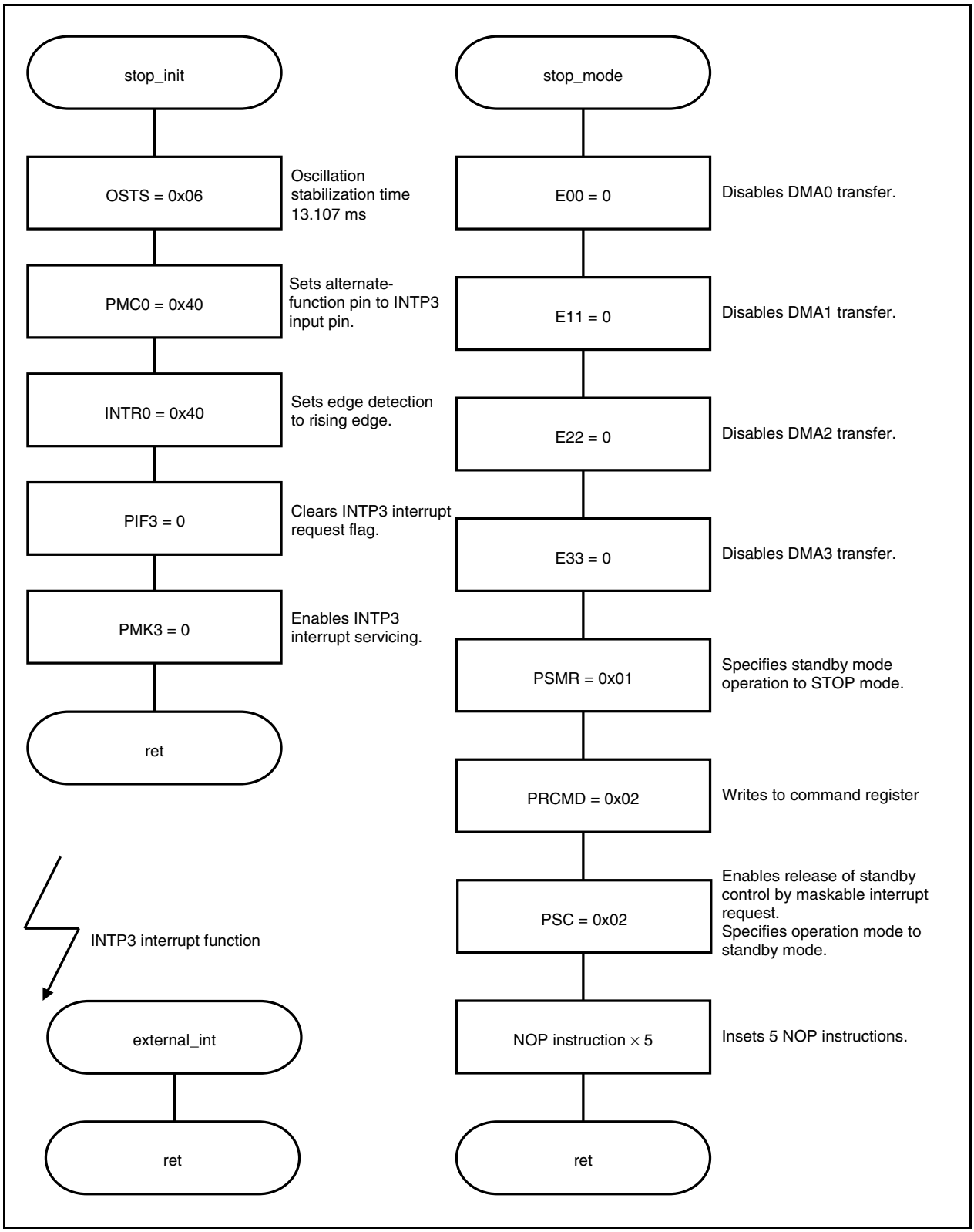


Figure 13-6. STOP Mode (2)

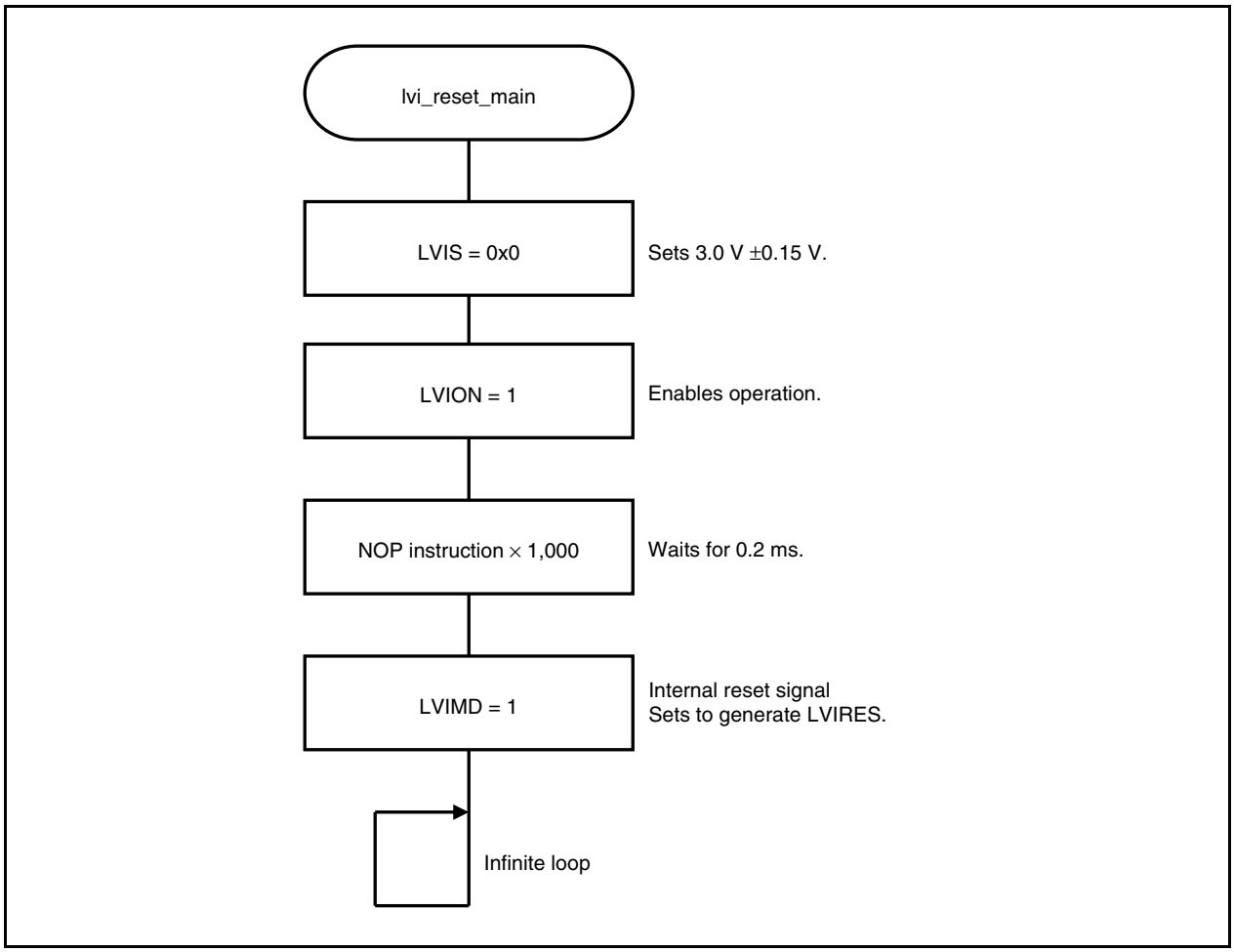


CHAPTER 14 LOW VOLTAGE DETECTOR (LVI)

14.1 Internal Reset Mode

[Function]	Compares the supply voltage (V_{DD}) and detected voltage (V_{LVI}) and generates internal reset signal when $V_{DD} < V_{LVI}$
[Function name]	lvi_reset_main
[Argument]	None
[Processing content]	Sets the LVIS register, LVION register, and LVIMD register, and generates internal reset signal when $V_{DD} < V_{LVI}$
[SFRs used]	LVIS: 0x0 (Sets detection level to 3.0 V ±0.15 V.) LVM.LVION: 1 (Sets to enable operation.) LVIS.LVIMD: 1 (Sets to generate internal reset signal LVIRES at low voltage detection.)
[call function]	None
[Variable]	None
[Interrupt]	None
[Interrupt source]	None
[File name]	lvi_reset\lvi_reset.c
[Caution]	When the LVIMD bit is set to 1, the LVIM and LVIS register settings cannot be changed until a reset request due to other than LVI is generated.

Figure 14-1. Internal Reset Mode



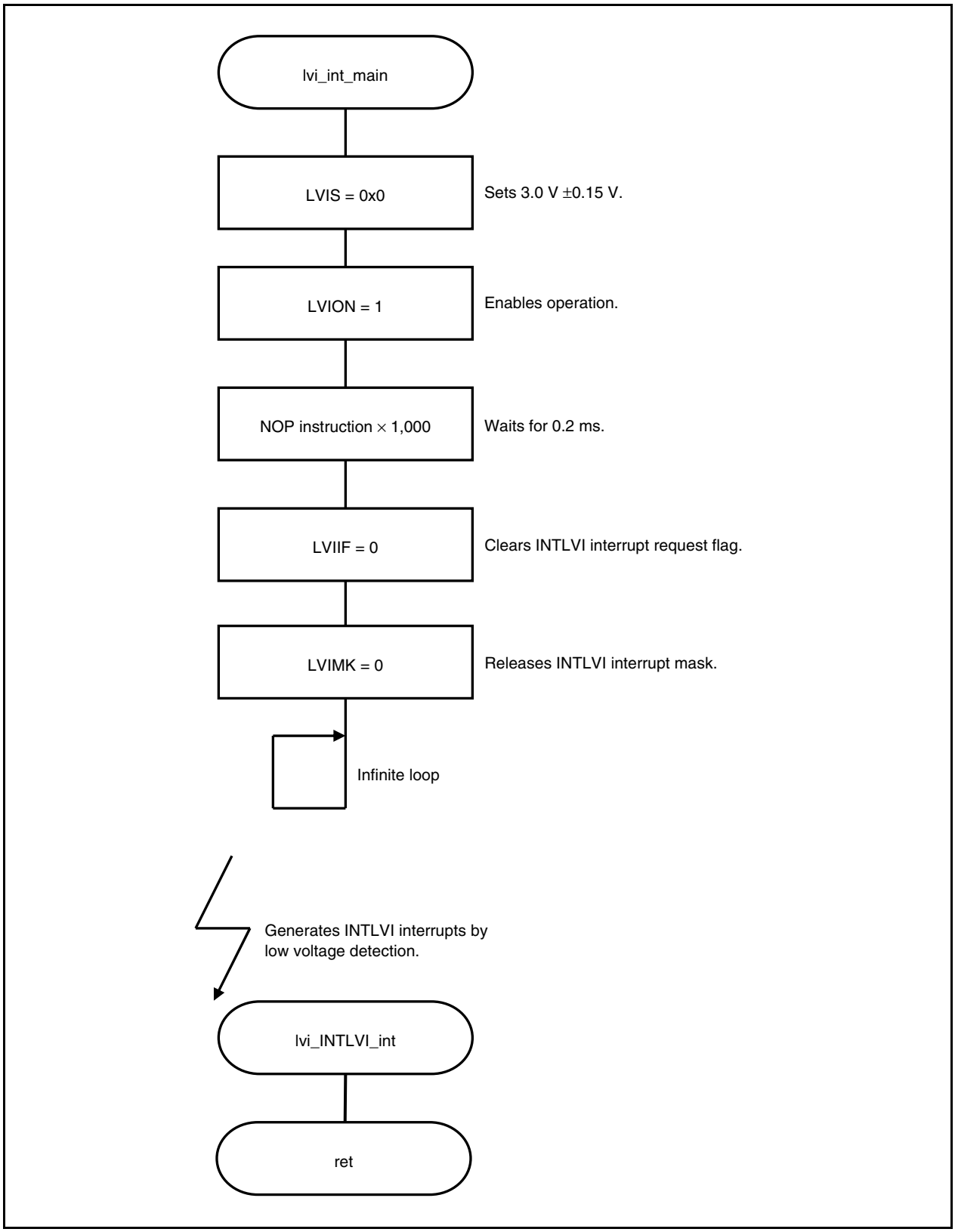
14.2 Internal Interrupt Mode

[Function]	Compares the supply voltage (V_{DD}) and detected voltage (V_{LVI}) and generates an internal interrupt signal (INTLVI) when $V_{DD} < V_{LVI}$.
[Function name]	lvi_int_main
[Argument]	None
[Processing content]	Sets the LVIS register, LVION register, and LVIMD register, and generates an internal reset signal when $V_{DD} < V_{LVI}$.
[SFRs used]	LVIS: 0x0 (Sets detection level to 3.0 V ±0.15 V.) LVIM.LVION: 1 (Sets to enable operation.) LVIIF: 0 (Clears INTLVI interrupt request flag.) LVIMF: 0 (Clears INTLVI interrupt mask.)
[call function]	None
[Variable]	None
[Interrupt]	lvi_INTLVI_int
[Interrupt source]	INTLVI
[File name]	lvi_int\lvi_int.c
[Caution]	None

Interrupt function

[Function name]	lvi_INTLVI_int
[Overview]	Defined by the user.
[Source]	INTLVI Low voltage detection
[call function]	None
[Variable]	None
[File name]	lvi_int\lvi_int.c
[Caution]	None

Figure 14-2. Internal Interrupt Mode



For further information, please contact:

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

Hanover Office
Podbielskistrasse 166 B
30177 Hannover
Tel: 0 511 33 40 2-0

Munich Office
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

Shanghai Branch
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai, P.R.China P.C:200120
Tel:021-5888-5400
http://www.cn.necel.com/

Shenzhen Branch
Unit 01, 39/F, Excellence Times Square Building,
No. 4068 Yi Tian Road, Futian District, Shenzhen,
P.R.China P.C:518048
Tel:0755-8282-9800
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
Unit 1601-1613, 16/F., Tower 2, Grand Century Place,
193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737
http://www.kr.necel.com/

G0706