To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

To all our customers

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

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

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

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

Cautions

Keep safety first in your circuit designs!

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

Notes regarding these materials

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

Application Note

32

SH7144 Series

Application Note

Renesas Electronics
www.renesas.com

Rev.1.0 2003.3

Cautions

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

Contents

Section 1 SH7144 Series Application Note— Application Section Usage Guide

This Application Note consists of two parts, as shown in figure 1.1.

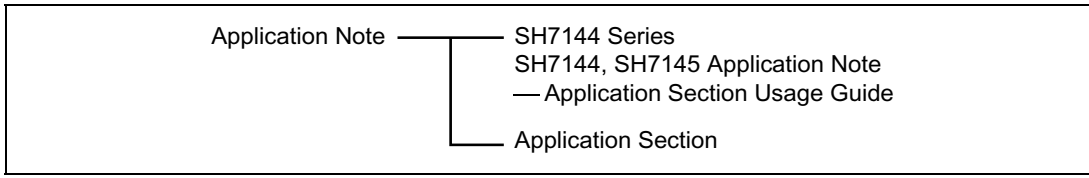


Figure 1.1 Organization of Application Note

(1) SH7144 Series—SH7144, SH7145 Application Note—Application Section Usage Guide

This section explains how to use the SH7144 Series—SH7144, SH7145 Application Note—Application Section

(2) Application Section

The use of a combination of SH7144 and SH7145 on-chip peripheral functions (timers, serial communication interface, A/D converter, PWM, I/O ports, interrupts, power-down mode, etc.) is explained based on simple sample tasks.

1.1 Organization of Application Section

The layout shown in figure 1.2 is used to describe the combined use of on-chip peripheral functions.

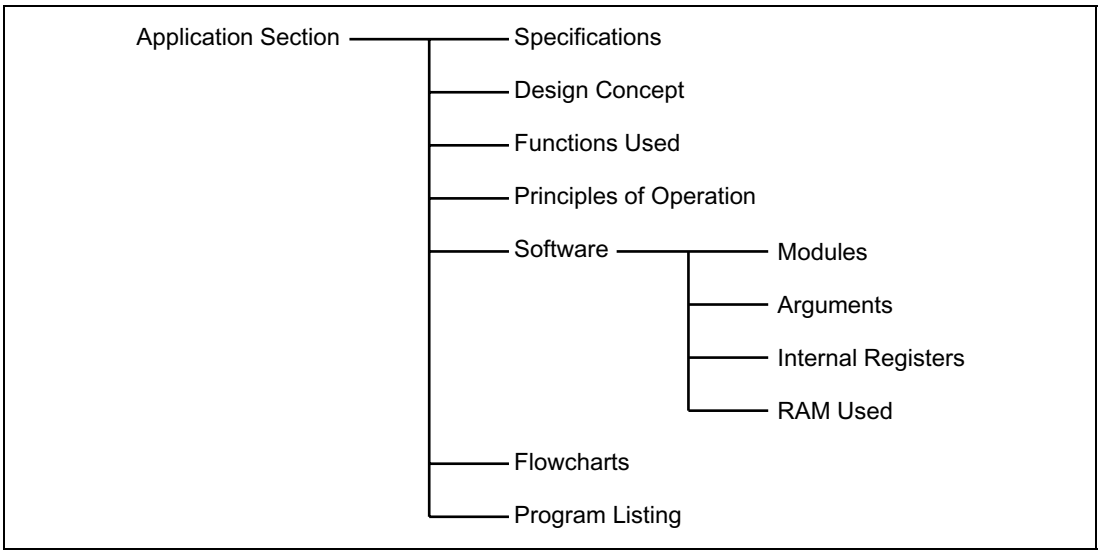


Figure 1.2 Organization of Application Section

(1) Specifications

Describes the system specifications of the sample task.

(2) Design Concept

Describes the method used to implement the sample task system.

(3) Functions Used

Describes the features of the peripheral function(s) used in the sample task, and peripheral function assignments.

(4) Principles of Operation

Describes the operation of the sample task, using timing charts.

(5) Software

(a) Modules

Describes the software modules used in the operation of the sample task.

(b) Arguments

Describes the input arguments needed to execute the modules, and the output arguments after execution.

(c) Internal Registers

Describes the peripheral function internal registers (timer control registers, serial mode registers, etc.) set by the modules.

(d) RAM Used

Describes the labels and functions of the RAM used by the modules.

(6) Flowcharts

Describes the software that executes the sample task, using flowcharts.

(7) Program Listing

Shows a program listing of the software that executes the sample task.

2.1 Pulse High and Low Width Measurement

Pulse High and Low Width Measurement	MCU: SH7144/45	Functions Used: MTU (Input Capture)

Specifications

(1) Pulse high width and low width times are measured and the results are stored in RAM as shown in figure 2.1.

(2) When operating with on-chip peripheral clock Pϕ = 40.0 MHz, the pulse high width and low width can be measured in a range of 25.0 ns to 1.63 ms in 25.0 ns units.

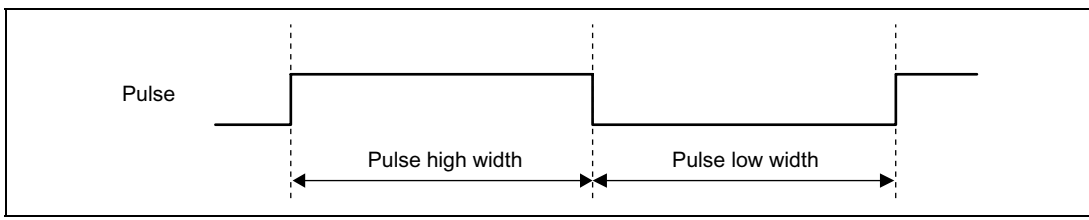


Figure 2.1 Pulse Width Measurement Timing

Functions Used

(1) In this sample task, the high width and low width of a pulse are measured using channel 0 (ch0).

(a) Figure 2.2 shows a block diagram of ch0. This task uses the following functions.

- A function that performs pulse rising edge and falling edge detection, and sets the timer value at that time in an internal register (input capture)
- A function that clears the timer counter when input capture occurs (counter clearing)
- A function that initiates interrupt handling when a pulse rising edge or falling edge is detected

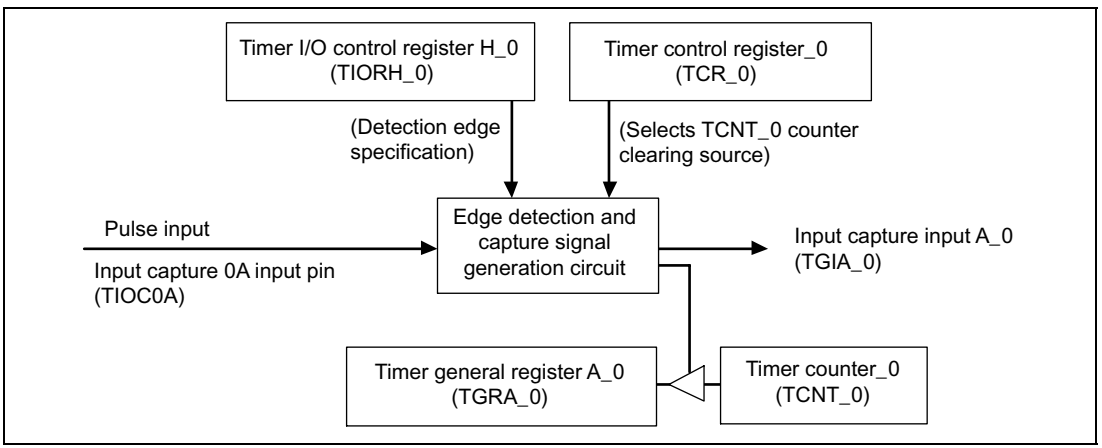


Figure 2.2 Block Diagram of MTU/ch0

(2) Table 2.1 shows the function assignments used in this sample task. The high width and low width of a pulse are measured by assigning MTU functions as shown in the table.

Table 2.1 Function Assignments

Pin or Register Name	Function Assignment
TCR_0	Counter clearing source selection
TIORH_0	Selects input edge of input capture signal
TIOC0A	Inputs pulse to be measured
TGRA_0	Detection of counter value at pulse rising edge or falling edge
TGIA_0	Initiates pulse high and low width measurement at pulse rising edge or falling edge

Principles of Operation

(1) Figure 2.3 illustrates the principles of operation of this sample task. Pulse high width and low width measurement is performed by SH7145 hardware and software processing as shown in the figure.

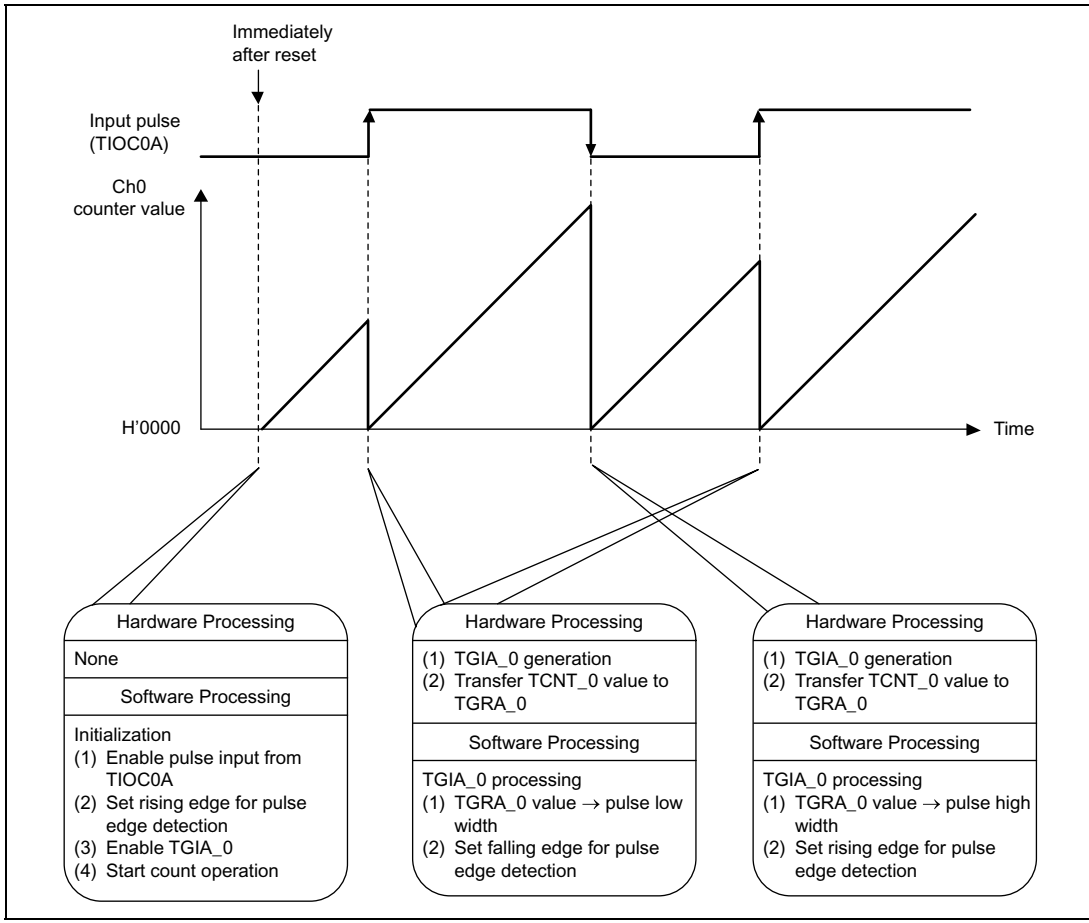


Figure 2.3 Principles of Operation of Pulse Width Measurement

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	pwhlmn	MTU initialization
Pulse high width and low width measurement	pwhl1	Initiated by TGIA_0. Measures pulse high width and low width based on TGRA_0 value, and stores results in RAM

(2) Arguments

Label or Register Name	Function Assignment	Data Length	Module	Input/ Output
pwh_hdata	Used to set timer value for pulse high width Pulse high width is calculated using following equation: Pulse high width (ns) = timer value × ϕ period (25.0 ns at 40.0 MHz operation)	1 word	Pulse high width and low width measurement	Output
pwh_ldata	Used to set timer value for pulse low width Pulse low width is calculated using following equation: Pulse low width (ns) = timer value × ϕ period (25.0 ns at 40.0 MHz operation)	1 word		

(3) Internal Registers Used

Register Name	Function	Address	Set Value
P_PORTE.PECRL2	Sets PE0 as TIOC0A input pin	H'FFFF83BA	H'0001
P_MTU0.TCR_0	TCNT_0 counter clock selection, and setting of TCNT_0 clearing by TGRA_0 input capture as counter clearing source	H'FFFF8260	H'20
P_MTU0.TIORH_0	Sets transfer of TCNT_0 value to TGRA_0 on detection of pulse rising edge or falling edge	H'FFFF8262	H'08
P_MTU0.TIER_0	Enables interrupt request by TGIA_0	H'FFFF8264	H'41
P_MTU0.TGRA_0	TCNT_0 values at time of pulse rising edge and falling edge are stored, and pulse period is calculated from these values	H'FFFF8268	pwh_ldata pwh_hdata
P_INTC.IPRD	Sets 15 as TGIA_0 interrupt priority level	H'FFFF834E	H'f000
P_STBY.MSTCR2	MTU module standby mode clearing	H'FFFF861E	H'd0fd

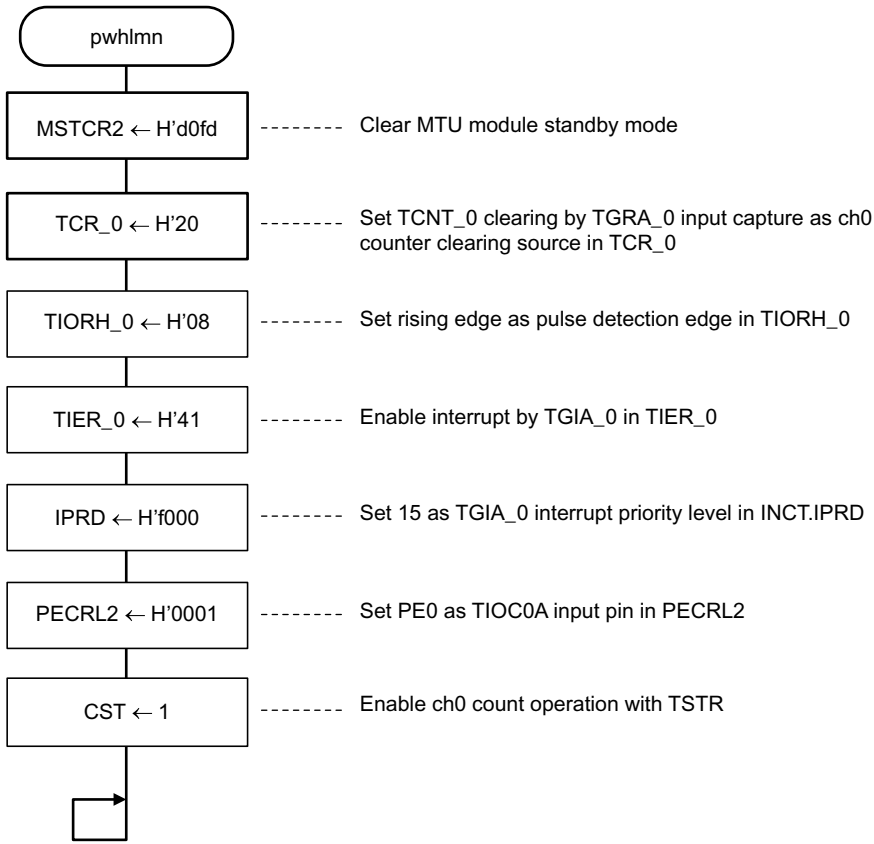
(4) RAM Used

This sample task does not use any RAM apart from the arguments.

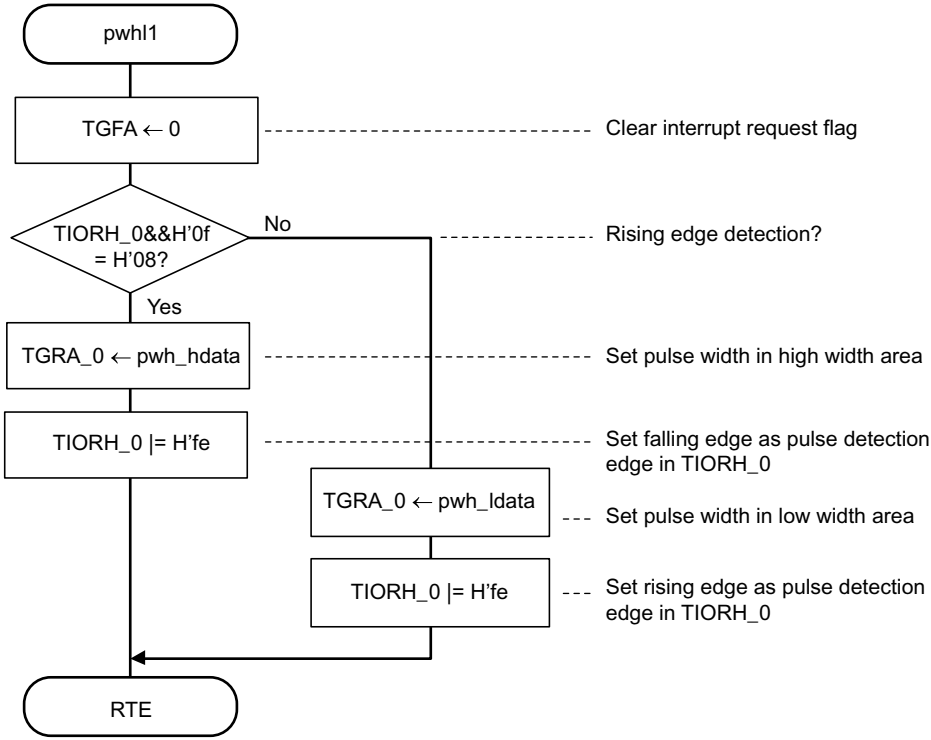
Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



(2) Pulse high and low width measurement



Program Listing

```
/*****************************************************************************/
/*                                INCLUDE FILE                               */
/*****************************************************************************/
#include <machine.h>
#include "iodefine_7145F.h"
/*****************************************************************************/
/*                                 PROTOTYPE                                 */
/*****************************************************************************/
void  pwhlmn(void);
#pragma interrupt(pwhl1)
/*****************************************************************************/
/*                               RAM ALLOCATION                              */
/*****************************************************************************/
#define pwh_hdata   (*(unsigned short  *)0xffffe000)
#define pwh_ldata   (*(unsigned short  *)0xffffe002)
/*****************************************************************************/
/*                               MAIN PROGRAM                                */
/*****************************************************************************/
void pwhlmn(void)
 {
      set_imask(0xf);
      P_STBY.MSTCR2.WORD = 0xd0fd;
      P_MTU0.TCR_0.BYTE = 0x20;        /* timer clear input capture with TGRA_0 */
                                       /* counter clock = Pϕ/1 */
      P_MTU0.TIORH_0.BYTE = 0x08;      /* input capture by TIOC0A rising edge */
      P_MTU0.TIER_0.BYTE = 0x41;       /* enable TGIA interrupt */
      P_INTC.IPRD.WORD = 0xf000;       /* set initialize level = 15 */
      P_PORTE.PECRL2.WORD = 0x0001;
      P_MTU34.TSTR.BIT.CST = 1;        /* start TCNT_0 */
      set_imask(0x0);
      while(1);
  }

void pwhl1()
 {
     P_MTU0.TSR_0.BIT.TGFA = 0;           /* clear interrupt flag */
     if((P_MTU0.TIORH_0.BYTE & 0x0f) == 0x08)
      {
         pwh_hdata = P_MTU0.TGRA_0.BYTE; /* set pwh */
         P_MTU0.TIORH_0 |= 0x01;         /* input capture falling edge TIOC0A */
      }
     else
      {
        pwh_ldata = P_MTU0.TGRA_0.BYTE;  /* set pwl */
        P_MTU0.TIORH_0 |= 0xfe;          /* input capture rising edge TIOC0A */
       }
    }
```

2.2 Pulse Output

Pulse Output	MCU: SH7144/45	Functions Used: MTU (Output Compare)

Specifications

(1) Using MTU ch0, a 50% duty pulse with a period set in RAM is output as shown in figure 2.4.

(2) When operating with on-chip peripheral clock Pϕ = 40.0 MHz, the output pulse width can be set arbitrarily in the range 50.0 ns to 1.63 ms.

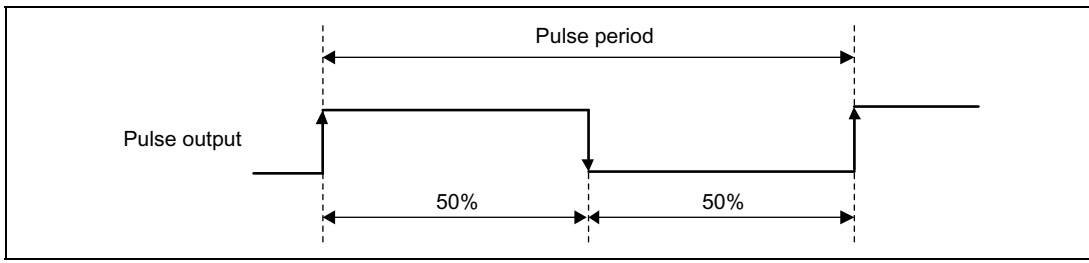


Figure 2.4 Pulse Output

Functions Used

(1) In this sample task, a pulse with a 50% duty cycle is output using MTU channel 0 (ch0).

(a) Figure 2.5 shows a block diagram of MTU/ch0 as used in this task.

The following ch0 functions are used.

- A function that outputs pulses automatically by hardware without software intervention (output compare)
- A function that clears a counter when a compare match occurs (counter clearing)
- A function that reverses output each time a compare match occurs (toggle output)

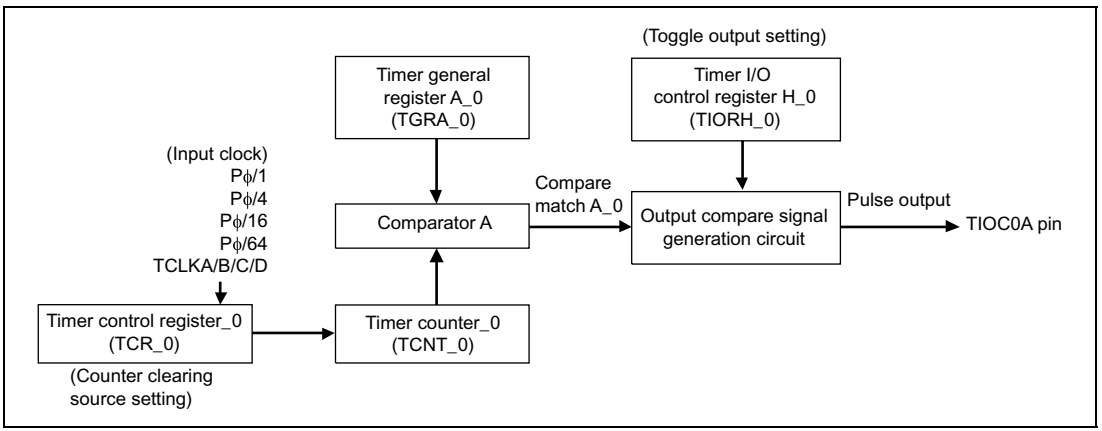


Figure 2.5 Block Diagram of MTU/ch0

(2) Table 2.2 shows the function assignments used in this sample task. Pulses are output by assigning MTU functions as shown in the table.

Table 2.2 Function Assignments

Pin or Register Name	Function Assignment
TIOC0A	Pulse output pin
TCR_0	Selection of counter clearing source and input clock
TIORH_0	Pulse output level setting
TGRA_0	Pulse 1/2 period setting

Principles of Operation

(1) Figure 2.6 illustrates the principles of operation of this sample task. Pulses are output by SH7145 hardware and software processing as shown in the figure.

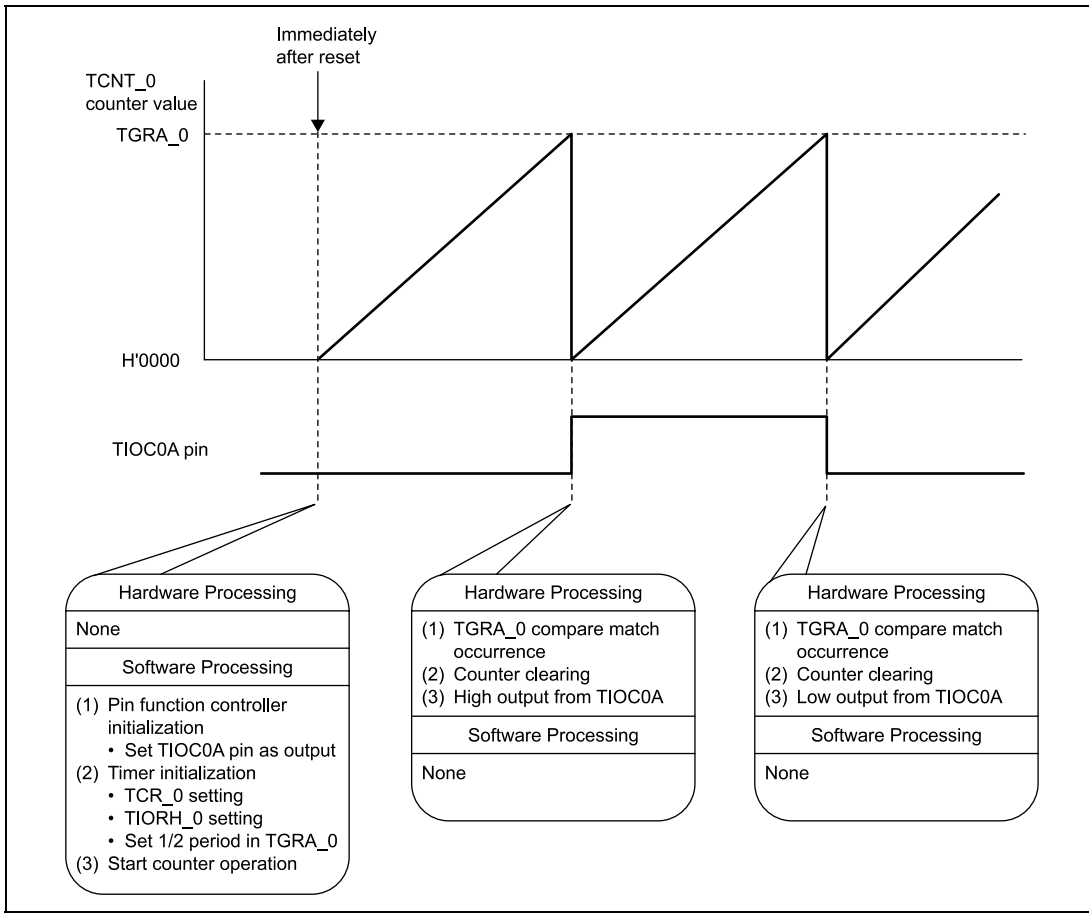


Figure 2.6 Principles of Operation of Pulse Output

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	puls_out	PFC and pulse output setting

(2) Arguments

Label or Register Name	Function Assignment	Data Length	Module	Input/ Output
pul_cyc	Used to set timer value for pulse 1/2 period Pulse period is calculated using following equation: Pulse period (ns) = timer value × ϕ period (25 ns at 40.0 MHz)	1 word	Main routine	Input

(3) Internal Registers Used

Register Name	Function	Address	Set Value
P_PORTE.PECRL2	Sets PE0 as TIOC0A output	H'FFFF83BA	H'0001
P_MTU0.TCR_0	Sets TGRA_0 compare match as counter clearing source Sets Pϕ/1 as input clock	H'FFFF8260	H'20
P_MTU0.TIORH_0	TIOC0A initial output 0, output toggled on compare match	H'FFFF8262	H'03
P_MTU0.TGRA_0	Output pulse 1/2 period setting	H'FFFF8268	pul_cyc
P_MTU0.TMDR_0	Sets ch0 to normal mode	H'FFFF8261	H'c0
P_STBY.MSTCR2	MTU module standby mode clearing	H'FFFF861E	H'd0fd

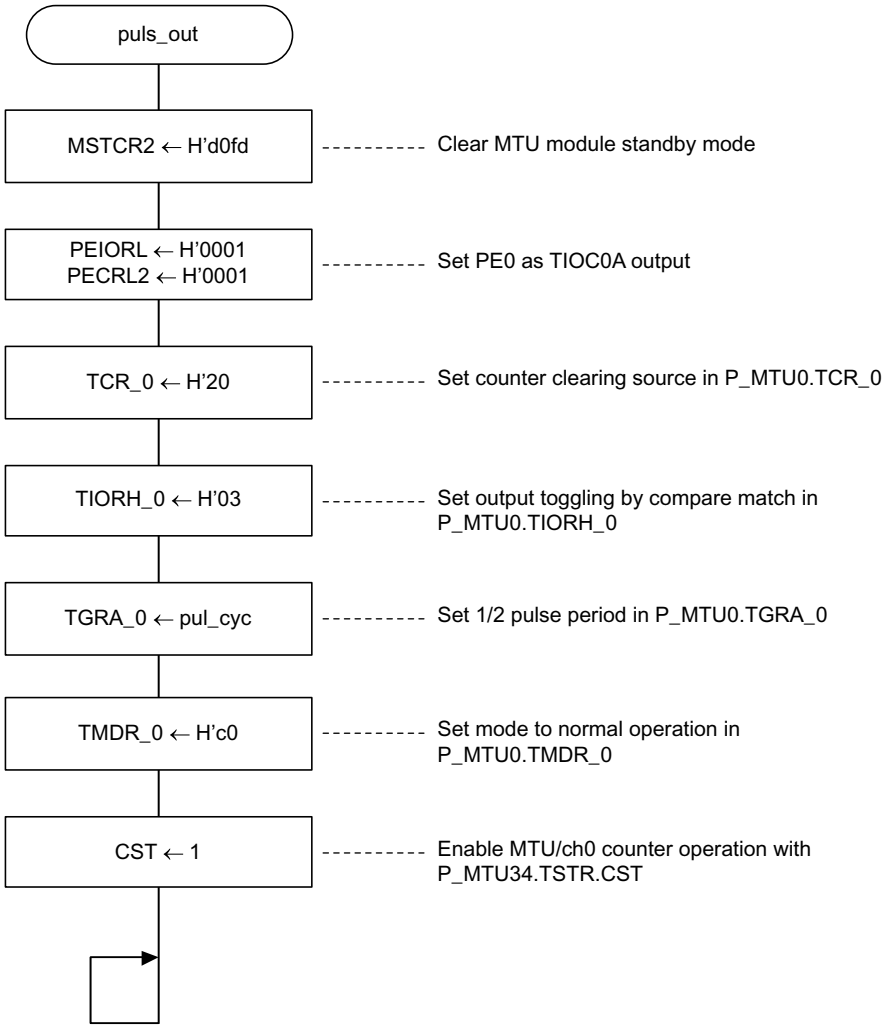
(4) RAM Used

This sample task does not use any RAM apart from the arguments.

Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



Program Listing

```
/************************************************************************/
/*                                INCLUDE FILE                          */
/************************************************************************/
#include<machine.h>
#include"iodefine_7145F.h"
/************************************************************************/
/*                                  PROTOTYPE                           */
/************************************************************************/
void puls_out(void);
/************************************************************************/
/*                                RAM ALLOCATION                        */
/************************************************************************/
#define  pul_cyc   (*(unsigned short *)0xffffe000)
/************************************************************************/
/*                                MAIN PROGRAM                          */
/************************************************************************/
void puls_out(void)
 {
  P_STBY.MSTCR2.WORD = 0xd0fd;       /* MTU module standby mode clear */
  P_PORTE.PEIORL.WORD = 0x0001;      /* TIOC0A = output */
  P_PORTE.PECRL2.WORD = 0x0001;      /* PE0 function = TIOC0A */

  P_MTU0.TCR_0.BYTE = 0x20;          /* Counter clear by TGRA_0 */
  P_MTU0.TIORH_0.BYTE = 0x03;        /* toggle output */
  P_MTU0.TGRA_0 = pul_cyc;           /* set 1/2 period */
  P_MTU0.TCNT_0 = 0x0000;            /* Clear timer counter */
  P_MTU0.TMDR_0.BYTE = 0xc0;         /* Set operation mode */
  P_MTU34.TSTR.CST.BIT = 1;          /* Start timer counter */
  while(1);
 }
```

2.3 PWM 4-Phase Output

PWM 4-Phase Output	MCU: SH7144/45	Functions Used: MTU (PWM Mode 1)

Specifications

(1) Using MTU PWM mode 1, 4-phase PWM output is performed based on a set duty cycle and period.

(2) In PWM mode 1, an arbitrary period can be set for each channel. Two outputs are possible for each of ch0, ch3, and ch4, and one output for each of ch1 and ch2. Thus for ch0, ch3, and ch4, waveforms can be generated with a different high width within the same period.

(3) A duty cycle of 0% to 100% can be set with a 1/65,535 resolution.

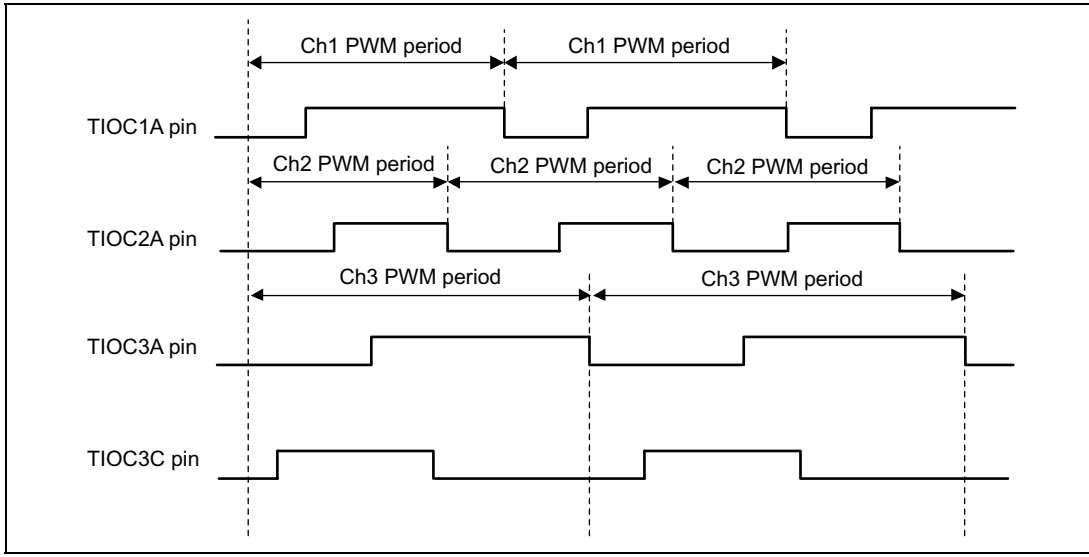


Figure 2.7 Example of PWM Output

Functions Used

(1) In this sample task, 4-phase PWM output is performed using MTU ch1 to ch3.

In PWM mode 1, PWM output is generated with TGRA paired with TGRB, and TGRC paired with TGRD. By using ch0 to ch4, a maximum of 8-phase PWM output is possible.

(a) Figure 2.8 shows a block diagram of the MTU as used in this sample task.

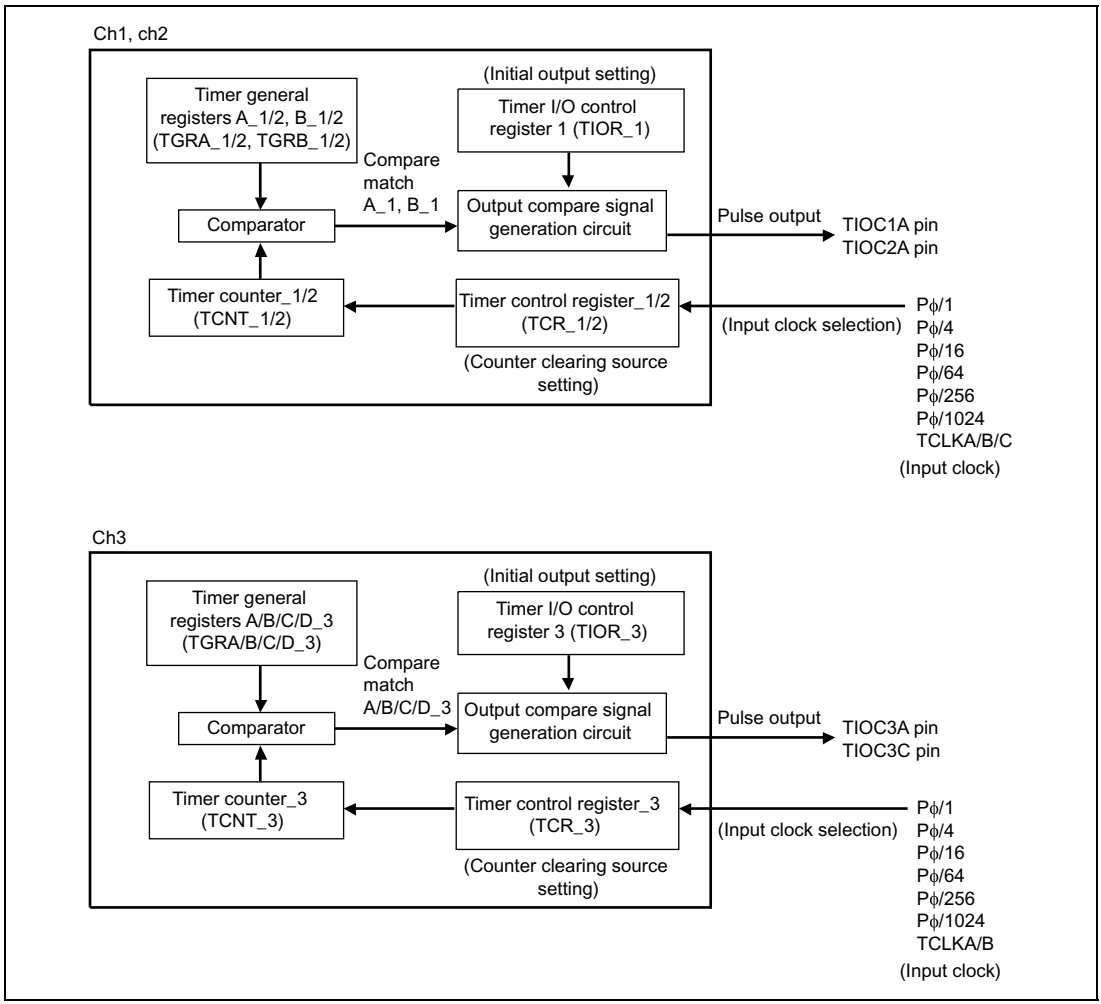


Figure 2.8 Block Diagram of MTU/ch1, ch2, ch3

(2) Table 2.3 shows the function assignments used in this sample task. PWM pulses are output by assigning MTU functions as shown in the table.

Table 2.3 Function Assignments

Pin or Register Name	Function Assignment
TIOC1A TIOC2A TIOC3A TIOC3C	PWM pulse output pins
TCR_1 TCR_2 TCR_3	Selection of ch1 to ch3 timer counter clearing sources and input clocks
TMDR_1 TMDR_2 TMDR_3	Operation of ch1 to ch3 in PWM mode 1
TGRA_1 TGRA_2 TGRA_3	PWM period setting
TGRB_1 TGRB_2 TGRB_3 TGRC_3 TGRD_3	Duty cycle setting

Principles of Operation

(1) Figure 2.9 illustrates the principles of operation of this sample task. Four-phase PWM output is performed from the ch1 to ch3 PWM output pins (TIOC1A, TIOC2A, TIOC3A/C) by SH7145 hardware and software processing as shown in the figure.

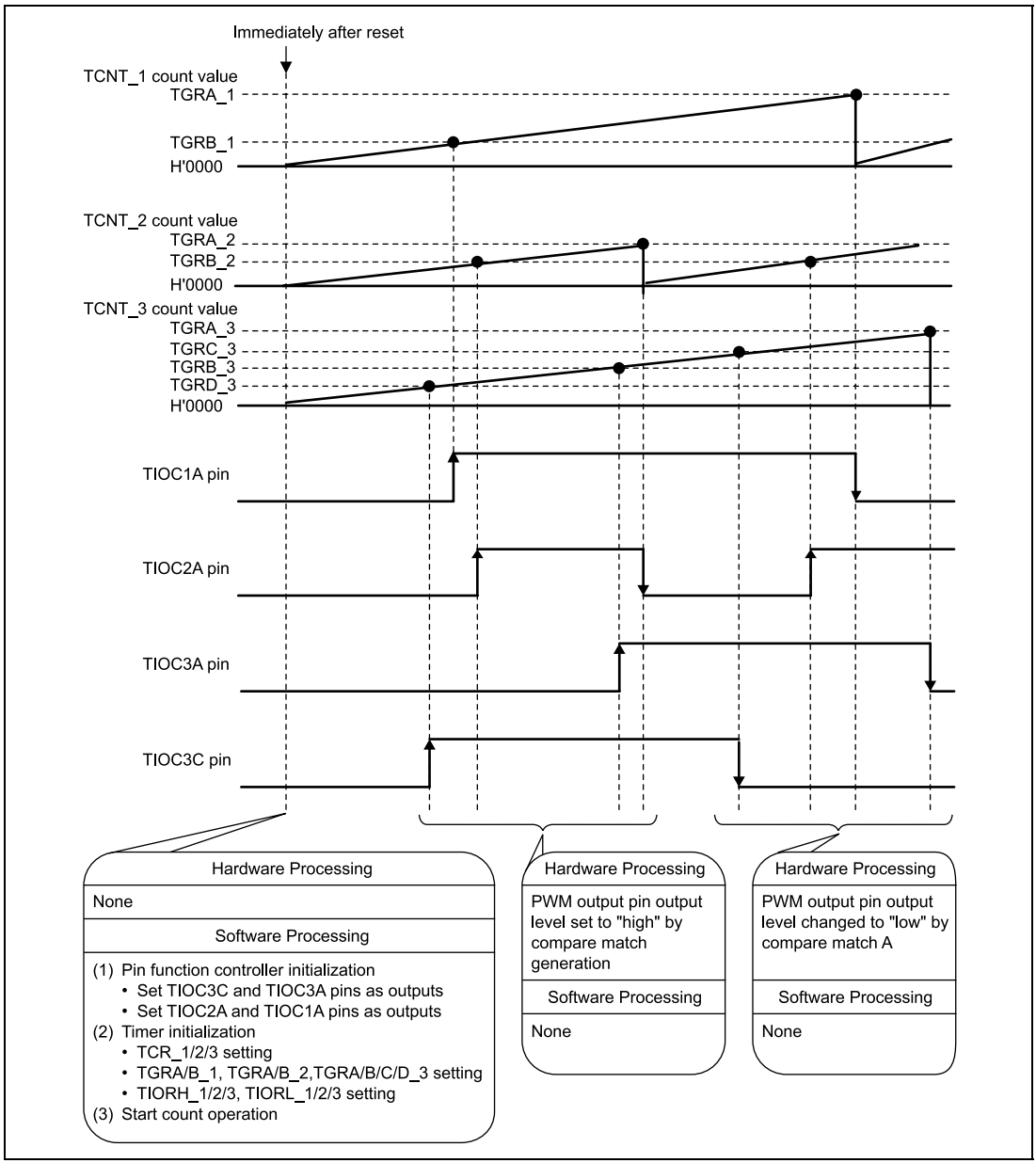


Figure 2.9 Principles of Operation of PWM Waveforms

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	pwm_1	PFC and PWM output setting

(2) Arguments

Label or Register Name	Function Assignment	Data Length	Module	Input/ Output
pul_cyc1 pul_cyc2 pul_cyc3	Used to set timer value for pulse period Pulse period is calculated using following equation: Pulse period (ns) = timer value × ϕ period (25 ns at 40 MHz operation)	1 word	Main routine	Input
pul_duty1b pul_duty2b pul_duty3b pul_duty3c pul_duty3d	Used to set TIOC pin output waveform transition timing			

(3) Internal Registers Used

Register Name	Function Assignment	Address	Set Value
P_STBY.MSTCR2	MTU module standby mode clearing	H'FFFF861E	H'd0fd
P_MTU1.TCR_1 P_MTU2.TCR_2 P_MTU3.TCR_3	Timer counter clearing sources cleared by TGRA_1, TGRA_2, TGRA_3 compare matches Pϕ/16 selected as input clock	H'FFFF8280 H'FFFF82A0 H'FFFF8200	H'22 H'22 H'22
P_MTU1.TGRA_1	Channel 1 PWM period setting	H'FFFF8288	pul_cyc1
P_MTU1.TGRB_1	Used to set timer counter value causing high output from TIOC1A	H'FFFF828A	pul_duty1b
P_MTU2.TGRA_2	Channel 2 PWM period setting	H'FFFF82A8	pul_cyc2
P_MTU2.TGRB_2	Used to set timer counter value causing high output from TIOC2A	H'FFFF82AA	pul_duty2b
P_MTU34.TGRA_3	Channel 3 PWM period setting	H'FFFF8218	pul_cyc3
P_MTU34.TGRB_3	Used to set timer counter value causing high output from TIOC3A	H'FFFF821A	pul_duty3b

Register Name	Function Assignment	Address	Set Value
P_MTU34.TGRC_3	Used to set timer counter value causing low output from TIOC3C	H'FFFF8224	pul_duty3c
P_MTU34.TGRD_3	Used to set timer counter value causing high output from TIOC3C	H'FFFF8226	pul_duty3d
P_MTU1.TIOR_1	Sets TGRA_1 initial output 0, 0 output on output compare, TGRB_1 initial output 0, 1 output on output compare	H'FFFF8282	H'21
P_MTU2.TIOR_2	Sets TGRA_2 initial output 0, 0 output on output compare, TGRB_2 initial output 0, 1 output on output compare	H'FFFF82A2	H'21
P_MTU34.TIORH_3	Sets TGRA_3 initial output 0, 0 output on output compare, TGRB_3 initial output 0, 1 output on output compare	H'FFFF8204	H'21
P_MTU34.TIORL_3	Sets TGRC_3 initial output 0, 0 output on output compare, TGRD_3 initial output 0, 1 output on output compare	H'FFFF8205	H'21
P_MTU1.TMDR_1 P_MTU2.TMDR_2 P_MTU34.TMDR_3	Set PWM mode 1 as operating mode	H'FFFF8281 H'FFFF82A1 H'FFFF8202	H'c2 H'c2 H'c2

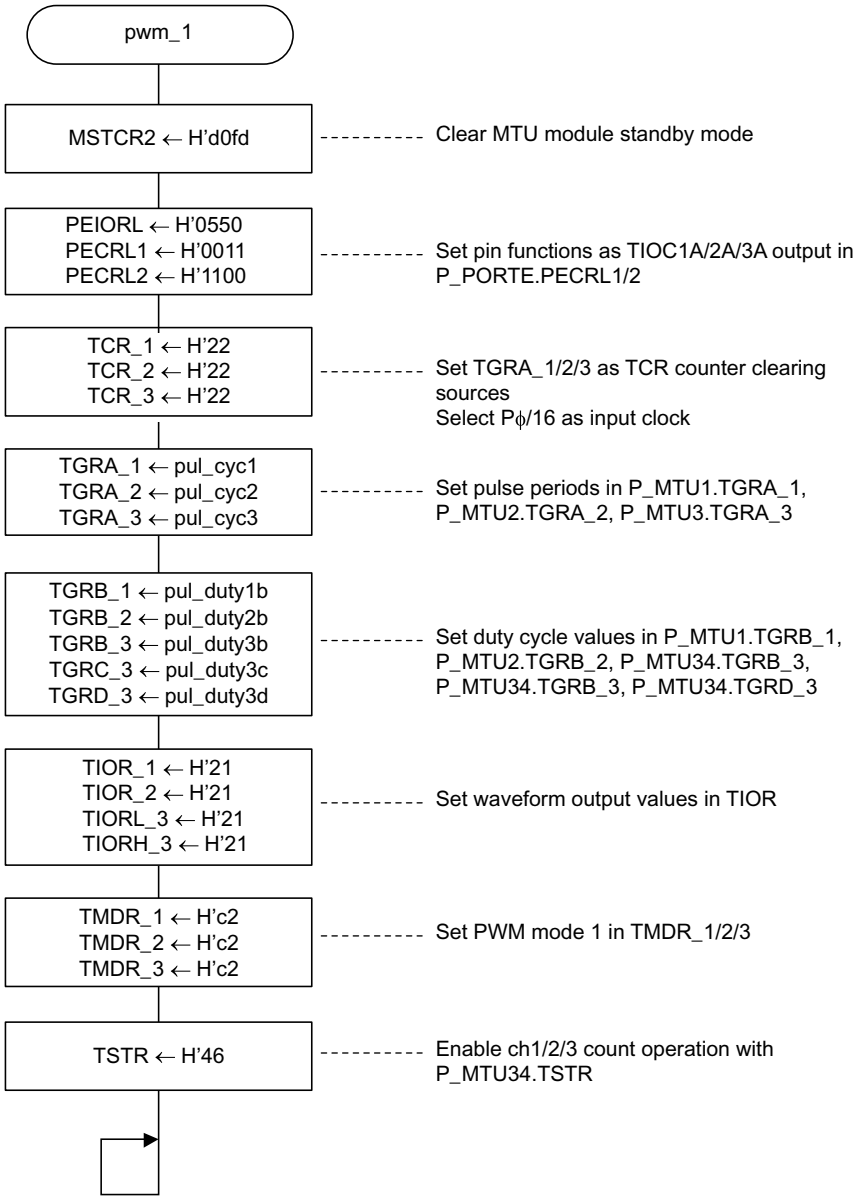
(4) RAM Used

This sample task does not use any RAM apart from the arguments.

Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



Program Listing

```
/************************************************************************/
/*                                INCLUDE FILE                          */
/************************************************************************/
#include<machine.h>
#include"iodefine_7145F.h"
/************************************************************************/
/*                                 PROTOTYPE                            */
/************************************************************************/
void pwm_1(void);
/************************************************************************/
/*                              RAM ALLOCATION                          */
/************************************************************************/
#define  pul_cyc1     (*(unsigned short  *)0xffffe000)
#define  pul_duty1b   (*(unsigned short  *)0xffffe002)
#define  pul_cyc2     (*(unsigned short  *)0xffffe004)
#define  pul_duty2b   (*(unsigned short  *)0xffffe006)
#define  pul_cyc3     (*(unsigned short  *)0xffffe008)
#define  pul_duty3b   (*(unsigned short  *)0xffffe00a)
#define  pul_duty3c   (*(unsigned short  *)0xffffe00c)
#define  pul_duty3d   (*(unsigned short  *)0xffffe00e)
/************************************************************************/
/*                              MAIN PROGRAM                            */
/************************************************************************/
void pwm_1(void)
 {
    P_STBY.MSTCR2.WORD = 0xd0fd;      /* Clear module standby mode */
    P_PORTE.PEIORL.WORD = 0x0550;     /* TIOC1A/2A/3A/3C = output */
    P_PORTE.PECRL1.WORD = 0x0011;
    P_PORTE.PECRL2.WORD = 0x1100;

    P_MTU1.TCR_1.BYTE = 0x22;         /* Counter1 clear by TGRA_1 */
    P_MTU2.TCR_2.BYTE = 0x22;         /* Counter2 clear by TGRA_2 */
    P_MTU34.TCR_3.BYTE = 0x22;        /* Counter3 clear by TGRA_3 */

    P_MTU1.TGRA_1 = pul_cyc1;         /* set TIOC1A period */
    P_MTU2.TGRA_2 = pul_cyc2;         /* set TIOC2A period */
    P_MTU34.TGRA_3 = pul_cyc3;        /* set TIOC3A period */
    P_MTU1.TGRB_1 = pul_duty1b;       /* set TIOC1A duty */
    P_MTU2.TGRB_2 = pul_duty2b;       /* set TIOC2A duty */
    P_MTU34.TGRB_3 = pul_duty3b;      /* set TIOC3A duty */
    P_MTU34.TGRC_3 = pul_duty3c;      /* set TIOC3C duty */
    P_MTU34.TGRD_3 = pul_duty3d;      /* set TIOC3C duty */
    P_MTU1.TIOR_1.BYTE = 0x21;        /* start"0",compare match"1"output*/
    P_MTU2.TIOR_2.BYTE = 0x21;        /* start"0",compare match"1"output*/
    P_MTU34.TIORL_3.BYTE = 0x21;      /* start"0",compare match"1"output*/
    P_MTU34.TIORH_3.BYTE = 0x21;      /* start"0",compare match"1"output*/
    P_MTU1.TMDR_1.BYTE = 0xc2;        /* PWM mode1 */
    P_MTU2.TMDR_2.BYTE = 0xc2;        /* PWM mode1 */
    P_MTU34.TMDR_3.BYTE = 0xc2;       /* PWM mode1 */

    P_MTU34.TSTR.BYTE = 0x46;         /* Timer counter start */
    while(1);
  }
```

2.4 PWM 7-Phase Output

PWM 7-Phase Output	MCU: SH7144/45	Functions Used: MTU (PWM Mode 2)

Specifications

(1) Seven-phase PWM output allowing the pulse high width and duty cycle to be varied is performed as shown in figure 2.10

(2) When operating with on-chip peripheral clock Pϕ = 40.0 MHz, the output PWM period can be set arbitrarily in the range 50 ns to 1.63 ms.

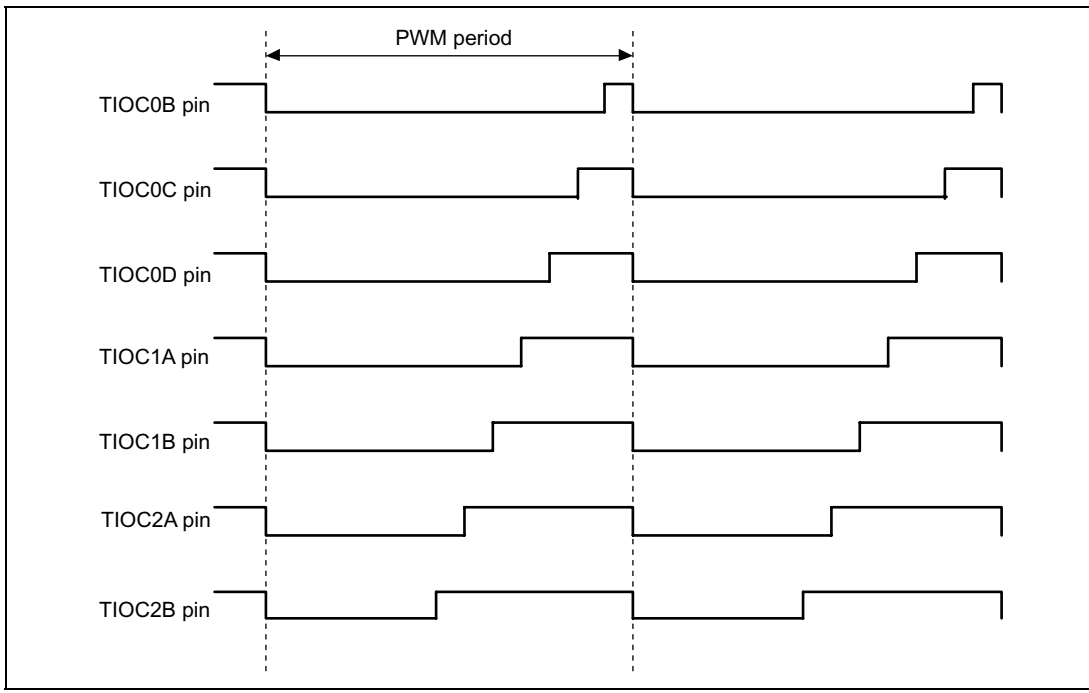


Figure 2.10 Example of 7-Phase PWM Waveform Output

Functions Used

(1) In this sample task, 7-phase PWM output is performed by synchronous operation of MTU ch0 to ch2.

(a) Figure 2.11 shows a block diagram of the MTU as used in this sample task.

This sample task uses the following MTU functions.

- A function that outputs pulses automatically by hardware without software intervention (output compare)
- A function that clears a counter when a compare match occurs (counter clearing)
- A function that reverses output each time a compare match occurs (toggle output)

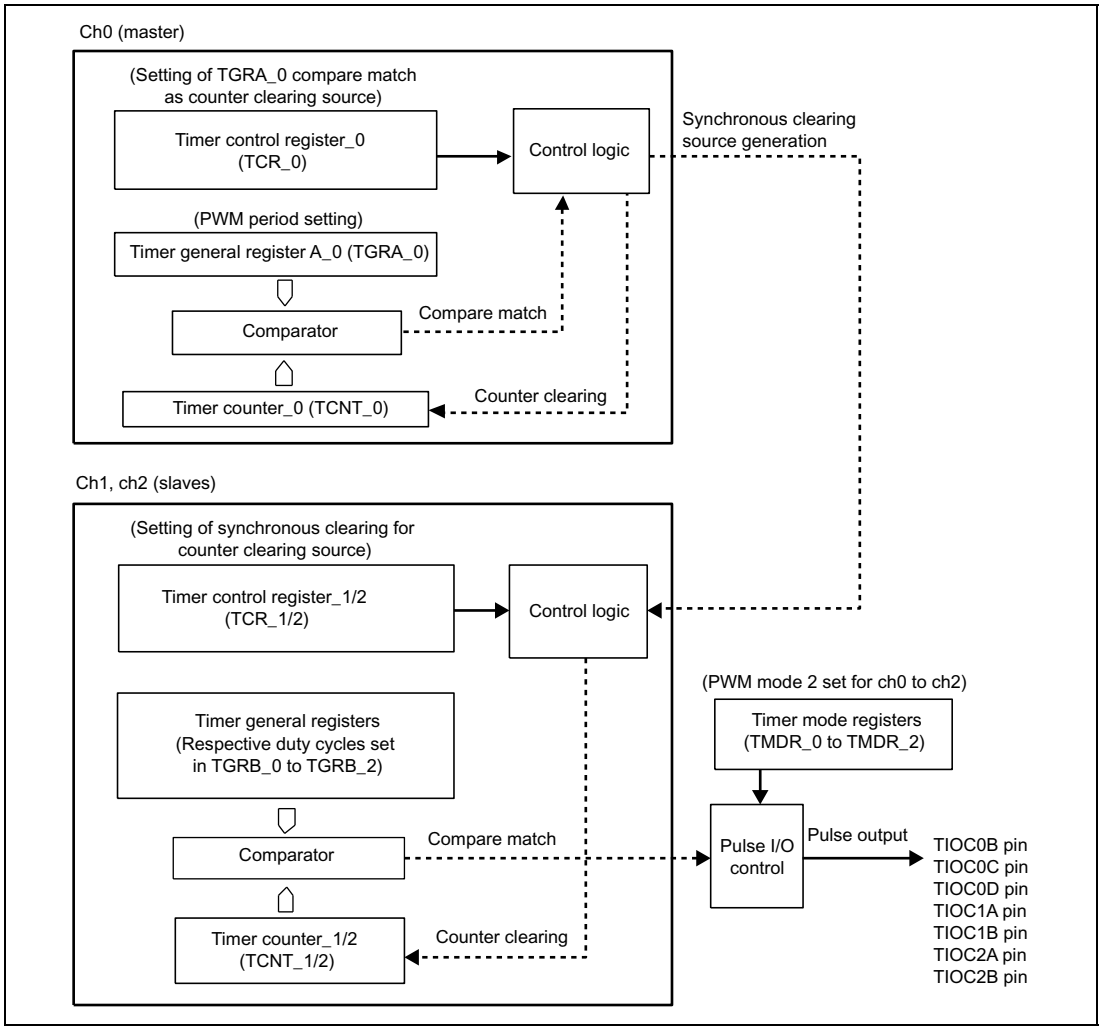


Figure 2.11 Block Diagram of Synchronous Clearing

(2) Table 2.4 shows the function assignments used in this task. PWM pulses are output by assigning MTU functions as shown in the table.

Table 2.4 MTU Function Assignments

Pin or Register Name	Function Assignment
TIOC0B TIOC0C TIOC0D TIOC1A TIOC1B TIOC2A TIOC2B	PWM pulse output pins
TSYR	Ch0/1/2 synchronous operation
TCR_0/1/2	Selection of ch0/1/2 timer counter clearing sources and input clocks
TGRA_0	PWM period setting
TGRB_0 TGRC_0 TGRD_0 TGRA_1 TGRB_1 TGRA_2 TGRB_2	Duty cycle setting
TMDR_0/1/2	Operation of ch0/1/2 in PWM Mode 2

Principles of Operation

(1) Figure 2.12 illustrates the principles of operation of this sample task. Seven-phase PWM output is performed from the ch0/1/2 PWM output pins (TIOC0B/C/D, TIOC1A/B, TIOC2A/B) by SH7145 hardware and software processing as shown in the figure.

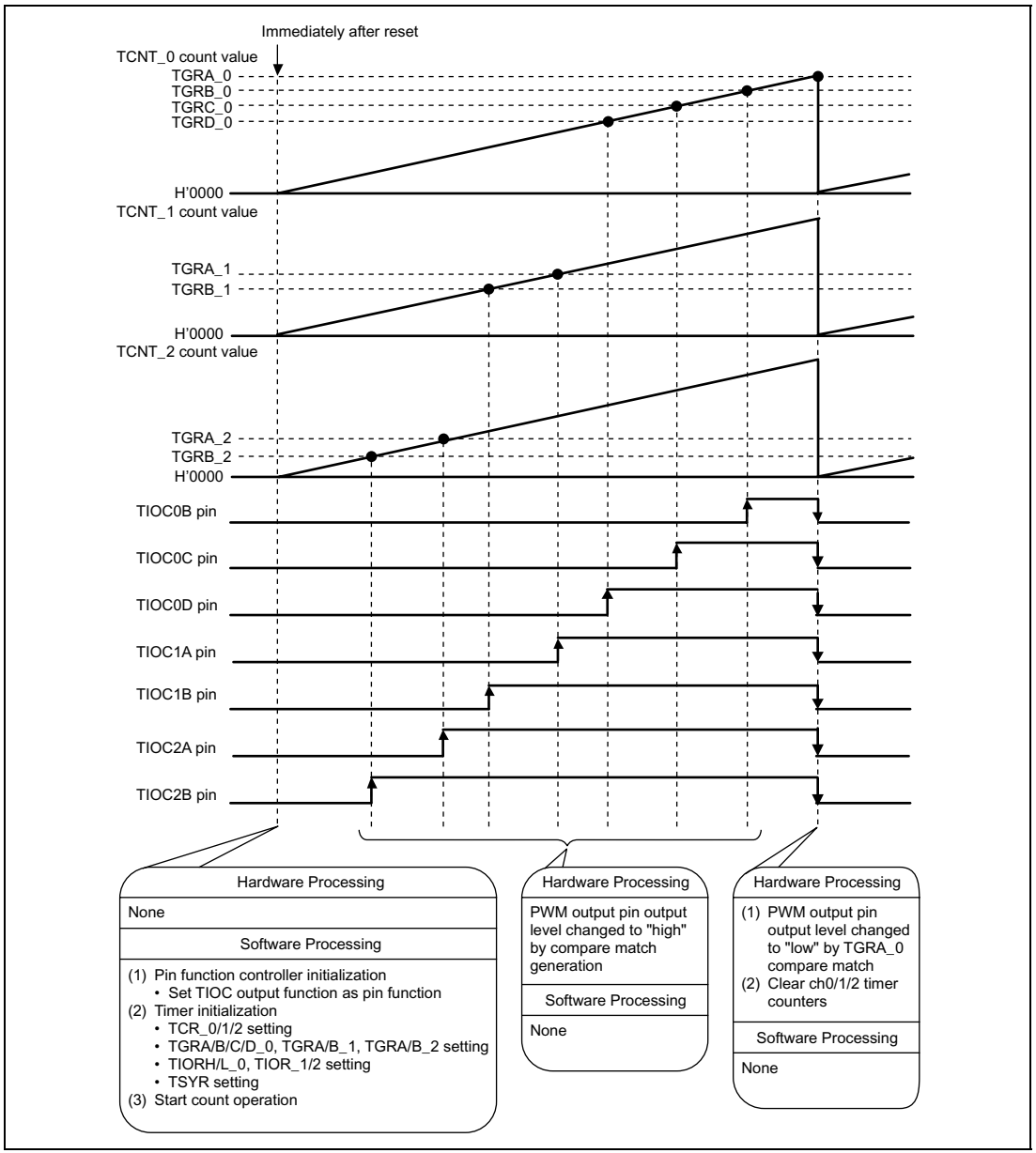


Figure 2.12 Principles of Operation of PWM Output (7-Phase) Using Sawtooth Waveform Generation

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	pwm_2	PFC and PWM output setting

(2) Arguments

Label or Register Name	Function Assignment	Data Length	Module	Input/ Output
pul_cyc0a	Used to set timer value for pulse period Pulse period is calculated using following equation: Pulse period (ns) = timer value × ϕ period (25 ns at 40.0 MHz operation)	1 word	Main routine	Input
pul_duty0b pul_duty0c pul_duty0d pul_duty1a pul_duty1b pul_duty2a pul_duty2b	Used to set TIOC pin output waveform transition timing			

(3) Internal Registers Used

Register Name	Function Assignment	Address	Set Value
P_STBY.MSTCR2	Module standby mode clearing	H'FFFF861E	H'd0fd
P_PORTE.PECRL2	Used to set TIOC0B/C/D, TIOC1A/B, TIOC2A/B output as pin functions	H'FFFF83BA	H'5554
P_MTU34.TSYR	Synchronous operation set for timer counters 0/1/2	H'FFFF8241	H'07
P_MTU0.TCR_0 P_MTU1.TCR_1 P_MTU2.TCR_2	Clearing by TGRA_0 compare match set as timer counter clearing source Pϕ/16 selected as input clock	H'FFFF8260 H'FFFF8280 H'FFFF82A0	H'22 H'62 H'62
P_MTU0.TGRA_0	PWM period setting	H'FFFF8268	pul_cyc0
P_MTU0.TGRB_0	Used to set timer counter value causing high output from TIOC0B	H'FFFF826A	pul_duty0b
P_MTU0.TGRC_0	Used to set timer counter value causing high output from TIOC0C	H'FFFF826C	pul_duty0c

Register Name	Function Assignment	Address	Set Value
P_MTU0.TGRD_0	Used to set timer counter value causing high output from TIOC0D	H'FFFF826E	pul_duty0d
P_MTU1.TGRA_1	Used to set timer counter value causing high output from TIOC1A	H'FFFF8288	pul_duty1a
P_MTU1.TGRB_1	Used to set timer counter value causing high output from TIOC1B	H'FFFF828A	pul_duty1b
P_MTU2.TGRA_2	Used to set timer counter value causing high output from TIOC2A	H'FFFF82A8	pul_duty2a
P_MTU2.TGRB_2	Used to set timer counter value causing high output from TIOC2B	H'FFFF82AA	pul_duty2b
P_MTU0.TIORH_0	Sets TGRA_0 initial output 0, 0 output on output compare, TGRB_0 initial output 0, 1 output on output compare	H'FFFF8262	H'21
P_MTU0.TIORL_0	Sets TGRC_0 initial output 0, 1 output on output compare, TGRD_0 initial output 0, 1 output on output compare	H'FFFF8263	H'22
P_MTU1.TIOR_1	Sets TGRA_1 initial output 0, 1 output on output compare, TGRB_1 initial output 0, 1 output on output compare	H'FFFF8282	H'22
P_MTU1.TIOR_2	Sets TGRA_2 initial output 0, 1 output on output compare, TGRB_2 initial output 0, 1 output on output compare	H'FFFF82A2	H'22
P_MTU0.TMDR_0 P_MTU1.TMDR_1 P_MTU2.TMDR_2	Used to set PWM Mode 2 as operating mode of each channel	H'FFFF8261 H'FFFF8281 H'FFFF82A1	H'c3 H'c3 H'c3

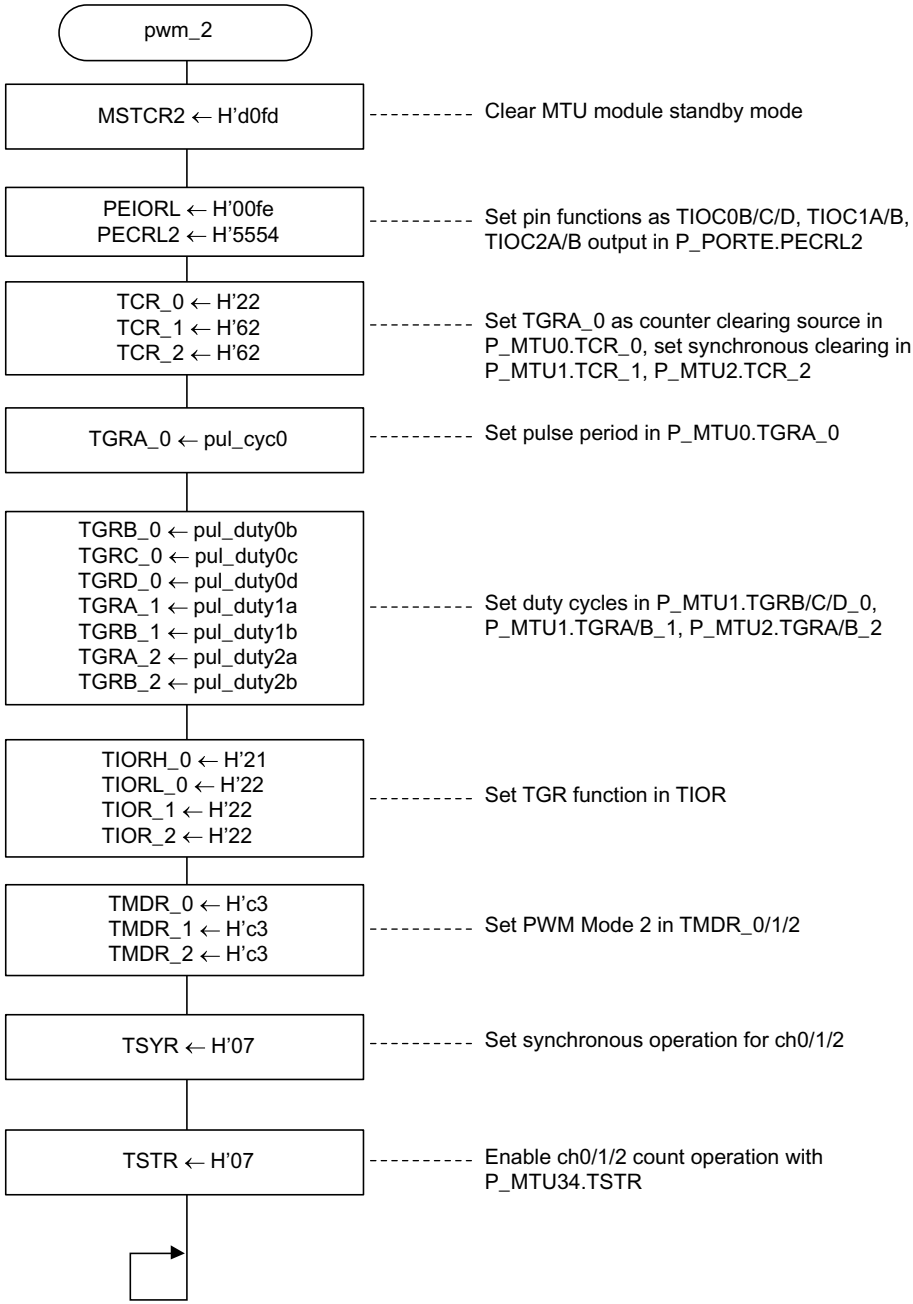
(4) RAM Used

This sample task does not use any RAM apart from the arguments.

Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



Program Listing

```
/************************************************************************/
/*                                INCLUDE   FILE                        */
/************************************************************************/
#include<machine.h>
#include"iodefine_7145F.h"
/************************************************************************/
/*                                  PROTOTYPE                           */
/************************************************************************/
void pwm_2(void);
/************************************************************************/
/*                                RAM ALLOCATION                        */
/************************************************************************/
#define pul_cyc0      (*(unsigned short *)0xffffe000)
#define pul_duty0b    (*(unsigned short *)0xffffe002)
#define pul_duty0c    (*(unsigned short *)0xffffe004)
#define pul_duty0d    (*(unsigned short *)0xffffe006)
#define pul_duty1a    (*(unsigned short *)0xffffe008)
#define pul_duty1b    (*(unsigned short *)0xffffe00a)
#define pul_duty2a    (*(unsigned short *)0xffffe00c)
#define pul_duty2b    (*(unsigned short *)0xffffe00e)
/************************************************************************/
/*                                 MAIN PROGRAM                         */
/************************************************************************/
void pwm_2(void)
 {
   P_STBY.MSTCR2.WORD = 0xd0fd;    /* Clear module standby mode */
   P_PORTE.PEIORL.WORD = 0x00fe;   /* TIOC0B/C/D,TIOC1A/B,TIOC2A/B output */
   P_PORTE.PECRL2.WORD = 0x5554;

   P_MTU0.TCR_0.BYTE = 0x22;       /* Counter clear by TGRA_0 */
   P_MTU1.TCR_1.BYTE = 0x62;       /* Counter clear by TGRA_0 */
   P_MTU2.TCR_2.BYTE = 0x62;       /* Counter clear by TGRA_0 */

   P_MTU0.TGRA_0 = pul_cyc0;       /* Set PWM period */
   P_MTU0.TGRB_0 = pul_duty0b;     /* Set PWM duty */
   P_MTU0.TGRC_0 = pul_duty0c;
   P_MTU0.TGRD_0 = pul_duty0d;
   P_MTU1.TGRA_1 = pul_duty1a;
   P_MTU1.TGRB_1 = pul_duty1b;
   P_MTU2.TGRA_2 = pul_duty2a;
   P_MTU2.TGRB_2 = pul_duty2b;

   P_MTU0.TIORH_0.BYTE = 0x21;     /* TIOC0B=start"0",compare match"1"output */
   P_MTU0.TIORL_0.BYTE = 0x22;     /* TIOC0B/0C=start"0",compare match"1"output*/
   P_MTU1.TIOR_1.BYTE = 0x22;      /* TIOC1A/1B=start"0",compare match"1"output */
   P_MTU2.TIOR_2.BYTE = 0x22;      /* TIOC2A/2B=start"0",compare match"1"output */

   P_MTU0.TMDR_0.BYTE = 0xc3;      /* Set PWM mode2 */
   P_MTU1.TMDR_1.BYTE = 0xc3;      /* Set PWM mode2 */
   P_MTU2.TMDR_2.BYTE = 0xc3;      /* Set PWM mode2 */

   P_MTU34.TSYR.BYTE = 0x07;       /* Synchronize ch0,ch1,ch2 */
   P_MTU34.TSTR.BYTE = 0x07;       /* Start timer counter */
   while(1);
  }
```

2.5 Positive-Phase/Negative Phase PWM 3-Phase Output

Positive-Phase/Negative Phase PWM 3-Phase Output	MCU: SH7144/45	Functions Used: MTU (Reset-Synchronized PWM Mode)

Specifications

(1) Positive-phase and negative-phase 3-phase pulse (duty pulse) output is performed that allows the pulse high width and duty cycle to be varied, as shown in figure 2.13.

(2) When operating with on-chip peripheral clock Pϕ = 40.0 MHz, the output pulse period can be set arbitrarily in the range 25.0 ns to 1.63 ms.

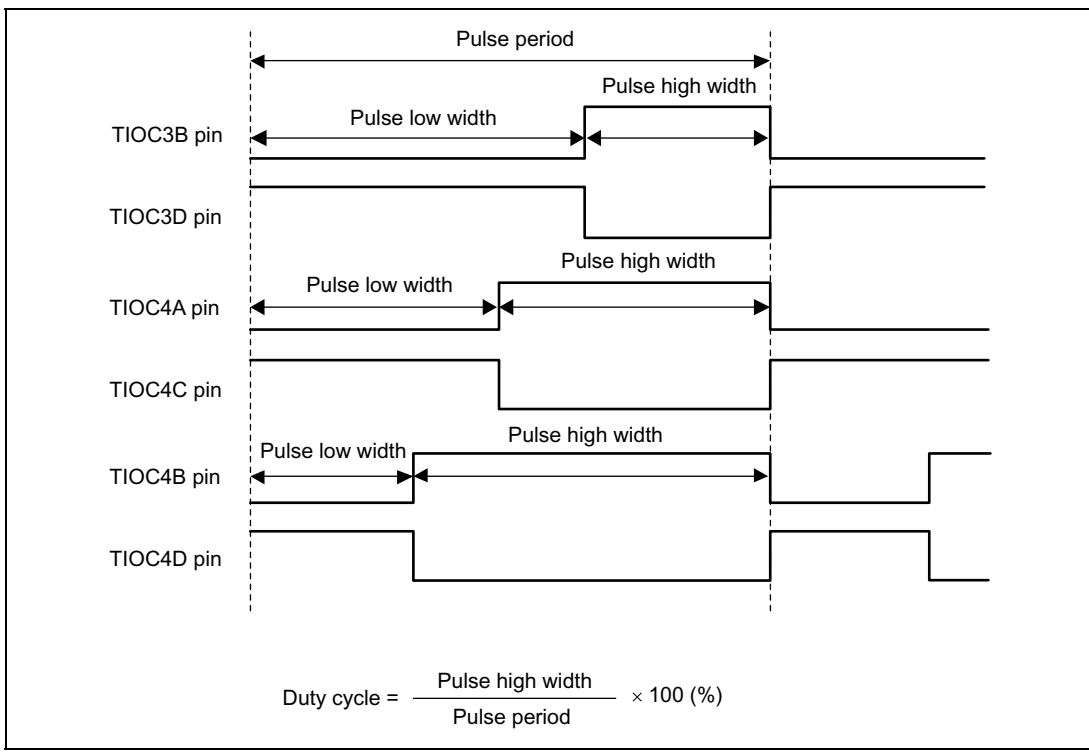


Figure 2.13 Positive-Phase/Negative Phase PWM 3-Phase Output Waveforms

Functions Used

(1) In this sample task, MTU ch3 and ch4 are used in combination, and 3-phase PWM waveform output is performed with one common transition point in the relationship between the positive phase and negative phase.

In reset-synchronized PWM mode, PWM waveforms are generated using buffer operation, with TGRA and TGRC operating as a pair, and TRGB and TGRD operating as a pair.

(a) Figure 2.14 shows a block diagram of the MTU as used in this sample task.

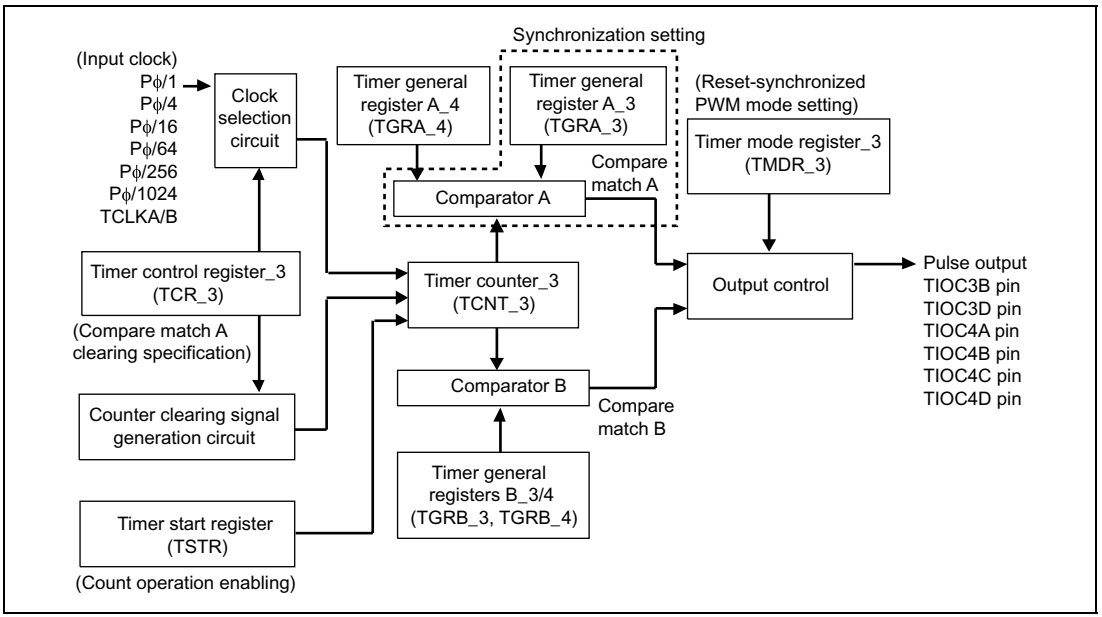


Figure 2.14 Block Diagram of MTU/ch3, ch4

(2) Table 2.5 shows the function assignments used in this task. PWM pulses are output by assigning MTU functions as shown in the table.

Table 2.5 Function Assignments

Pin or Register Name	Function Assignment
TIOC3B	PWM output 1
TIOC3D	Negative-phase waveform of PWM output 1
TIOC4A	PWM output 2
TIOC4B	PWM output 3
TIOC4C	Negative-phase waveform of PWM output 2
TIOC4D	Negative-phase waveform of PWM output 3
TCR_3	Selection of ch3 timer counter clearing source and input clock
TMDR_3	Ch3 set to operate in reset-synchronized PWM mode
TGRA_3	PWM period setting
TGRB_3 TGRA_4 TGRB_4	Duty cycle setting

Principles of Operation

(1) Figure 2.15 illustrates the principles of operation of this sample task. Three-phase PWM waveforms are output from the PWM output pins (TIOC3B/D, TIOC4A/B/C/D) by SH7145 hardware and software processing as shown in the figure.

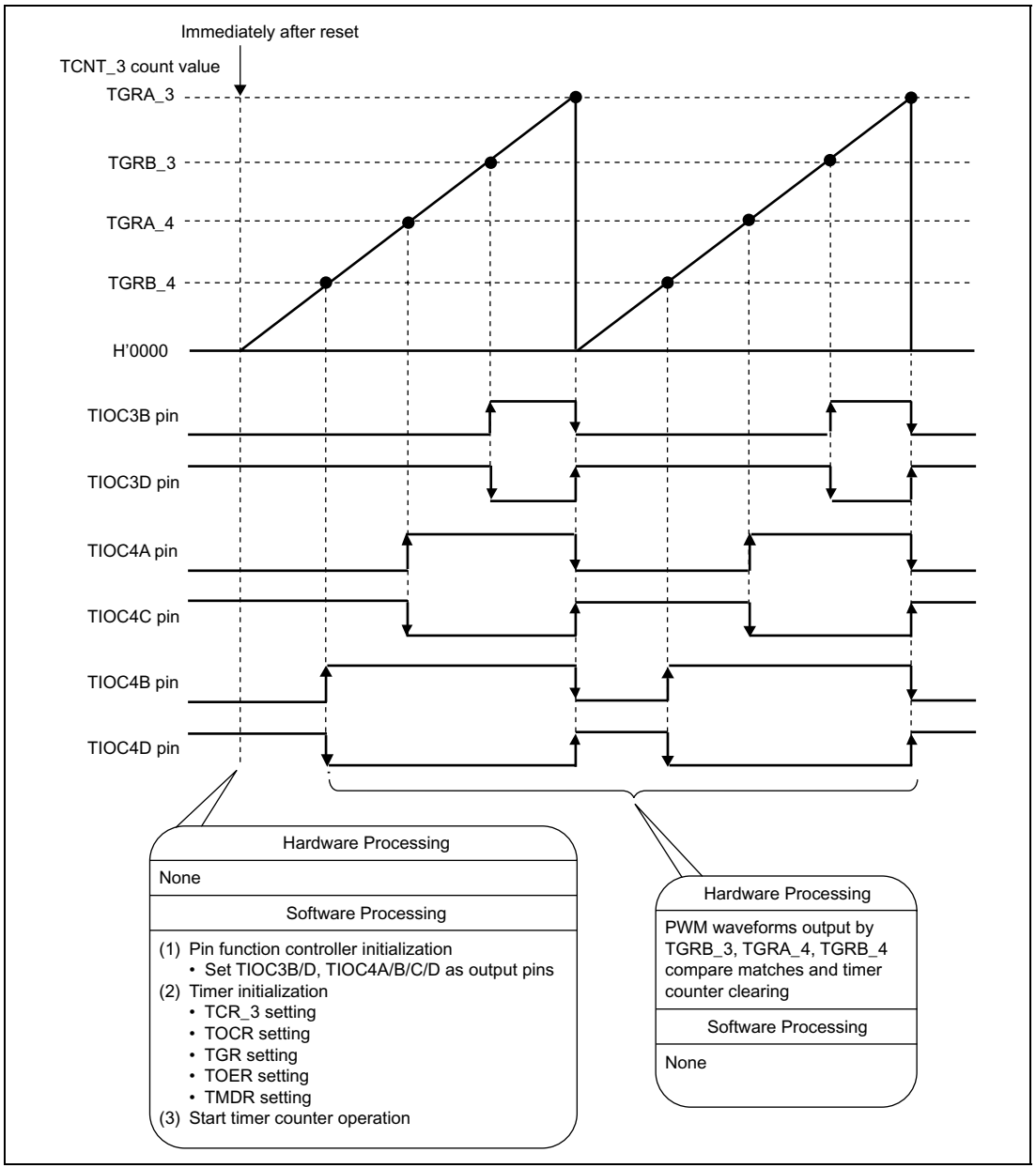


Figure 2.15 Principles of Operation of Reset-Synchronized PWM Waveforms

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	rst_pwm	PFC and PWM output setting

(2) Arguments

Label or Register Name	Function	Data Length	Module	Input/ Output
pul_cyc1	Used to set timer value for pulse period Pulse period is calculated using following equation: Pulse period (ns) = timer value × ϕ period (25.0 ns at 40.0 MHz operation)	1 word	Main routine	Input
pul_duty3b pul_duty4a pul_duty4b	Used to set TIOC pin output waveform transition timing			

(3) Internal Registers Used

Register Name	Function	Address	Set Value
P_PORTE.PECRL1	Sets TIOC3B/D, TIOC4A/B/C/D as output pins	H'FFFF83B8	H'5544
P_MTU34.TCR_3	Sets TGRA_3 compare match as timer counter 3 counter clearing source Selects Pϕ/16 as input clock	H'FFFF8200	H'22
P_MTU34.TOCR	Positive-phase/negative-phase PWM output level inversion control	H'FFFF820B	H'03
P_MTU34.TGRA_3	PWM period setting	H'FFFF8218	pul_cyc1
P_MTU34.TGRB_3	Used to set TIOC3B/D pin PWM output waveform transition timing	H'FFFF821A	pul_duty3b
P_MTU34.TGRA_4	Used to set TIOC4A/C pin PWM output waveform transition timing	H'FFFF821C	pul_duty4a
P_MTU34.TGRB_4	Used to set TIOC4B/D pin PWM output waveform transition timing	H'FFFF821E	pul_duty4b
P_MTU34.TOER	Sets enabling of reset-synchronized PWM output	H'FFFF821E	H'ff
P_MTU34.TMDR_3	Sets reset-synchronized PWM mode for ch3/4	H'FFFF8202	H'c8
P_STBY.MSTCR2	MTU module standby mode clearing	H'FFFF861E	H'd0fd

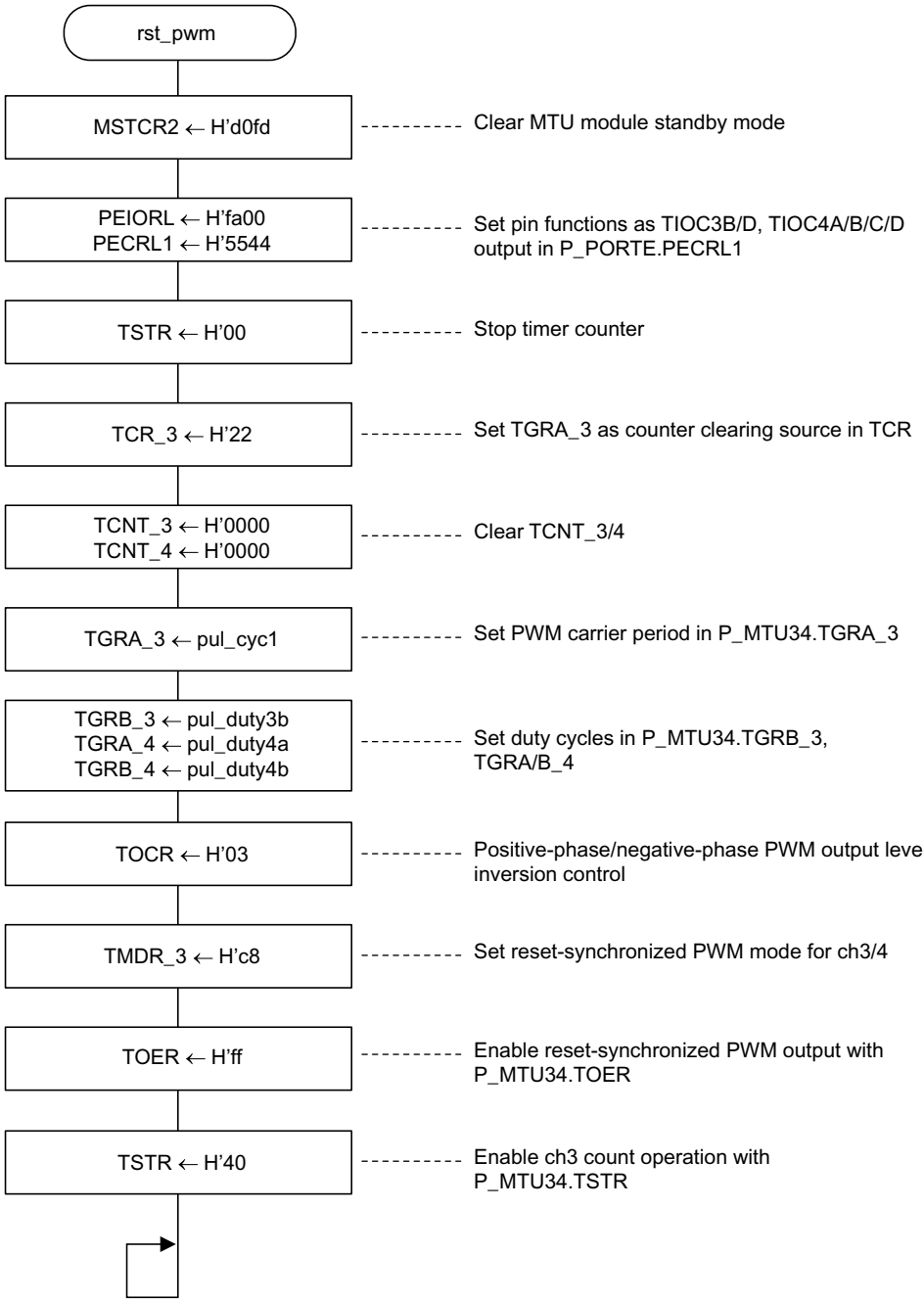
(4) RAM Used

This sample application does not use any RAM apart from the arguments.

Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



Program Listing

```
/************************************************************************/
/*                              INCLUDE FILE                            */
/************************************************************************/
#include<machine.h>
#include"iodefine_7145F.h"
/************************************************************************/
/*                               PROTOTYPE                              */
/************************************************************************/
void rst_pwm(void);
/************************************************************************/
/*                            RAM ALLOCATION                            */
/************************************************************************/
#define pul_cyc1     (*(unsigned short *)0xffffe000)
#define pul_duty3b   (*(unsigned short *)0xffffe002)
#define pul_duty4a   (*(unsigned short *)0xffffe004)
#define pul_duty4b   (*(unsigned short *)0xffffe006)
/************************************************************************/
/*                             MAIN PROGRAM                             */
/************************************************************************/
void rst_pwm(void)
 {
   P_STBY.MSTCR2.WORD = 0xd0fd;        /* Clear MTU module standby mode */
   P_PORTE.PEIORL.WORD = 0xfa00;       /* TIOC3B/D,TIOC4A/B/C/D output */
   P_PORTE.PECRL1.WORD = 0x5544;
   P_MTU34.TSTR.BYTE = 0x00;
   P_MTU34.TCR_3.BYTE = 0x20;          /* Counter clear by TGRA_3 */
   P_MTU34.TCNT_3 = 0x0000;            /* Clear timer counter */
   P_MTU34.TCNT_4 = 0x0000;
   P_MTU34.TGRA_3 = pul_cyc1;          /* Set PWM period */
   P_MTU34.TGRB_3 = pul_duty3b;        /* Set duty */
   P_MTU34.TGRA_4 = pul_duty4a;
   P_MTU34.TGRB_4 = pul_duty4b;
   P_MTU34.TOCR.BYTE = 0x03;           /* timer output control register */
   P_MTU34.TMDR_3.BYTE = 0xc8;         /* Reset synchronized PWM mode */
   P_MTU34.TOER.BYTE = 0xff;           /* Enable timer output */
   P_MTU34.TSTR = 0x40;                /* Start timer counter */
   while(1);
  }
```

2.6 Complementary PWM 3-Phase Output

Complementary PWM 3-Phase Output	MCU: SH7144/45	Functions Used: MTU (Complementary PWM Mode)

Specifications

(1) Three-phase PWM waveform output is performed with a non-overlapping relationship between positive and negative phases, as shown in figure 2.16.

(2) The duty cycle can be changed between 0% and 100% by setting an arbitrary value in RAM.

$$\text{Duty cycle} = \frac{\text{Pulse high width}}{\text{Pulse period}} \times 100\ (\%)$$

(3) Toggle waveform output is performed synchronized with the period.

(4) When operating with on-chip peripheral clock Pϕ = 40.0 MHz, the output pulse period can be set arbitrarily in the range 50.0 ns to 1.63 ms.

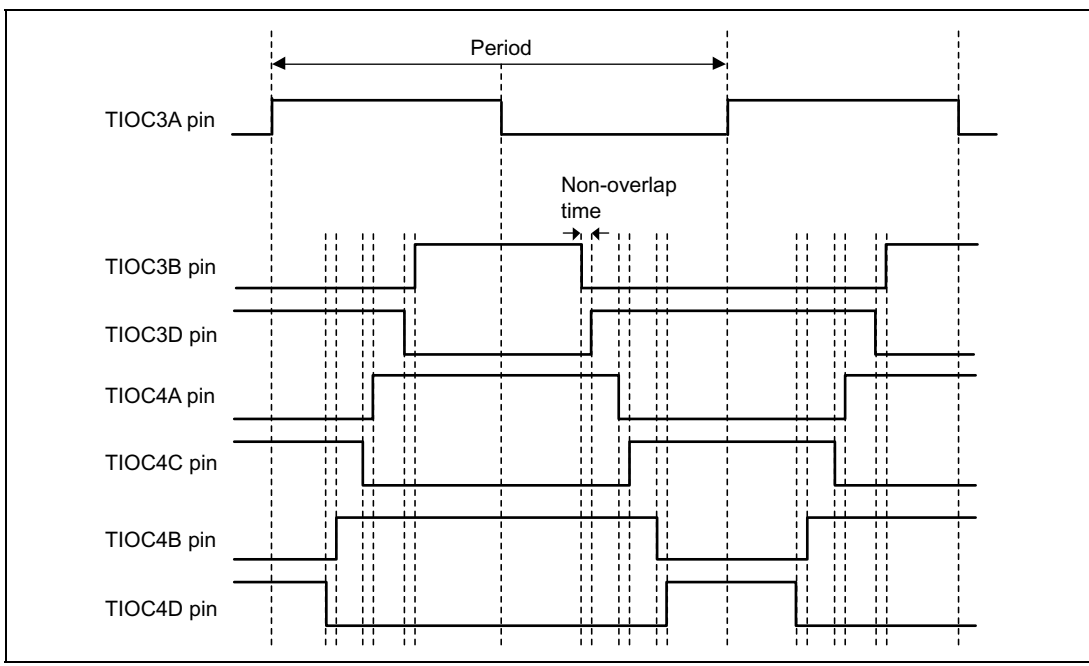


Figure 2.16 Complementary PWM 3-Phase Output Waveforms

Functions Used

(1) In this sample task, 3-phase PWM waveform output with a non-overlapping relationship between positive and negative phases is performed using MTU channels 3 and 4.

(a) Figure 2.17 shows a block diagram of the MTU/ch3, ch4 as used in this sample task.

This sample task uses the following functions.

- A function that performs 3-phase PWM waveform output with a non-overlapping relationship between positive and negative phases (complementary PWM mode)
- A function that transfers buffer register (TGRC/D_3, TGRC/D_4) contents to compare registers (TGRA/B_3, TGRA/B_4) when a compare match occurs
- A function that outputs a toggle waveform synchronized with the PWM waveform period

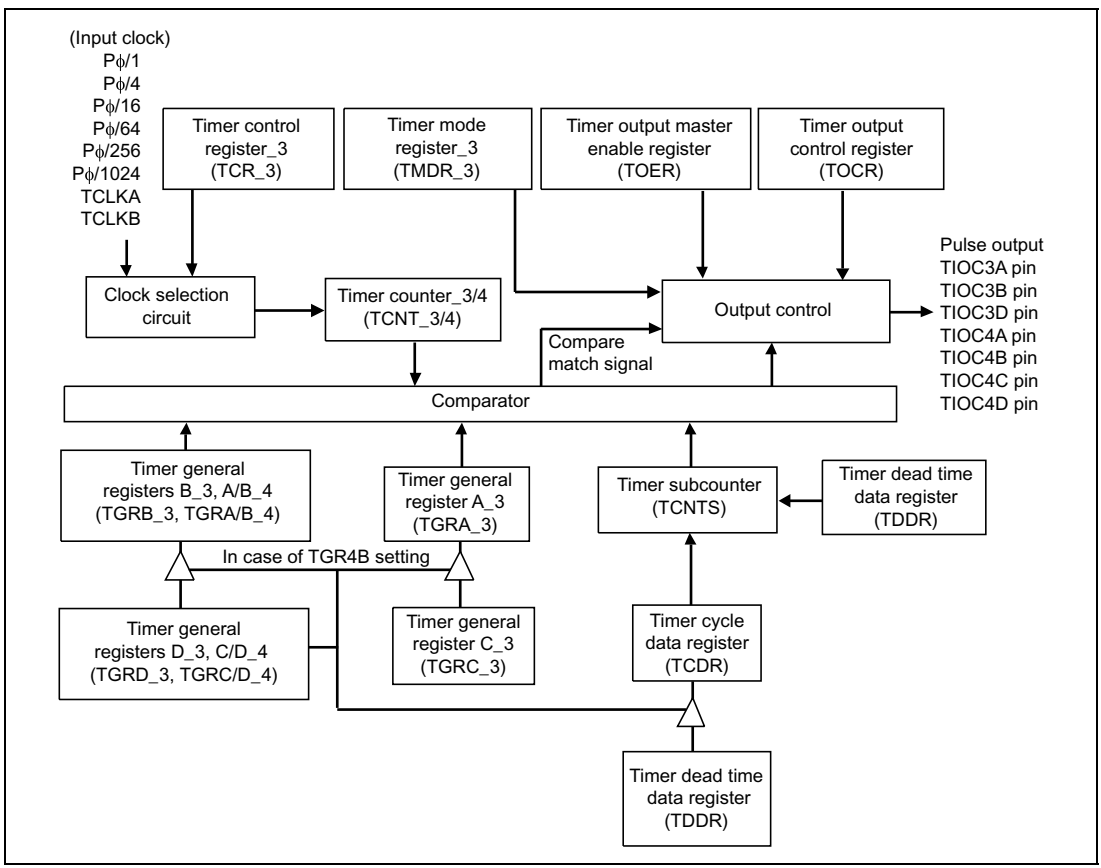


Figure 2.17 Block Diagram of MTU/ch3, ch4

(2) Table 2.6 shows the function assignments used in this task. PWM pulses are output by assigning MTU functions as shown in the table.

Table 2.6 Function Assignments

Pin or Register Name	Function Assignment
TIOC3A	Toggle output synchronized with PWM period
TIOC3C	PWM output 1
TIOC3D	Negative-phase waveform in non-overlapping relationship with PWM output 1
TIOC4A	PWM output 2
TIOC4B	PWM output 3
TIOC4C	Negative-phase waveform in non-overlapping relationship with PWM output 2
TIOC4D	Negative-phase waveform in non-overlapping relationship with PWM output 3
TOCR	Enabling/disabling of toggle output synchronized with PWM period
TOER	MTU output pin output enabling/disabling
TCR_3	Selection of ch3 timer counter clearing source and input clock
TMDR_3	Ch3, ch4 set to complementary PWM mode
TGRA_3	TCNT_3 upper-limit value setting (1/2 carrier period + dead time)
TGRC_3	TGRA_3 buffer register
TGRB_3	Output pulse transition point setting (compare register)
TGRA_4	
TGRB_4	
TGRC_4	TGRA_4 buffer register
TGRD_4	TGRB_4 buffer register
TDDR	Dead time setting
TCDR	TCNT_4 upper-limit value setting (1/2 carrier period)
TCBR	TCDR buffer register

Principles of Operation

(1) Figure 2.18 illustrates the principles of operation. Complementary PWM waveform output is performed by SH7145 hardware and software processing.

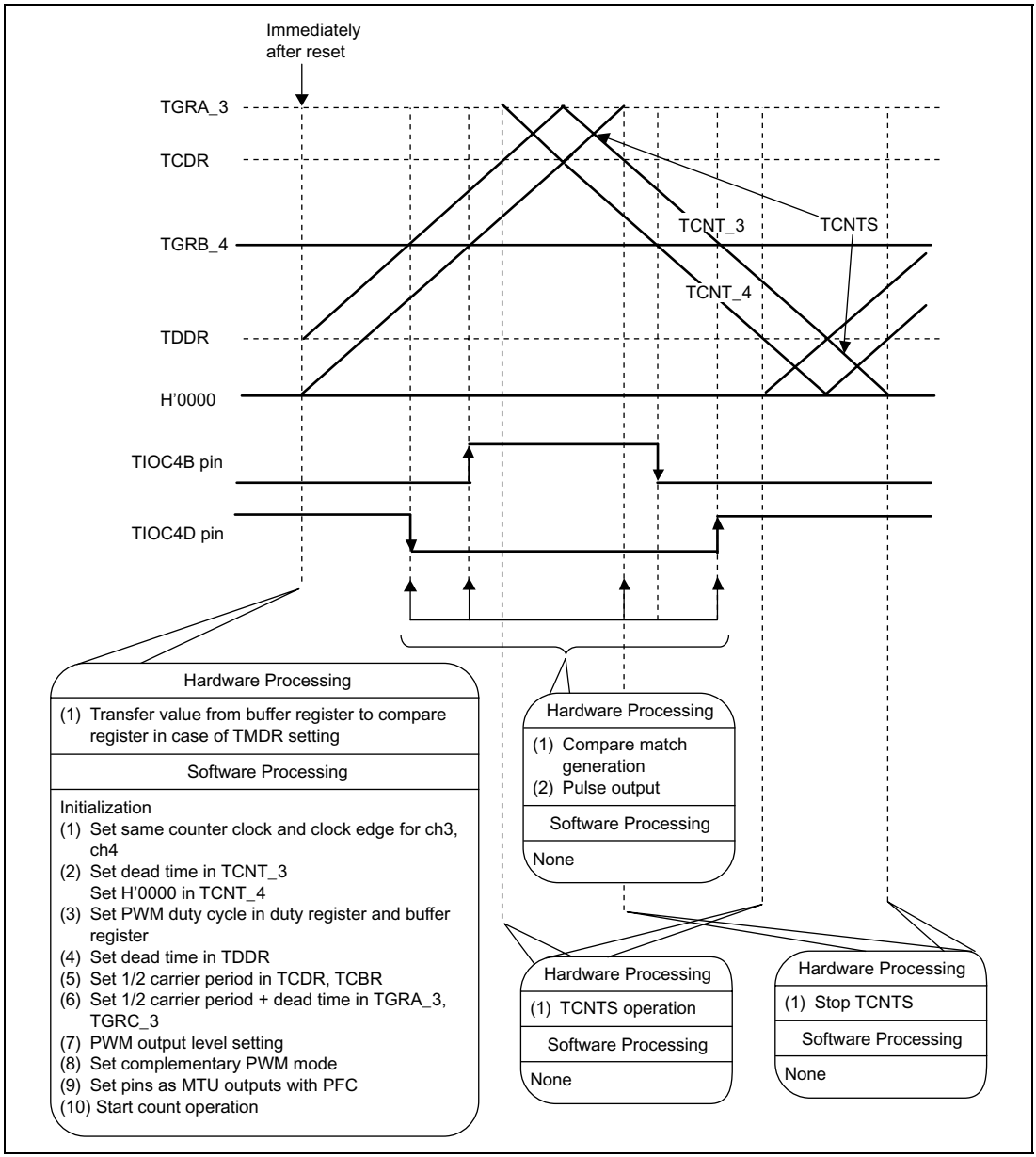


Figure 2.18 Principles of Operation of Complementary PWM Single-Phase Waveform Output

(2) Figure 2.19 shows the PWM waveform output method. When complementary PWM mode is set, the following rules apply to data transfer and compare operations.

Data Transfer

— In period Ta, data written to a buffer register is always transferred to a temporary register.

— In period Tb1, when the transfer mode is set to transfer at the peak, data is not transferred from a buffer register to a temporary register. In period Tb2, the operation is the same as in period Ta. Similarly, when a trough setting is made, data is not transferred in period Tb2.

— Data transfer to a buffer register can be performed arbitrarily.

Compare Match

— In period Tb, two registers—the temporary register and compare register—and three counters—TCNT_3/4 and TCNTS—are compared, and the PWM waveform is controlled.

— In area (a), pre-change data and compare matches (3) and (4) have priority.

— In area (b), post-change data and compare matches (1) and (2) have priority.

Generation of a compare match whereby the output waveform goes to the active level (compare match (1) or (3)) occurs only after generation of a compare match whereby the respective output waveform goes to the positive level (compare match (4) or (2)).

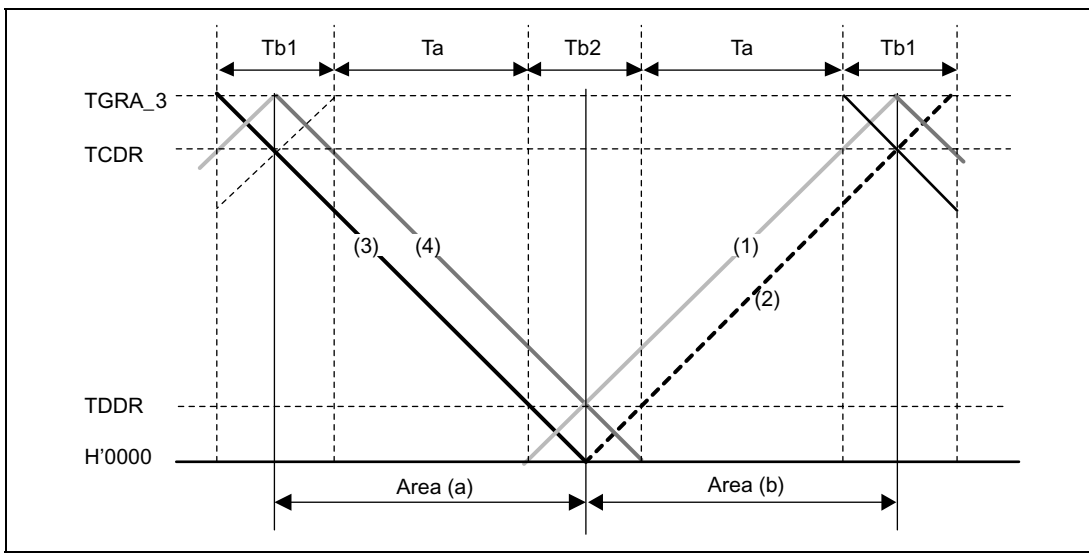


Figure 2.19 Principles of Operation of PWM Waveform Output Method

(3) Figure 2.20 illustrates the principles of operation. Complementary PWM waveform output is performed by SH7145 hardware and software processing.

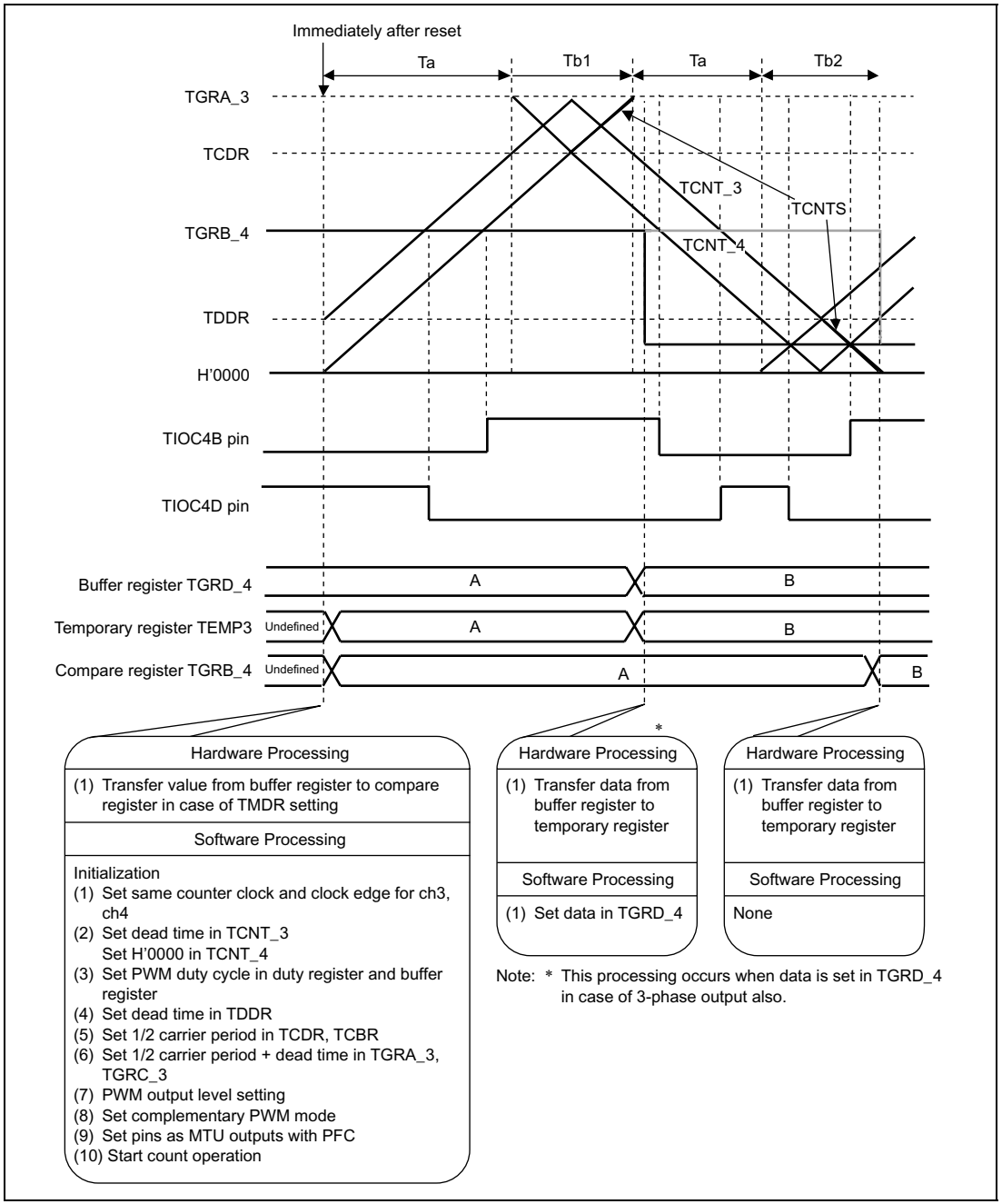


Figure 2.20 Principles of Operation of Complementary PWM Single-Phase Waveform Output

(4) Figure 2.21 illustrates the principles of operation. Three-phase PWM output is performed from the ch3 and ch4 PWM output pins (TIOC3B/D, TIOC4A/B/C/D) by SH7145 hardware and software processing as shown in the figure.

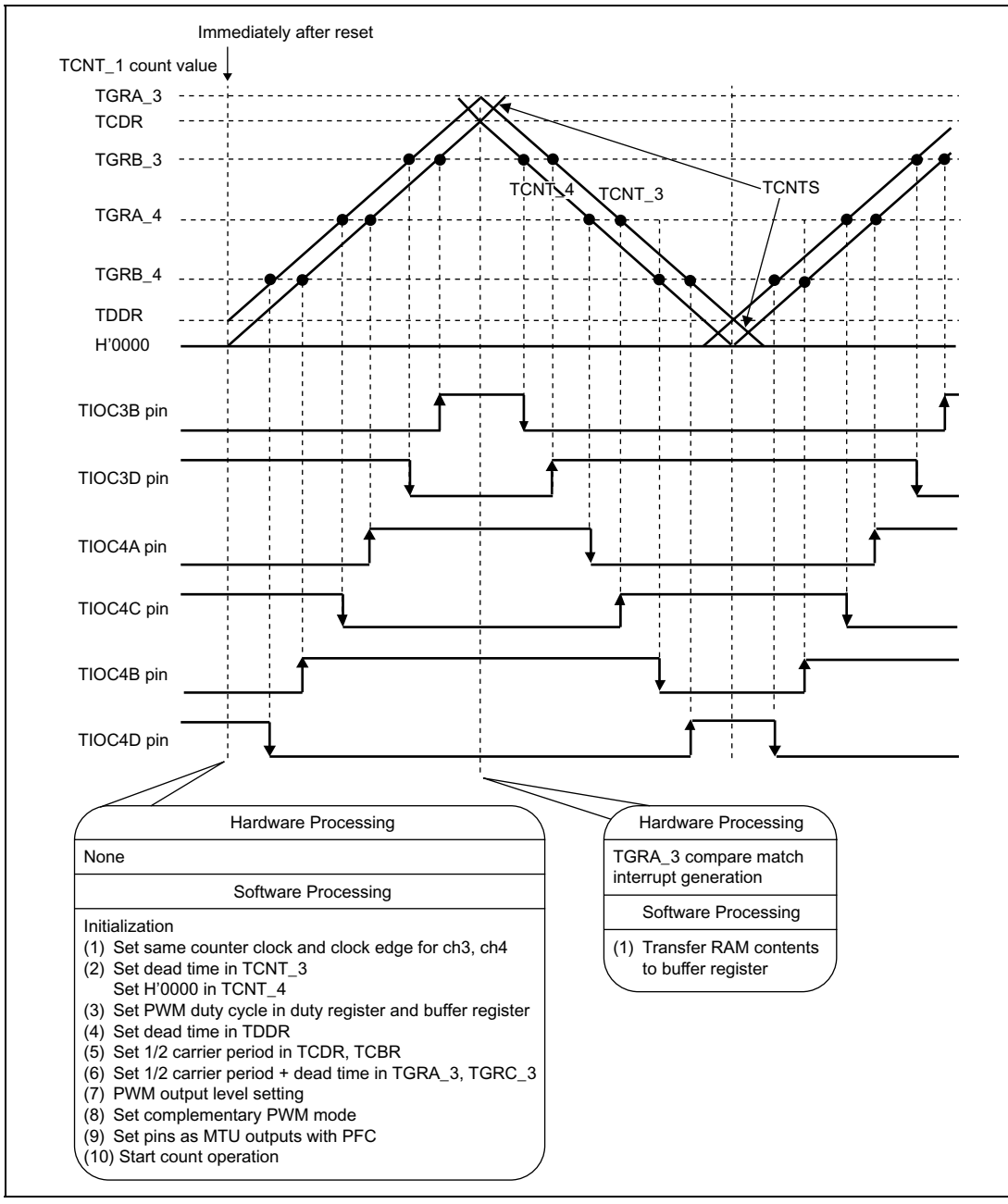


Figure 2.21 Principles of Operation of PWM Waveforms

(5) Figure 2.22 illustrates the principles of operation. Toggle output synchronized with the PWM period is performed from the TIOC3A pin by SH7145 hardware and software processing.

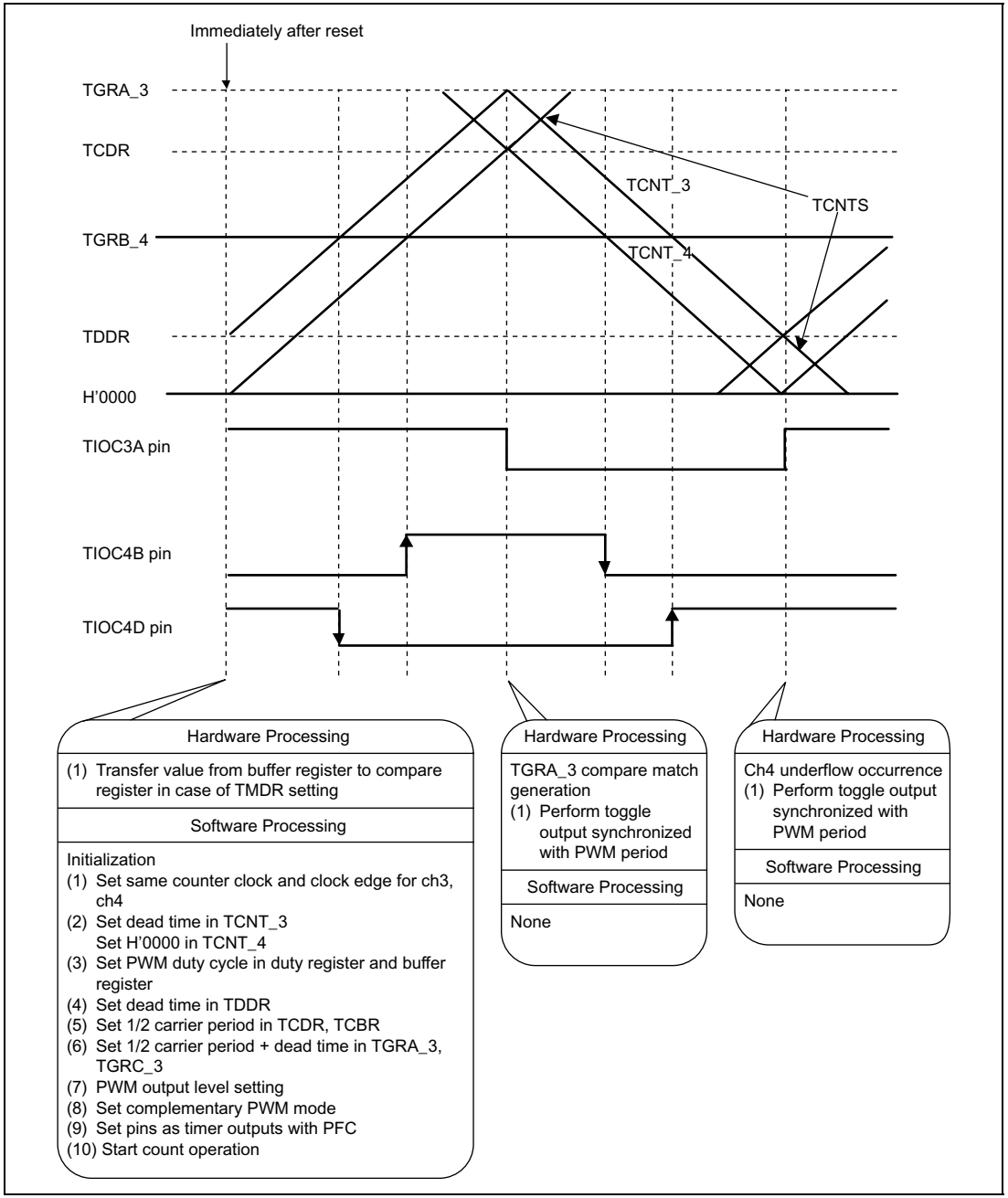


Figure 2.22 Principles of Operation of Toggle Waveform Output Synchronized with PWM Period

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	comple	Complementary PWM output setting
Data setting	setdata	Sets waveform transition timing in buffer register

(2) Arguments

Label or Register Name	Function	Data Length	Module	Input/ Output
pul_cyc1	Used to set pulse 1/2 period + dead time value Pulse period is calculated using following equation: Pulse period (ns) = timer value × ϕ period (25.0 ns at 40.0 MHz operation)	1 word	Main routine	Input
pul_duty3d	Used to set TIOC pin output waveform transition timing			
pul_duty4c				
pul_duty4d				
c_cyc	PWM carrier period register value setting			
dead_time	Non-overlap time setting		Main routine Data setting	

(3) Internal Registers Used

Register Name	Function	Address	Set Value
P_STBY.MSTCR2	MTU module standby mode clearing	H'FFFF861E	H'd0fd
P_PORTE.PEIORL	Sets TIOC3B/D, TIOC4A/B/C/D pins as outputs	H'FFFF83B4	H'fb00
P_PORTE.PECRL1	Sets pin functions as TIOC3B/D, TIOC4A/B/C/D	H'FFFF83B8	H'5545
P_MTU34.TCR_3	MTU/ch3 counter clock and clock edge setting	H'FFFF8200	H'01
P_MTU34.TCR_4	MTU/ch4 counter clock and clock edge setting	H'FFFF8201	H'01
P_MTU34.TIER_3	Enables TGIA_3 interrupt	H'FFFF8208	H'41

Register Name	Function	Address	Set Value
P_MTU34.TGRA_3	1/2 carrier period + dead time setting	H'FFFF8218	pul_cyc1
P_MTU34.TGRC_3	P_MTU34.TGRA_3 buffer register (same value set as P_MTU34.TGRA_3)	H'FFFF8224	pul_cyc1
P_MTU34.TCNT_3	Dead time setting	H'FFFF8210	dead_time
P_MTU34.TCNT_4	H'0000 set	H'FFFF8212	H'0000
P_MTU34.TGRB_3	Setting of PWM duty cycle output from TIOC3B, TIOC3D	H'FFFF821A	pul_duty3d
P_MTU34.TGRA_4	Setting of PWM duty cycle output from TIOC4A, TIOC4C	H'FFFF821C	pul_duty4c
P_MTU34.TGRB_4	Setting of PWM duty cycle output from TIOC4B, TIOC4D	H'FFFF821E	pul_duty4d
P_MTU34.TGRD_3	P_MTU34.TGRB_3 buffer register (same value set as P_MTU34.TGRB_3)	H'FFFF8226	pul_duty3d
P_MTU34.TGRC_4	P_MTU34.TGRA_4 buffer register (same value set as P_MTU34.TGRA_4)	H'FFFF8228	pul_duty4c
P_MTU34.TGRD_4	P_MTU34.TGRB_4 buffer register (same value set as P_MTU34.TGRB_4)	H'FFFF822A	pul_duty4d
P_MTU34.TDDR	Dead time setting	H'FFFF8216	dead_time
P_MTU34.TCDR	1/2 carrier period setting	H'FFFF8214	c_cyc
P_MTU34.TCBR	P_MTU34.TCDR buffer register (same value set as P_MTU34.TCDR)	H'FFFF8222	c_cyc
P_MTU34.TOCR	Enabling of toggle output synchronized PWM period, and positive-phase/negative phase output level setting	H'FFFF820B	H'43
P_MTU34.TMDR_3	Sets complementary PWM mode	H'FFFF8202	H'ff
P_MTU34.TOER	Output enable setting for PWM waveform output pins	H'FFFF820A	H'ff
P_INTC.IPRE	Sets 15 as MTU channel 3 interrupt priority level	H'FFFF8350	H'00f0

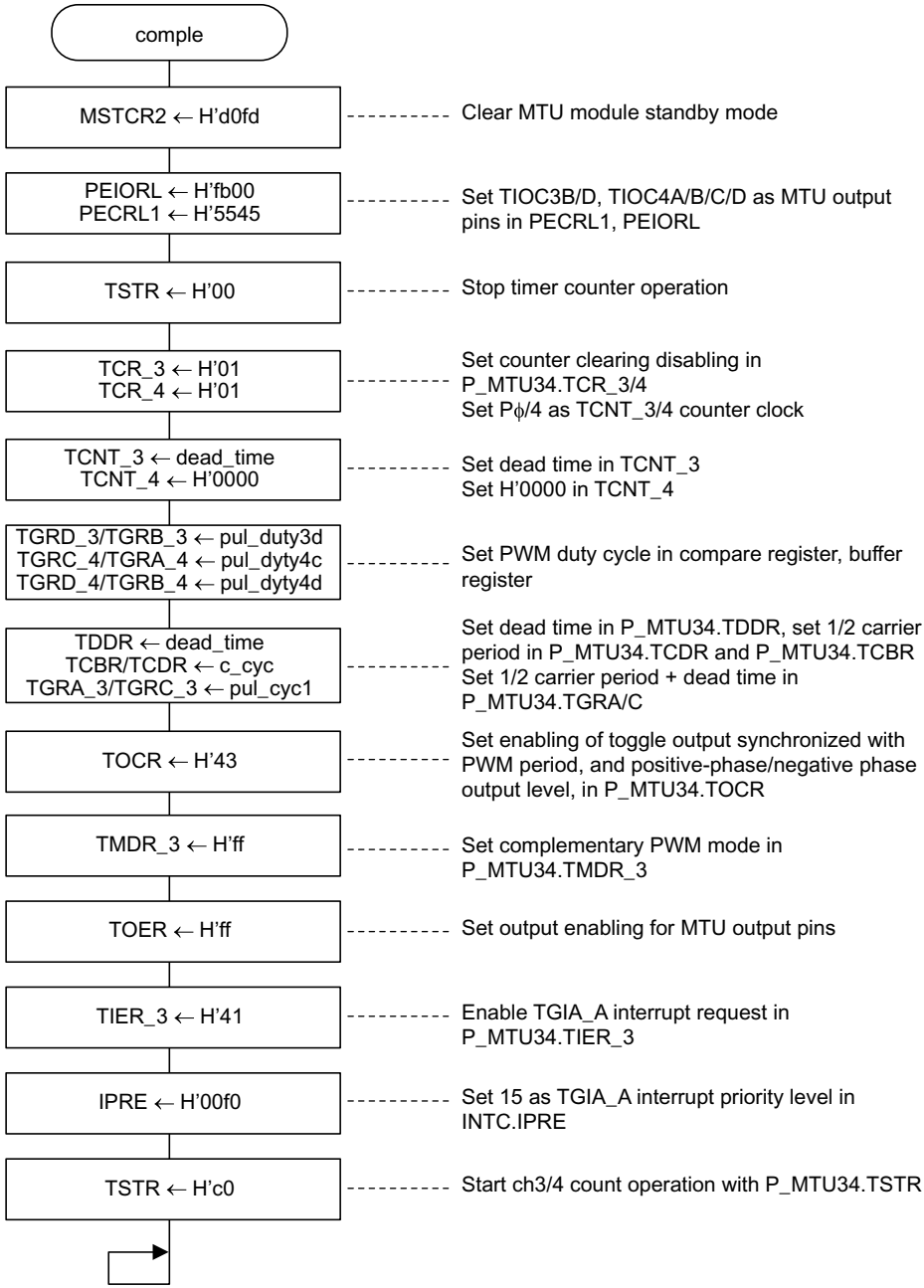
(4) RAM Used

This sample application does not use any RAM apart from the arguments.

Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



Program Listing

```
/*------------------------------------------------------------------------*/
/*                              INCLUDE FILE                              */
/*------------------------------------------------------------------------*/
#include <machine.h>
#include "iodefine_7145F.h"
/*------------------------------------------------------------------------*/
/*                              PROTOTYPE                                 */
/*------------------------------------------------------------------------*/
extern void comple(void);
#pragma interrupt(setdata)
/*------------------------------------------------------------------------*/
/*                              RAM ALLOCATION                            */
/*------------------------------------------------------------------------*/
#define pul_cyc1          (*(unsigned short *)0xffffe000)
#define pul_duty3d        (*(unsigned short *)0xffffe002)
#define pul_duty4c        (*(unsigned short *)0xffffe004)
#define pul_duty4d        (*(unsigned short *)0xffffe006)
#define c_cyc             (*(unsigned short *)0xffffe008)
#define dead_time         (*(unsigned short *)0xffffe00a)
/*------------------------------------------------------------------------*/
/*                              MAIN PROGRAM                              */
/*------------------------------------------------------------------------*/
void comple(void)
{
  P_STBY.MSTCR2.WORD = 0xd0fd;     /* MTU module standby mode clear */
  P_PORTE.PEIORL.WORD = 0xfb00;    /* TIOC3B/D,TIOC4A/B/C/D = output */
  P_PORTE.PECRL1.WORD = 0x5545;

  P_MTU34.TSTR.BYTE = 0x00;        /* Stop timer count */
  P_MTU34.TCR_3.BYTE = 0x01;       /* Don't clear TCNT_3 */
  P_MTU34.TCR_4.BYTE = 0x01;       /* Don't clear TCNT_4 */
  P_MTU34.TCNT_3 = dead_time;      /* Set dead time */
  P_MTU34.TCNT_4 = 0x0000;
  P_MTU34.TGRD_3 = pul_duty3d;     /* TGRB_3 buffer register */
  P_MTU34.TGRB_3 = pul_duty3d;     /* PWM output1 compare register */
  P_MTU34.TGRC_4 = pul_duty4c;     /* TGRA_4 buffer register */
  P_MTU34.TGRA_4 = pul_duty4c;     /* PWM output2 compare register */
  P_MTU34.TGRD_4 = pul_duty4d;     /* TGRB_4 buffer register */
  P_MTU34.TGRB_4 = pul_duty4d;     /* PWM output3 compare register */
  P_MTU34.TDDR = dead_time;        /* Dead time register */
  P_MTU34.TCBR = c_cyc;            /* 1/2 carrier period */
  P_MTU34.TCDR = c_cyc;            /* TCDR buffer register */
  P_MTU34.TGRA_3 = pul_cyc1;       /* 1/2 carrier period + dead time */
  P_MTU34.TGRC_3 = pul_cyc1;       /* TGRA_3 buffer register */
  P_MTU34.TOCR.BYTE = 0x43;        /* Timer output control register */
  P_MTU34.TMDR_3.BYTE = 0xff;      /* Set complementary-pwm mode */
  P_MTU34.TOER.BYTE = 0xff;        /* Timer output enable register */
  P_MTU34.TIER_3.BYTE = 0x41;      /* Timer interrupt enable register */

  INTC.IPRE.WORD = 0x00f0;         /* Set initialize level = 15 */
  set_imask(0x0);                  /* Set imask level = 0 */
  P_MTU34.TSTR.BYTE = 0xc0;        /* Start timer counter3/4 */
  while(1);                        /* Loop */
```

```
void setdata()
 {
   P_MTU34.TSR_3.BYTE &= 0xfe;     /* interrupt flag clear */
   P_MTU34.TCBR = c_cyc;
   P_MTU34.TGRC_3 = pul_cyc1;
   P_MTU34.TGRD_3 = pul_duty3d;
   P_MTU34.TGRC_4 = pul_duty4c;
   P_MTU34.TGRD_4 = pul_duty4d;
  }
```

2.7 2-Phase Encoder Count

2-Phase Encoder Count	MCU: SH7144/45	Functions Used: MTU (Phase Counting Mode)

Specifications

(1) Two external clocks are input to channel 1 (ch1), and a counter is incremented or decremented according to the phase difference of the pulses, as shown in figure 2.23. The ch1 count is measured in synchronization with a measurement times set in ch0 (measurement times 1 and 2), and the result is set in RAM.

(2) H'0000 is set as the timer counter initial value, and counting can be performed from -2,147,483,648 to 2,147,483,647 using a software counter.

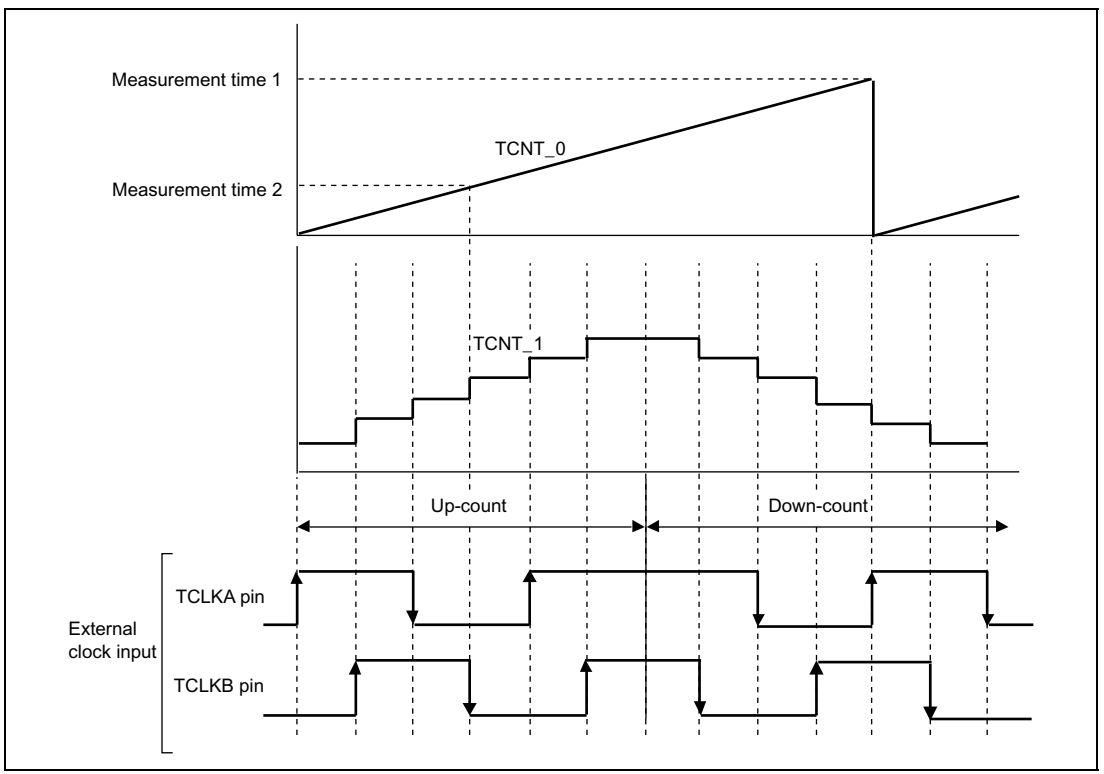


Figure 2.23 2-Phase Encoder Counter Capture

Functions Used

(1) In this sample task, measurement times are set in TGRA/B_0 using an MTU ch1 up/down-counter.

Using a TGRA/B_0 output compare as a trigger, the TCNT_1 value for the control period is captured by ch1 input capture. In addition, the ch1 counter input clock width is captured using ch0 input capture.

(a) Figure 2.24 shows a block diagram of ch0. In ch0, a ch1 input capture trigger is output every measurement time using the following functions. In ch1, the TCNT_1 value is measured when an input capture signal is input.

- A function that outputs pulses automatically by hardware without software intervention (output compare)
- A function that performs pulse input edge detection, and captures a timer value in an internal register (input capture)

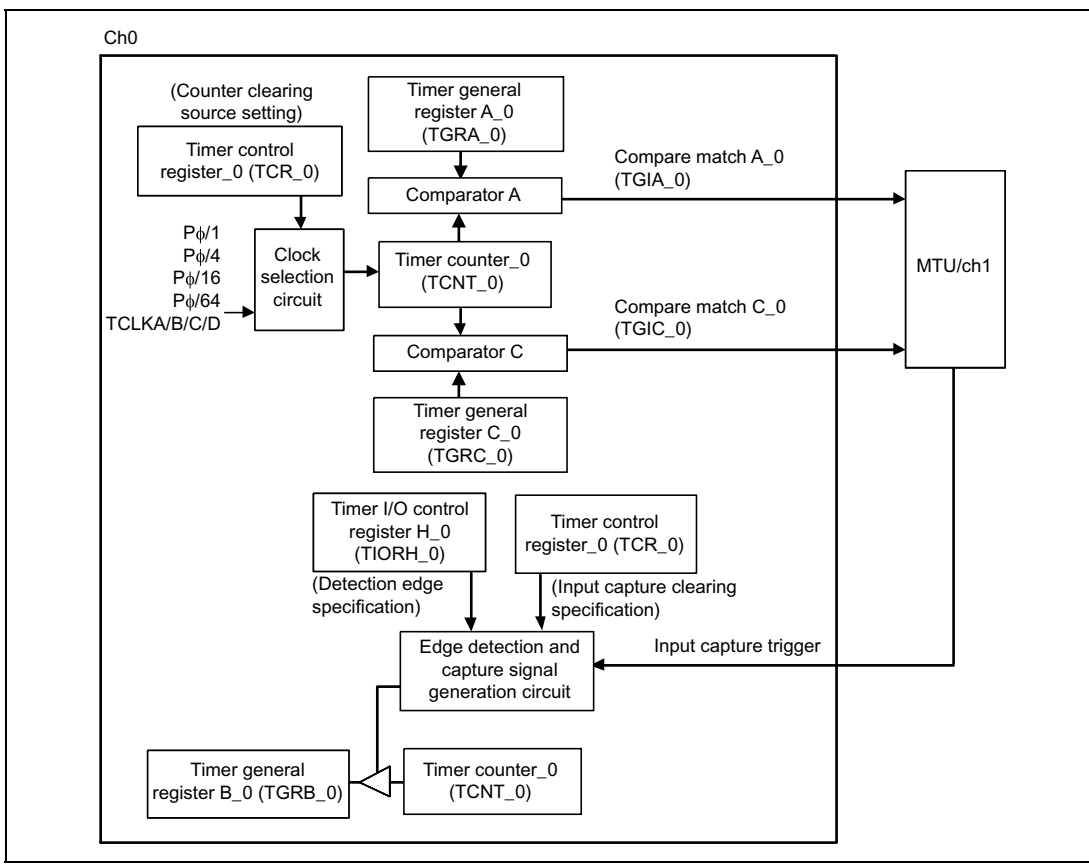


Figure 2.24 Block Diagram of MTU/ch0

(b) Figure 2.25 shows a block diagram of ch1. In ch1, a timer counter is incremented/decremented using the following functions. The counter value when an input capture signal rising edge is detected is taken as the measurement result.

- A function that detects the phase difference between two external clocks, and increments/decrements a timer counter (phase counting mode)
- A function that performs pulse input edge detection, and captures a timer value in an internal register (input capture)
- A function that initiates interrupt handling when input capture occurs
- A function that clears the timer counter when a pulse input edge is detected
- A function that initiates interrupt handling when timer counter overflow or underflow is detected

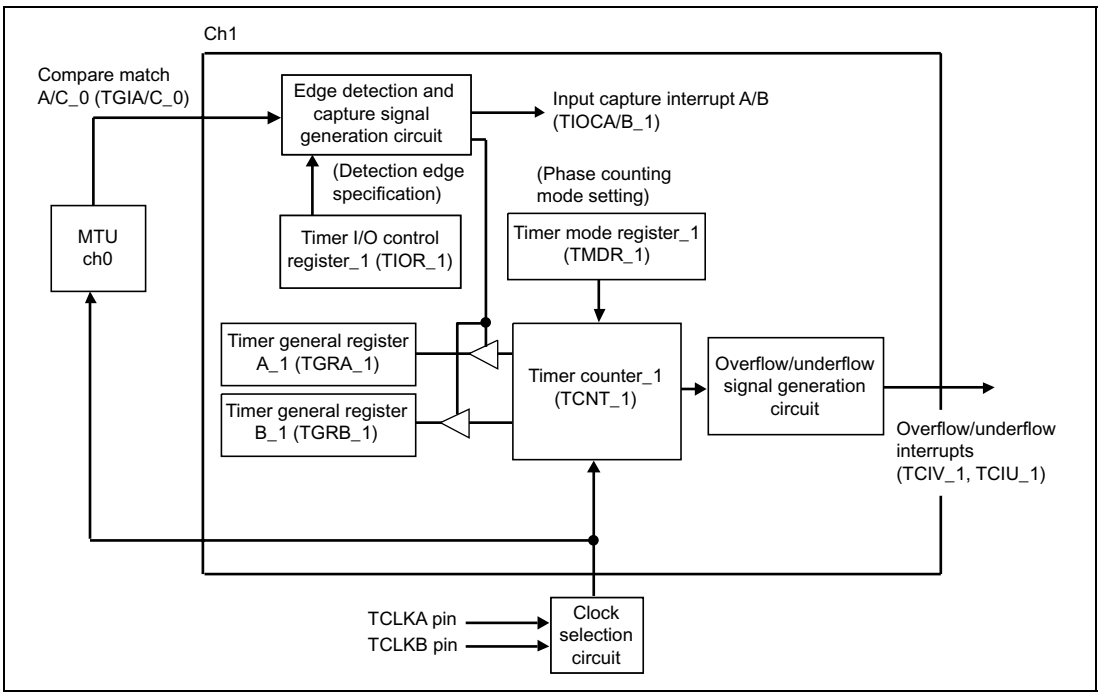


Figure 2.25 Block Diagram of MTU/ch1

(2) Table 2.7 shows the function assignments used in this sample task. MTU functions are assigned as shown in the table to detect the phase difference between two 2-phase encoder pulses, and increment/decrement a counter.

Table 2.7 Function Assignments

Pin or Register Name	Function Assignment
TCLKA	External clock input pins
TCLKB	
TSTR	Enabling/disabling of MTU ch0, ch1 timer counter operation
TCR_0	Selection of counter clock and counter clearing source
TIORH_0	TIOC0A output compare setting. Setting of TIOC0B for input capture on TCNT_1 increment/decrement
TIORL_0	TIOC0C output compare setting
TGRA_0	Measurement time 1 setting
TGRB_0	Count result stored on input capture B
TGRC_0	Measurement time 2 setting
TMDR_1	Sets phase counting mode for MTU/ch1
TCR_1	Selection of counter clock and counter clearing source
TIOR_1	Setting of TIOC1A/B for input capture on TGRA_0, TGRC_0 output compare occurrence
TIER_1	Enables TIOC1A/B, TCIU_1, TCIV_1 interrupts
TGRA_1	Count result storage on input capture
TGRB_1	

Principles of Operation

(1) Figure 2.26 illustrates the principles of operation. A counter is incremented or decremented by SH7145 hardware and software processing.

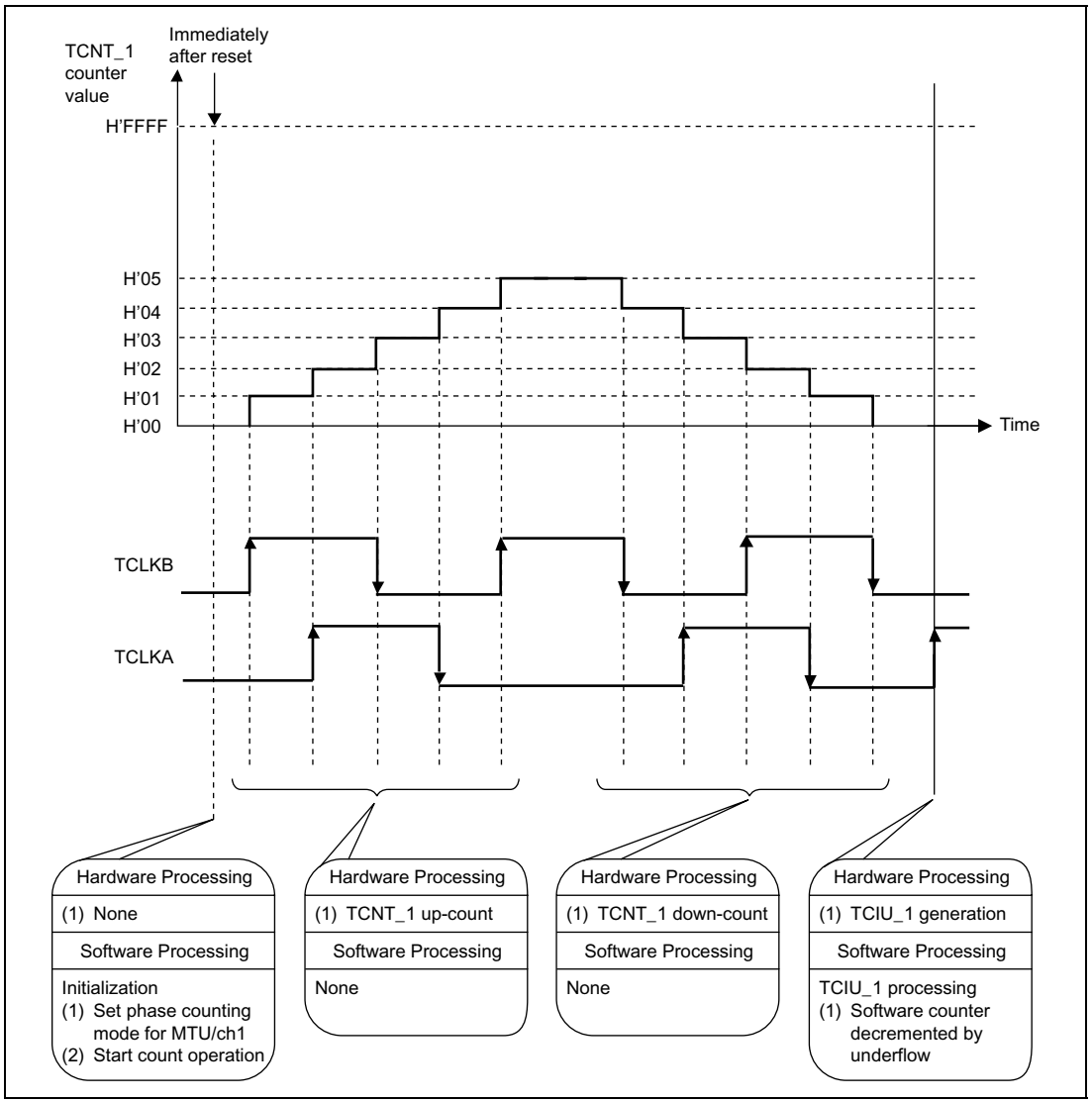


Figure 2.26 Principles of Operation in Phase Counting Mode (1)

(2) The TCNT_1 count is measured on MTU/ch0 output compare occurrence by means of SH7145 hardware and software processing as shown in figure 2.27.

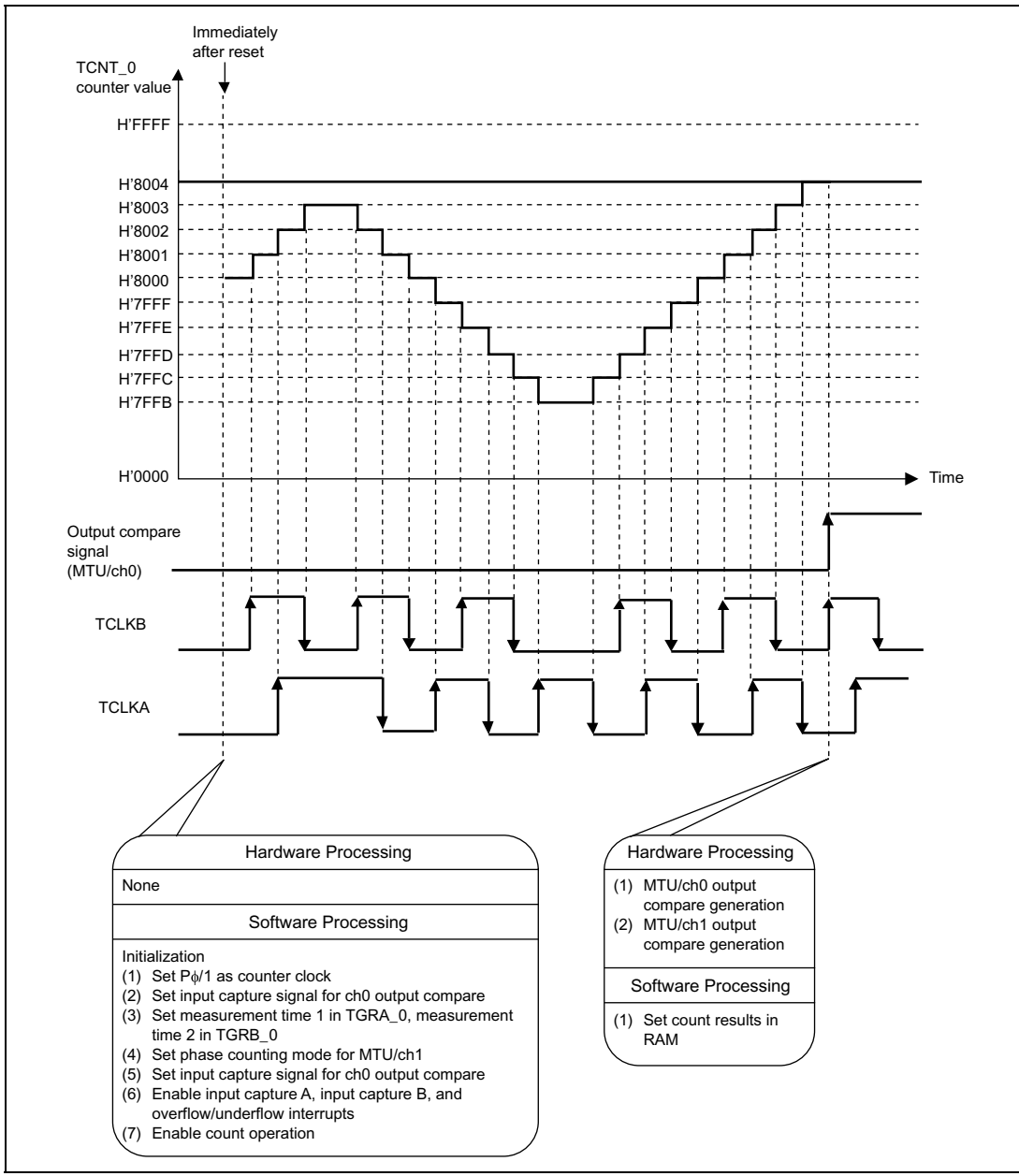


Figure 2.27 Principles of Operation in Phase Counting Mode (2)

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	en2	Initialization of MTU, etc.
Counter value measurement 1	phacnt1	Initiated by TGIA_1. Sets up/down-count result in RAM based on TGRA_1 value. Sets counter period result in RAM based on TGRB_0 value
Counter value measurement 2	phacnt2	Initiated by TGIB_1. Sets up/down-count result in RAM based on TGRB_1 value
Overflow	ovf1	Initiated by TGIV_1. Software counter incrementing
Underflow	unf1	Initiated by TGIU_1. Software counter decrementing

(2) Arguments

Label or Register Name	Function Assignment	Data Length	Module	Input/ Output
msr_tim1 msr_tim2	Used to set timer value for counter measurement time Measurement time is calculated using following equation: Measurement time (ns) = timer value × ϕ period (25.0 ns at 40.0 MHz operation)	Word	Main routine	Input
cnt_data1 cnt_data2	Used to set up/down-count results	Longword	Counter value measurement 1 Counter value measurement 2	Output
p_cycle	Used to set count period result	Longword	Counter value measurement 2	

(3) Internal Registers Used

Register Name	Function	Address	Set Value
P_STBY.MSTCR2	MTU module standby mode clearing	H'FFFF861E	H'd0fd
P_PORTA.PACRL2	Sets PA6 as TCLKA input pin, PA7 as TCLKB input pin	H'FFFF838E	H'5000
P_MTU0.TCR_0	Selection of counter clock and counter clearing source	H'FFFF8260	H'20
P_MTU0.TIORH_0	TIOC0A output compare setting. Setting of TIOBC0B for input capture on TCNT_1 increment/decrement	H'FFFF8262	H'f0
P_MTU0.TIORL_0	TIOC0C output compare setting	H'FFFF8263	H'00
P_MTU0.TGRA_0	Measurement time 1 setting	H'FFFF8268	msr_tim1
P_MTU0.TGRC_0	Measurement time 2 setting	H'FFFF826C	msr_tim2
P_MTU1.TMDR_1	Sets phase counting mode	H'FFFF8281	H'04
P_MTU1.TIOR_1	Setting of TIOC1A/B for input capture on TGRA_0, TGRC_0 output compare occurrence	H'FFFF8282	H'ff
P_MTU1.TIER_1	Enables interrupts by TGIA/B, TCIU_1, TCIV_1	H'FFFF8284	H'73
P_INTC.IPRD	Sets 15 as MTU0, MTU1 interrupt priority level	H'FFFF834E	H'00ff

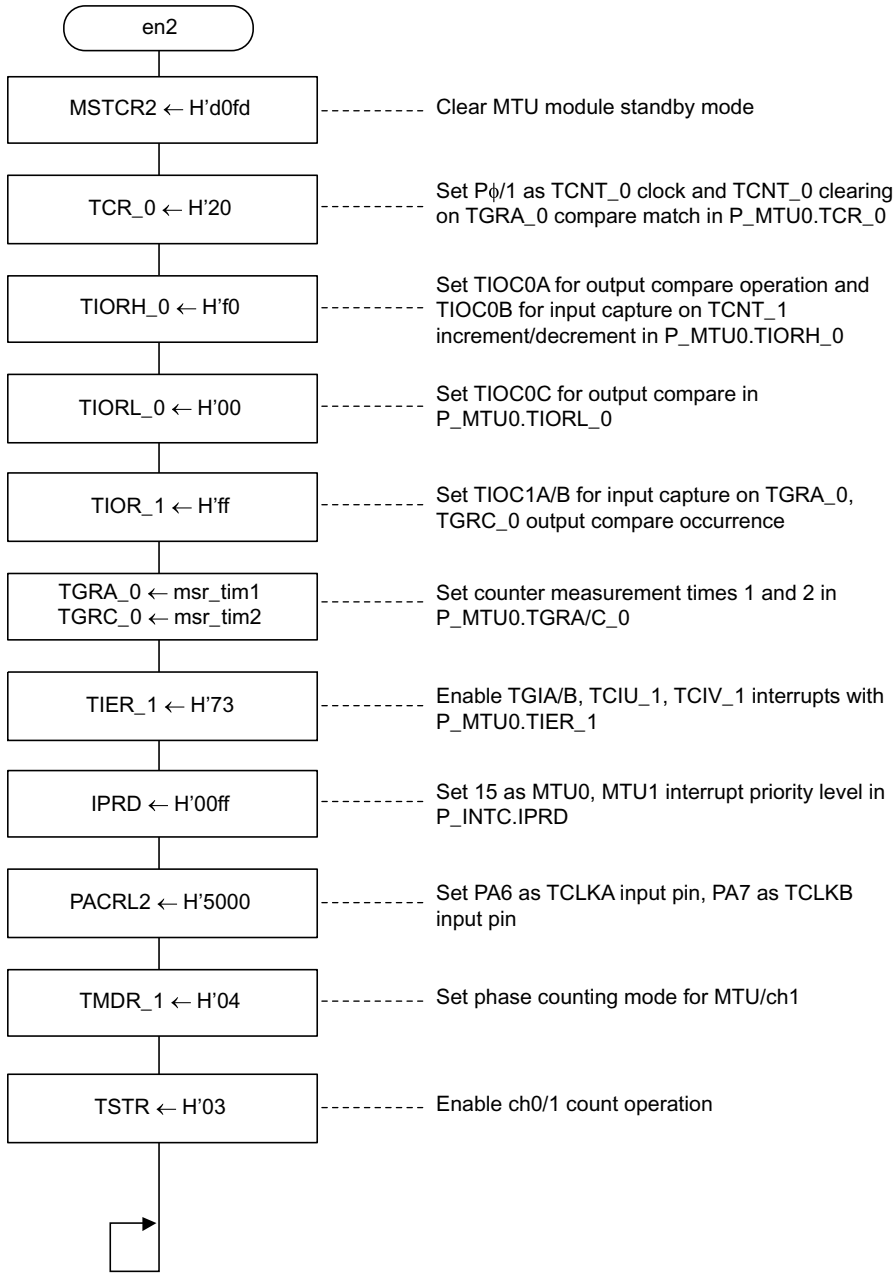
(4) RAM Used

Module	Label	Function Assignment
Counter value measurement 1, 2	wrk	Used as work area for data setting
All modules	cnt	Software counter

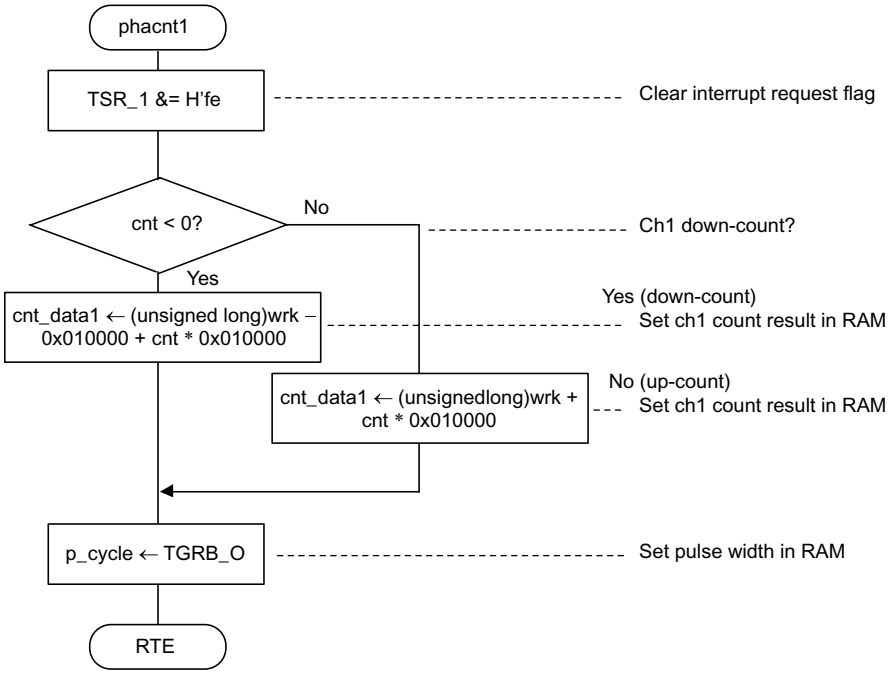
Note: SH7145 header file names are used for register label names.

Flowcharts

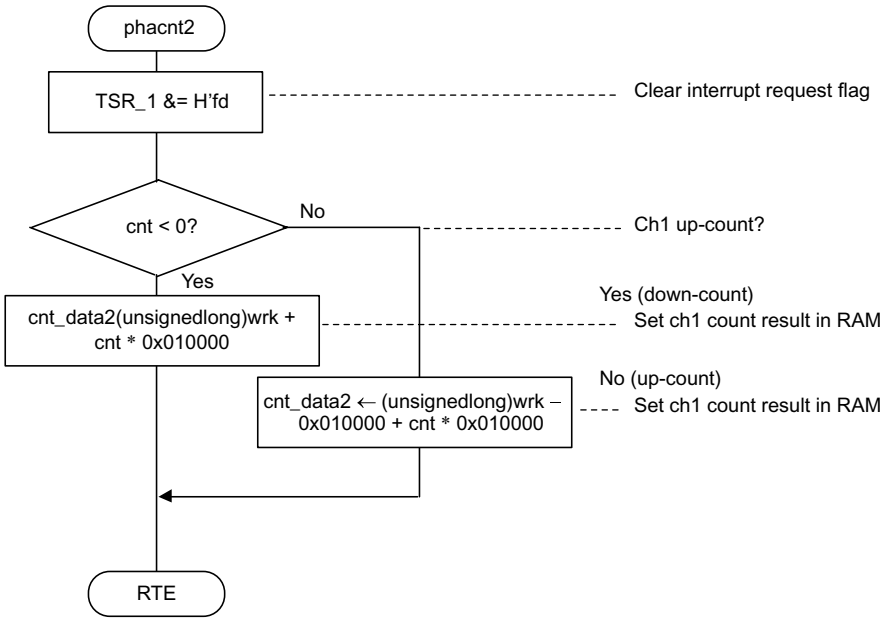
(1) Main routine



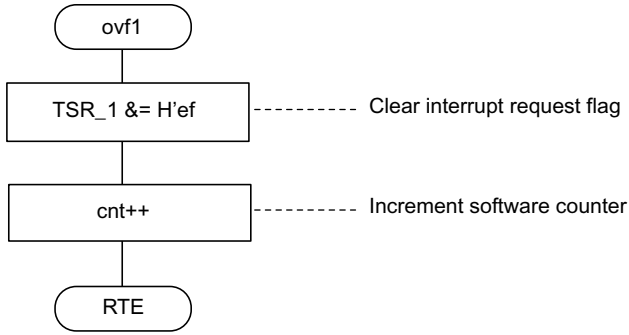
(2) Counter value measurement 1



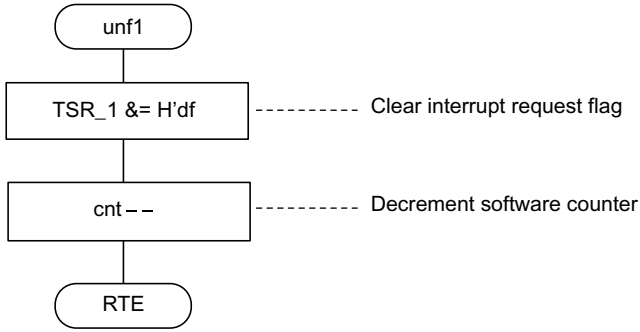
(3) Counter value measurement 2



(4) Overflow



(5) Underflow



Program Listing

```
/*---------------------------------------------------------------------*/
/*                           INCLUDE FILE                              */
/*---------------------------------------------------------------------*/
#include <machine.h>
#include "iodefine_7145F.h"
/*---------------------------------------------------------------------*/
/*                            PROTOTYPE                                */
/*---------------------------------------------------------------------*/
void en2(void);
#pragma interrupt(phacnt1,phacnt2,ovf1,unf1)
/*---------------------------------------------------------------------*/
/*                        RAM ALLOCATION                               */
/*---------------------------------------------------------------------*/
#define msr_tim1    (*(unsigned short *)0xffffe000)
#define msr_tim2    (*(unsigned short *)0xffffe002)
#define cnt_data2   (*(signed long *)0xffffe004)
#define cnt_data1   (*(signed long *)0xffffe008)
#define p_cycle     (*(unsigned long *)0xffffe00c)
#define cnt         (*(signed long *)0xffffe010)
#define wrk         (*(unsigned short *)0xffffe014)
/*---------------------------------------------------------------------*/
/*                          MAIN PROGRAM                               */
/*---------------------------------------------------------------------*/
void en2(void)
 {
   P_STBY.MSTCR2.WORD = 0xd0fd;    /* MTU module stop mode clear */
   P_MTU_0.TCR_0.BYTE = 0x20;      /* timer clear output compare TGRA_0 */
   P_MTU_0.TIORH_0.BYTE = 0xf0;    /* output compare TIOC0B */
   P_MTU0.TIORL_0.BYTE = 0x00;     /* output compare TIOC0C */
   P_MTU1.TIOR_1.BYTE = 0xff;      /* input capture TIOC1A,B */
   P_MTU1.TIER_1.BYTE = 0x73;      /* enable TGIA,TGIB,TCIU,TCIV */
   P_MTU0.TGRC_0 = msr_tim2;       /* set position cycle */
   P_MTU0.TGRA_0 = msr_tim1;       /* set speed cycle */
   INTC.IPRD.WORD = 0x00ff;        /* set interrupt level=15 */
   P_PORTA.PACRL2.WORD = 0x5000;   /* TCLKA,TCLKB select */
   P_MTU1.TMDR_1.BYTE = 0x04;      /* set phase counting mode1 */
   P_MTU34.TSTR.BYTE = 0x03;       /* start MTU/ch0,1 */
   set_imask(0x0);                 /* set imask level=0 */
   while(1);                       /* loop */
  }
```

```
void phacnt1(void)
 {
  P_MTU1.TSR_1.BYTE &= 0xfe;                                  /* clear flag */
  wrk = P_MTU1.TGRA_1;
  if(cnt < 0)                                                 /* down count? */
     cnt_data1 = (unsigned long)wrk-0x010000+cnt*0x010000;    /* set sp */
  else
     cnt_data1 = (unsigned long)wrk+cnt*0x010000;             /* set sp */
     p_cycle = P_MTU0.TGRB_0;                                 /* set width pulse */
 }


void phacnt2(void)
 {
   P_MTU1.TSR_1.BYTE &= 0xfd;                                 /* clear flag */
   wrk = P_MTU1.TGRB_1;
   if(cnt < 0)
      cnt_data2 = (unsigned long)wrk+cnt*0x010000;            /* set po */
   else
      cnt_data2 = (unsigned long)wrk-0x010000+cnt*0x010000;   /* set po */
 }


void ovf1(void)
 {
    P_MTU1.TSR_1.BYTE &= 0xef;                                /* clear flag */
     cnt++;                                                   /* count up */
 }


void unf1(void)
 {
    P_MTU1.TSR_1.BYTE &= 0xdf;                                /* clear flag */
    cnt--;                                                    /* count down */
 }
```

2.8 Externally Triggered Timer Waveform Cutoff

Externally Triggered Timer Waveform Cutoff	MCU: SH7144/45	Functions Used: MTU, POE

Specifications

(1) Timer output waveform cutoff is performed by driving timer output waveforms to the high-impedance state in synchronization with the falling edge of an external signal, as shown in figure 2.28.

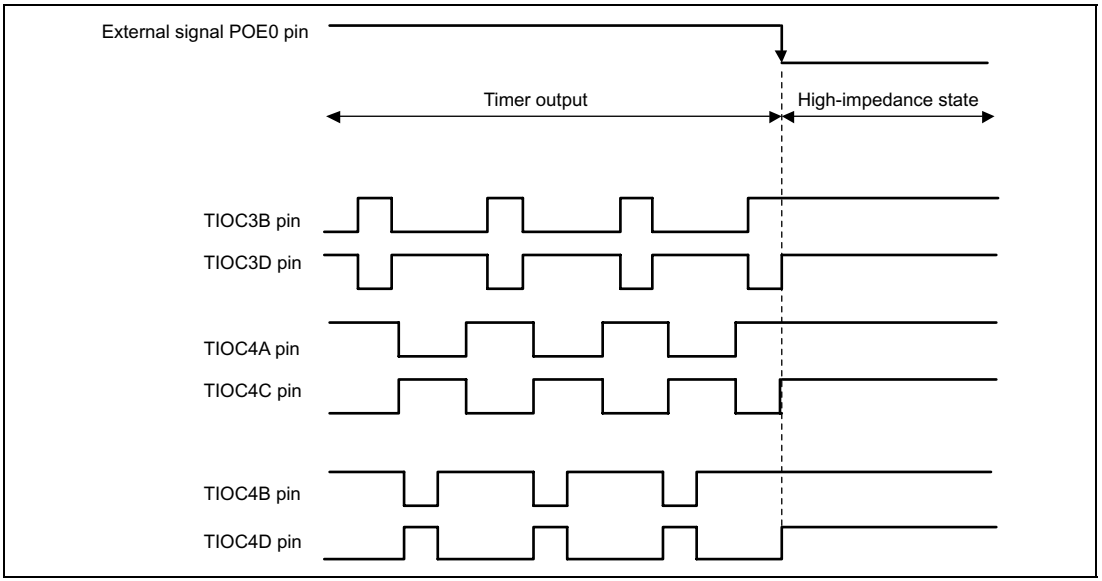


Figure 2.28 Example of Externally Triggered Waveform Cutoff

Functions Used

(1) In this sample task, waveforms output by MTU ch3/4 (reset-synchronized PWM mode) are cut by being driven to the high-impedance state in synchronization with the falling edge of an external signal.

(a) Figure 2.29 shows a block diagram of MTU/ch3 and ch4, and the POE.

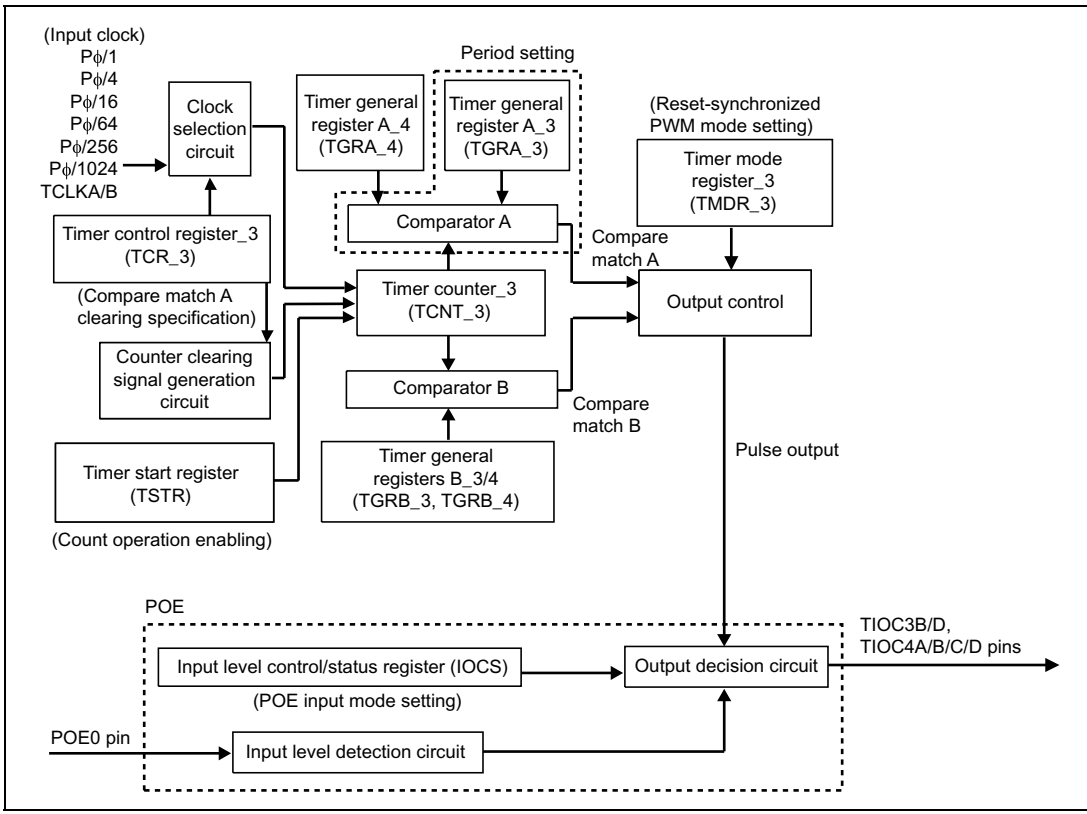


Figure 2.29 Block Diagram of MTU/ch3, ch4 and POE

(2) Table 2.8 shows the function assignments used in this task. Waveform cutoff is performed by assigning MTU and POE functions as shown in the table.

Table 2.8 Function Assignments

Pin or Register Name	Function Assignment
TIOC3B	Pulse output pins
TIOC3D	
TIOC4A	
TIOC4B	
TIOC4C	
TIOC4D	
POE0	Waveform cutoff external signal input pin
TSTR_3	Sets enabling/disabling of ch3 timer counter operation
TCR_3	Selection of ch3 timer counter clearing source and input clock
TMDR_3	Sets reset-synchronized PWM mode for ch3, ch4
TGRA_3	PWM period setting
TGRB_3	PWM duty cycle setting
TGRA_4	
TGRB_4	
TOER	Sets enabling/disabling of TIOC3B/D and TIOC4A/B/C/D pin timer output
ICSR	POE input mode selection

Principles of Operation

(1) Figure 2.30 illustrates the principles of operation of this sample task. Waveform cutoff is performed automatically by hardware. (See section 2.5, Positive-Phase/Negative Phase PWM 3-Phase Output, in this Application Note for the principles of reset-synchronized PWM operation.)

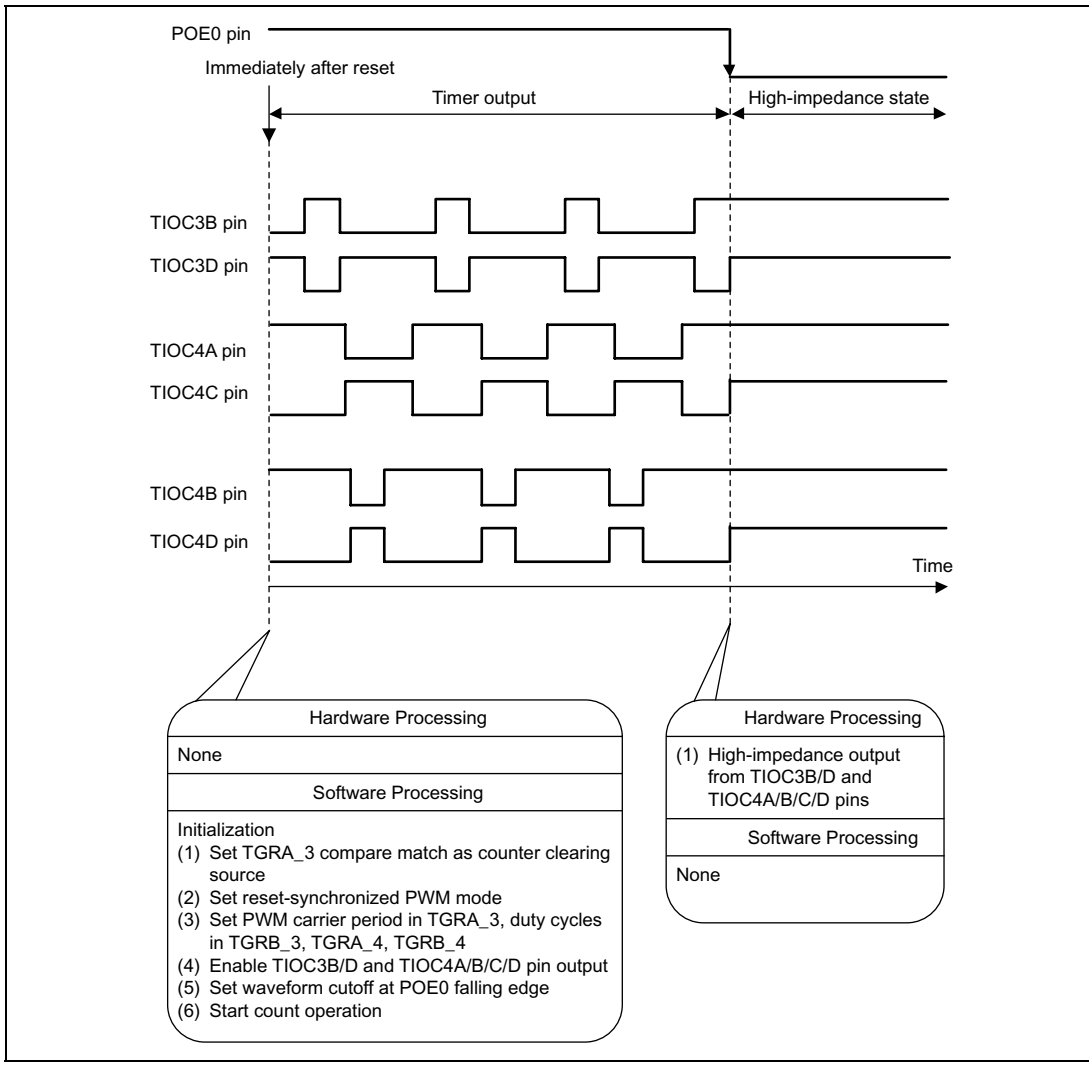


Figure 2.30 Principles of Operation of Externally Triggered Waveform Cutoff

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	down	DC motor control waveform generation

(2) Arguments

Label or Register Name	Function	Data Length	Module	Input/ Output
cycle	PWM period setting	1 word	Main routine	Input
duk1	Used to set TIOC3B/D output waveform transition timing			
duk2	Used to set TIOC4A/C output waveform transition timing			
duk3	Used to set TIOC4B/D output waveform transition timing			

(3) Internal Registers Used

Register Name	Function	Address	Set Value
P_STBY.MSTCR2	MTU module standby mode clearing	H'FFFF861E	H'd0fd
P_PORTE.PEIORL	Sets TIOC3B/D, TIOC4A/B/C/D as output pins	H'FFFF83B4	H'fa00
P_PORTE.PECRL1	Sets TIOC3B/D, TIOC4A/B/C/D as MTU output pins	H'FFFF83B8	H'5544
P_PORTB.PBCR2	Sets PB2 as POE0 pin	H'FFFF839A	H'0020
P_MTU34.TCR_3	Selection of timer counter clearing source and input clock	H'FFFF8200	H'22
P_MTU34.TOCR	Enabling of toggle output synchronized with PWM period, and positive-phase/negative-phase output level setting	H'FFFF820B	H'43
P_MTU34.TGRA_3	PWM period setting	H'FFFF8218	cycle
P_MTU34.TGRB_3	Used to set TIOC3B, TIOC3D output waveform transition timing	H'FFFF821A	duk1
P_MTU34.TGRA_4	Used to set TIOC4A, TIOC4C output waveform transition timing	H'FFFF821C	duk2
P_MTU34.TGRB_4	Used to set TIOC4B, TIOC4D output waveform transition timing	H'FFFF821E	duk3
P_MTU34.TOER	Sets TIOC3B/D, TIOC4A/B/C/D as MTU output pins	H'FFFF820A	H'ff
P_MTU34.TMDR_3	Sets reset-synchronized PWM mode	H'FFFF8202	H'c8
P_MTU.ICSR1	Sets high-impedance output synchronized with falling edge of POE0 pin input signal	H'FFFF83C0	H'0000

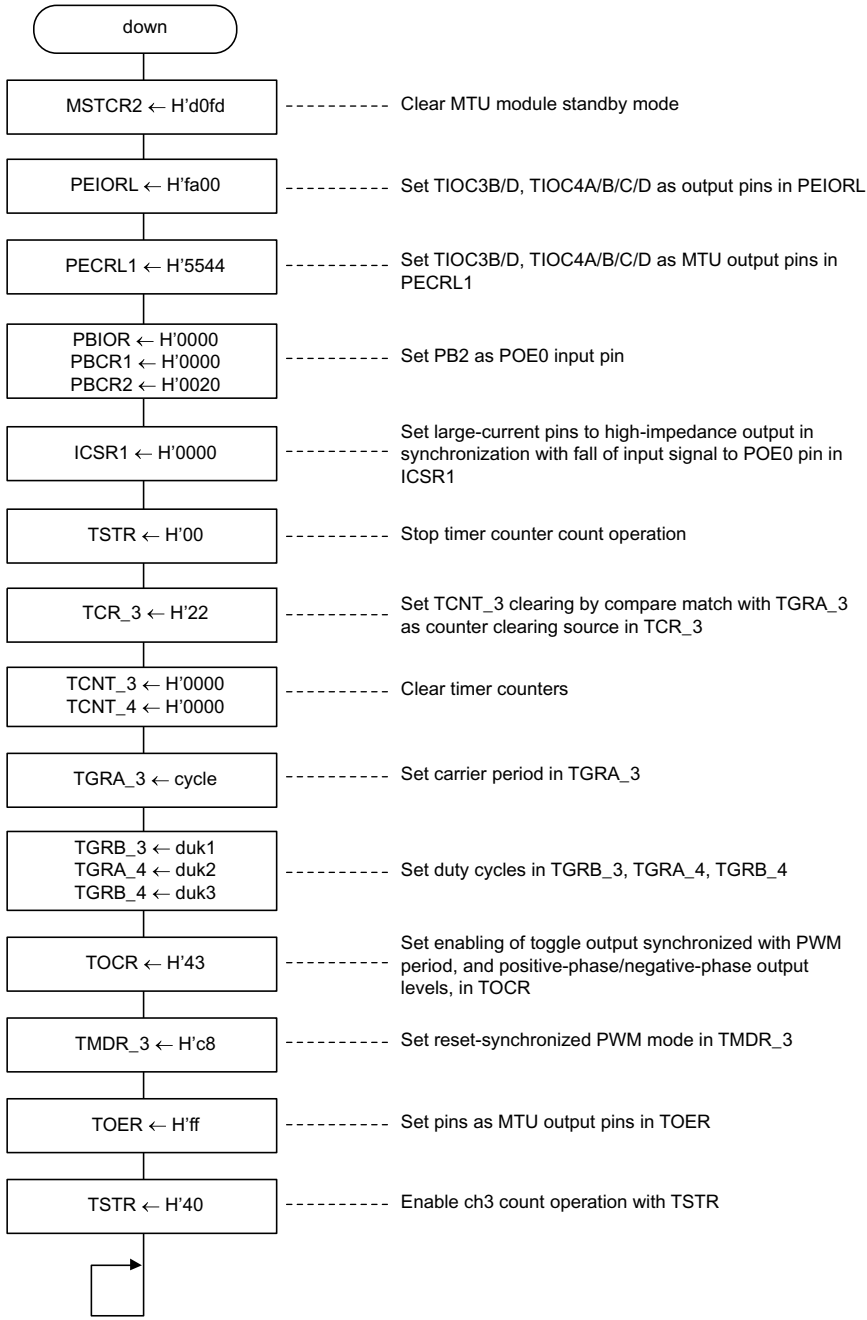
(4) RAM Used

This sample task does not use any RAM apart from the arguments.

Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



Program Listing

```
/*---------------------------------------------------------------------*/
/*                          INCLUDE FILE                               */
/*---------------------------------------------------------------------*/
#include <machine.h>
#include "iodefine_7145F.h"
/*---------------------------------------------------------------------*/
/*                           PROTOTYPE                                 */
/*---------------------------------------------------------------------*/
void down(void);
/*---------------------------------------------------------------------*/
/*                         RAM ALLOCATION                              */
/*---------------------------------------------------------------------*/
#define cycle                     (*(unsigned short *)0xffffe000)
#define duk1                      (*(unsigned short *)0xffffe002)
#define duk2                      (*(unsigned short *)0xffffe004)
#define duk3                      (*(unsigned short *)0xffffe006)
/*---------------------------------------------------------------------*/
/*                        MAIN PROGRAM                                 */
/*---------------------------------------------------------------------*/
void down(void)
 {

   P_STBY.MSTCR2.WORD = 0xd0fd;   /* MTU module standby mode clear */
   P_PORTE.PEIORL.WORD = 0xfa00;  /* TIOC3B/D,TIOC4A/B/C/D=output */
   P_PORTE.PECRL1.WORD = 0x5544;  /* TIOC3B/D,TIOC4A/B/C/D=output */

   P_PORTB.PBIOR.WORD = 0x0000;   /* POE0 enable */
   P_PORTB.PBCR1.WORD = 0x0000;
   P_PORTB.PBCR2.WORD = 0x0020;
   P_MTU.ICSR1.WORD = 0x0000;     /* stop timer POE0 falling edge */
   P_MTU.OCSR.WORD = 0x0000;
   P_MTU34.TSTR.BYTE = 0x00;
   P_MTU34.TCR_3.BYTE = 0x22;     /* timer clear input capture TGRA_3 */
   P_MTU34.TCNT_3 = 0x0000;       /* set timer counter 0x0000 */
   P_MTU34.TCNT_4 = 0x0000;
   P_MTU34.TGRA_3 = cycle;        /* set PWM period */
   P_MTU34.TGRB_3 = duk1;         /* set duty */
   P_MTU34.TGRA_4 = duk2;
   P_MTU34.TGRB_4 = duk3;
   P_MTU34.TOCR.BYTE = 0x43;      /* set output level */
   P_MTU34.TMDR_3.BYTE = 0xc8;    /* Reset-synchronized PWM mode */
   P_MTU34.TOER.BYTE = 0xff;      /* set timer3,4 output */
   P_MTU34.TSTR.BYTE = 0x40;      /* start timer3 */
     while(1);                    /* loop*/
  }
```

2.9 DC Motor Control Signal Output

DC Motor Control Signal Output	MCU: SH7144/45	Functions Used: MTU (Gate Control Register)

Specifications

(1) Waveforms necessary for DC brushless motor control are output as shown in figure 2.31. The output waveforms are output by chopping the respective pin gate signals and reset-synchronized PWM output.

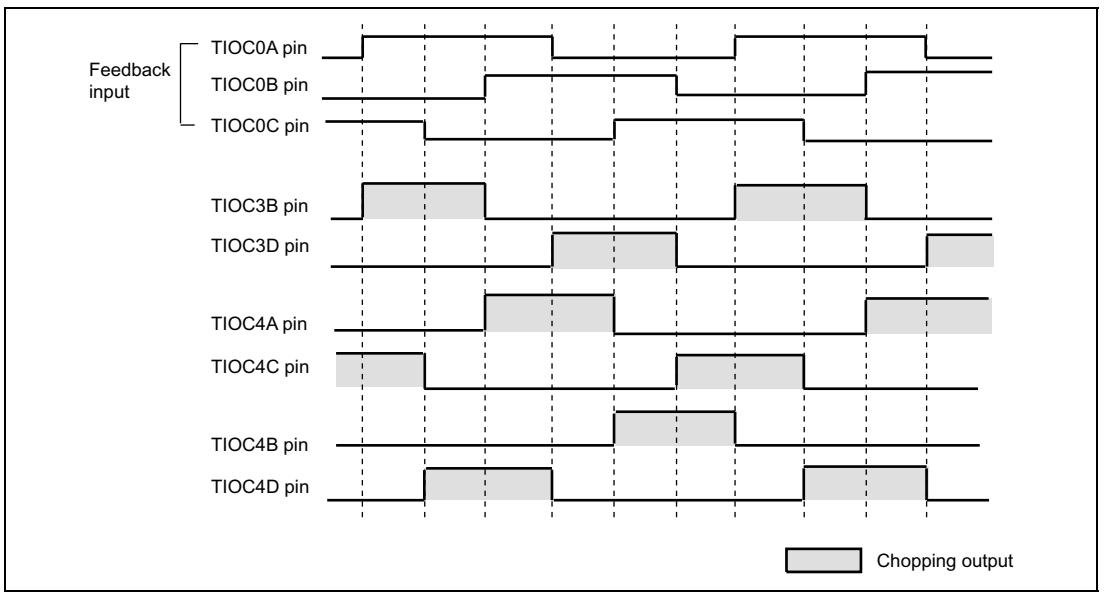


Figure 2.31 Example of DC Brushless Motor Control Signal Output

Functions Used

(1) In this sample task, MTU channels 3 and 4 are used in combination, and 3-phase PWM waveform output is performed with one common transition point in the relationship between the positive phase and negative phase. Gate signals generated from the generated waveforms and feedback input are chopped and output.

(a) Figure 2.32 shows a block diagram of the MTU as used in this sample task.

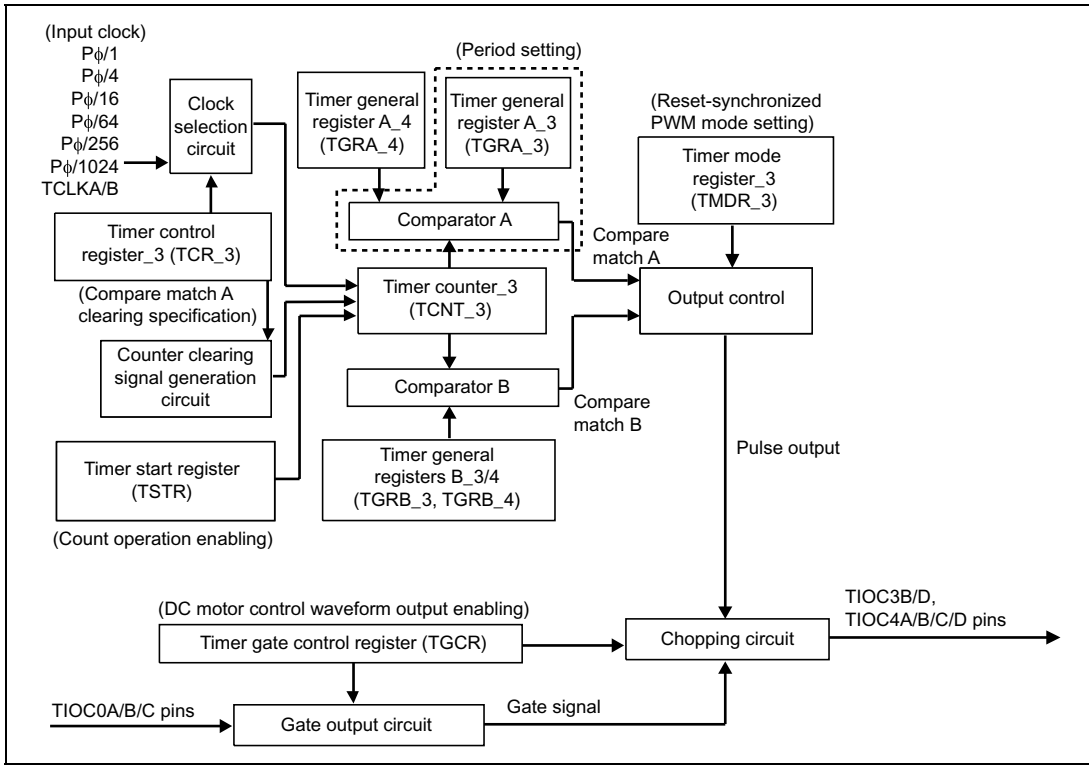


Figure 2.32 Block Diagram of MTU/ch3, ch4

(2) Table 2.9 shows the function assignments used in this task. DC motor control waveform output is performed by assigning MTU functions as shown in the table.

Table 2.9 Function Assignments

Pin or Register Name	Function Assignment
TIOC3B	PWM pulse output pins
TIOC3D	
TIOC4A	
TIOC4B	
TIOC4C	
TIOC4D	
TIOC0A	Feedback signal input pins
TIOC0B	
TIOC0C	
TSTR	Enabling/disabling of ch3, ch4 timer counter operation
TCR_3	Selection of ch3 timer counter clearing source and input clock
TMDR_3	Ch3, ch4 set to operate in reset-synchronized PWM mode
TGRA_3	PWM period setting
TGRB_3	Output waveform transition timing setting
TGRC_3	
TGRD_3	
TGRA_4	
TGRB_4	
TOER	Enabling/disabling of TIOC3B/D and TIOC4A/B/C/D pin timer output
TGCR	Enabling/disabling of DC motor control waveform output

Principles of Operation

(1) Figure 2.33 illustrates the principles of operation of this sample task. DC motor control waveform output is performed automatically by hardware. (See section 2.5, Positive-Phase/Negative Phase PWM 3-Phase Output, in this Application Note for the principles of reset-synchronized PWM operation.)

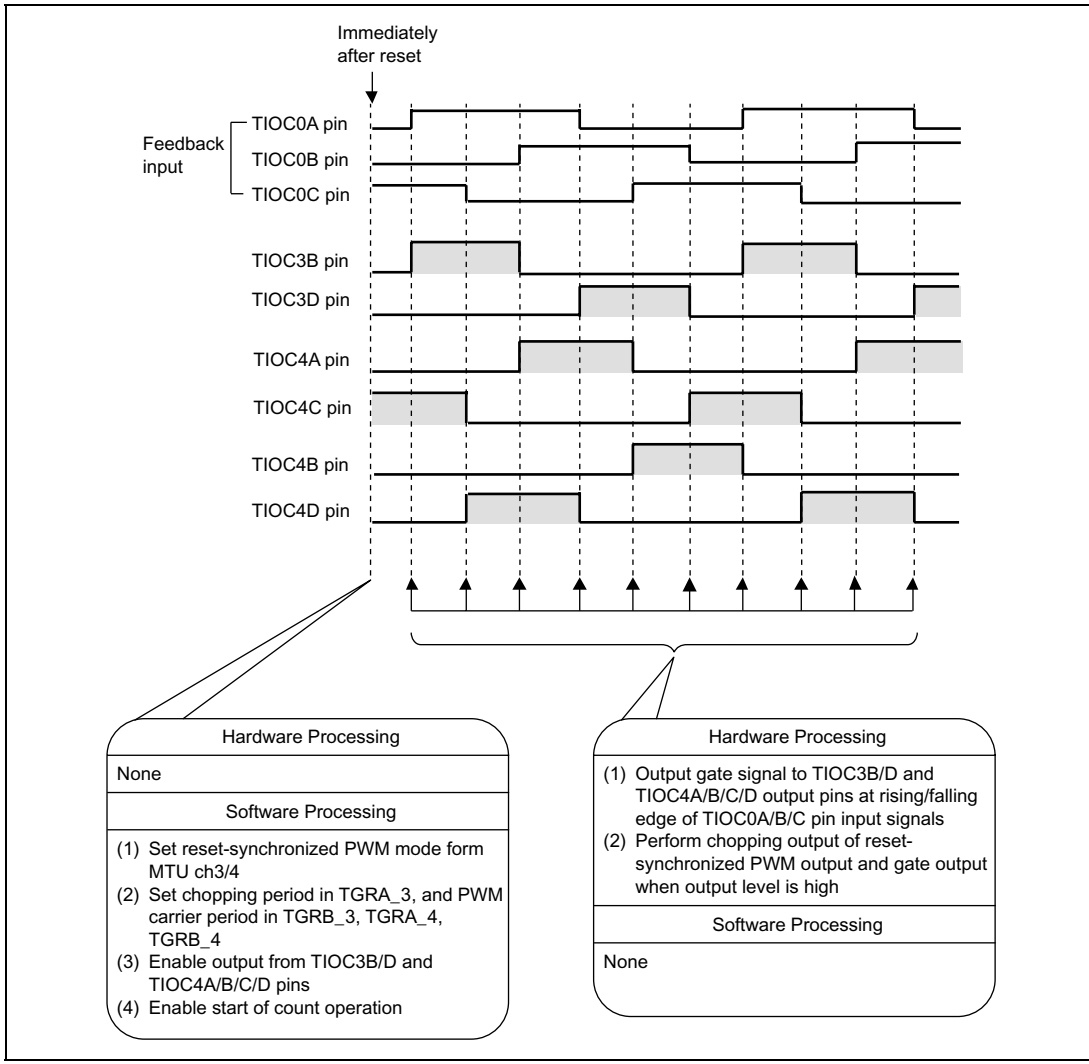


Figure 2.33 Principles of Operation of DC Motor Control Signal Output

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	dc_3	DC motor control waveform generation

(2) Arguments

Label or Register Name	Function	Data Length	Module	Input/ Output
cycle	Used to set timer value for PWM pulse period	1 word	Main routine	Input
duk1	Used to set TIOC3B/3D output waveform transition timing			
duk2	Used to set TIOC4A/4C output waveform transition timing			
duk3	Used to set TIOC4B/4D output waveform transition timing			

(3) Internal Registers Used

Register Name	Function	Address	Set Value
P_STBY.MSTCR2	MTU module standby mode clearing	H'FFFF861E	H'd0fd
P_PORTE.PEIORL	Sets TIOC3B/D, TIOC4A/B/C/D as output pins	H'FFFF83B4	H'fa00
P_PORTE.PECRL1	Sets TIOC3B/D, TIOC4A/B/C/D as MTU input/output pins	H'FFFF83B8	H'5544
P_PORTE.PECRL2	Sets TIOC0A, TIOC0B,TIOC0C as MTU input pins	H'FFFF83BA	H'0015
P_MTU34.TCR_3	Selection of timer counter clearing source and input clock	H'FFFF8200	H'22
P_MTU34.TOCR	Enabling of toggle output synchronized with PWM period, and positive-phase/negative-phase output level setting	H'FFFF820B	H'03
P_MTU34.TGRA_3	PWM carrier period setting	H'FFFF8218	cycle
P_MTU34.TGRB_3	Used to set TIOC3B, TIOC3D output waveform transition timing	H'FFFF821A	duk1
P_MTU34.TGRA_4	Used to set TIOC4A, TIOC4C output waveform transition timing	H'FFFF821C	duk2
P_MTU34.TGRB_4	Used to set TIOC4B, TIOC4D output waveform transition timing	H'FFFF821E	duk3
P_MTU34.TOER	Sets TIOC3B/3D, TIOC4A/4B/4C/4D as MTU output pins	H'FFFF820A	H'ff
P_MTU34.TMDR_3	Sets reset-synchronized PWM mode	H'FFFF8202	H'c8
P_MTU34.TGCR	Enables DC motor control waveform output	H'FFFF820D	H'f0

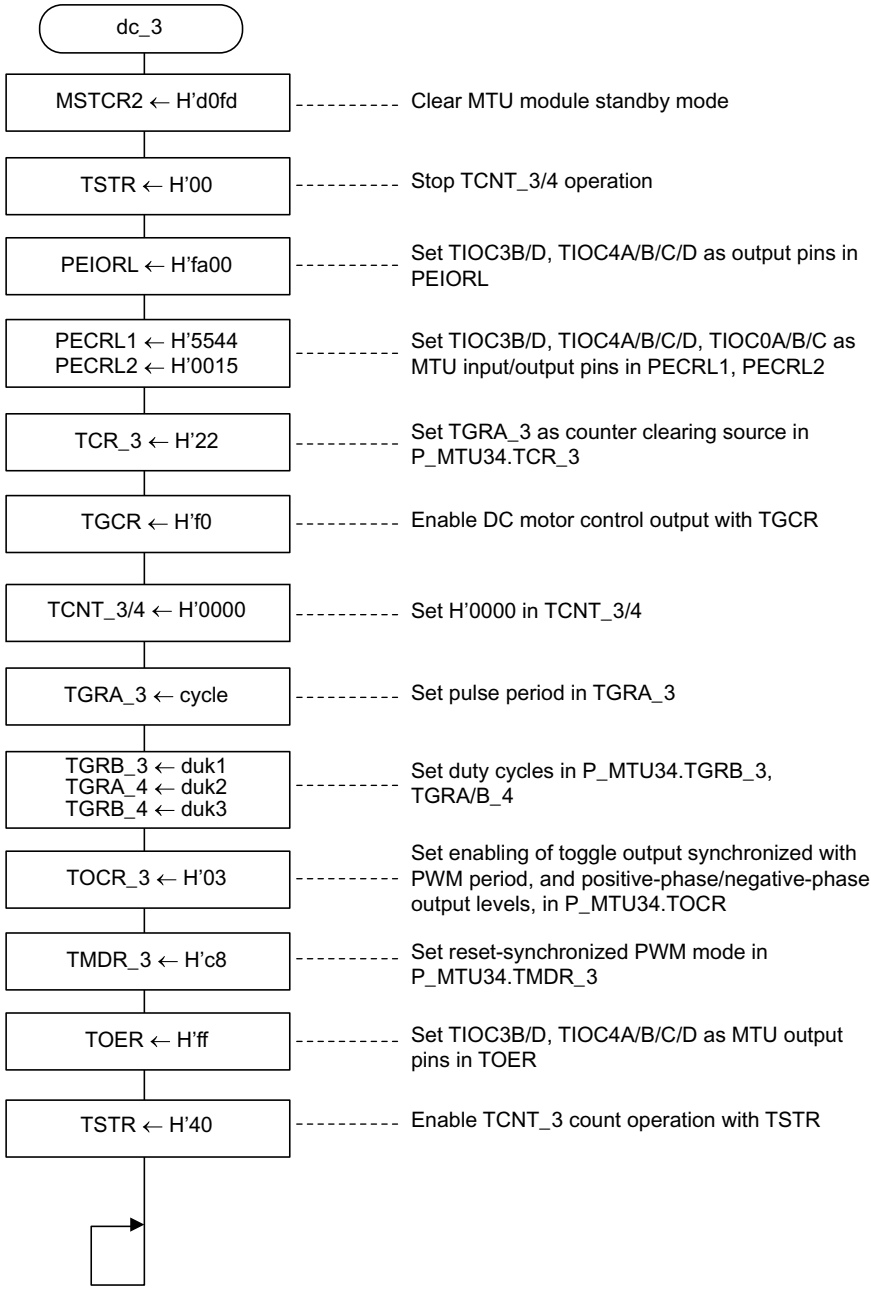
(4) RAM Used

This sample task does not use any RAM apart from the arguments.

Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



Program Listing

```
/*------------------------------------------------------------------------*/
/*                                INCLUDE FILE                            */
/*------------------------------------------------------------------------*/
#include <machine.h>
#include "iodefine_7145F.h"
/*------------------------------------------------------------------------*/
/*                                PROTOTYPE                               */
/*------------------------------------------------------------------------*/
void dc_3(void);
/*------------------------------------------------------------------------*/
/*                              RAM ALLOCATION                            */
/*------------------------------------------------------------------------*/
#define cycle      (*(unsigned short *)0xffffe000)
#define duk1       (*(unsigned short *)0xffffe002)
#define duk2       (*(unsigned short *)0xffffe004)
#define duk3       (*(unsigned short *)0xffffe006)
/*------------------------------------------------------------------------*/
/*                               MAIN PROGRAM                             */
/*------------------------------------------------------------------------*/
void dc_3(void)
 {
  P_STBY.MSTCR2.WORD = 0xd0fd;    /* MTU module standby mode clear */
  P_MTU34.TSTR.BYTE = 0x00;       /* Stop timer counter */
  P_PORTE.PEIORL.WORD = 0xfa00;   /* TIOC3B/D,TIOC4A/B/C/D output */
  P_PORTE.PECRL1.WORD = 0x5544;   /* TIOC3B/D,TIOC4A/B/C/D output */
  P_PORTE.PECRL2.WORD = 0x0015;   /* TIOC0A/B/C = input */
  P_MTU34.TCR_3.BYTE = 0x22;      /* Counter clear by input capture TGRA_3 */
  P_MTU34.TGCR.BYTE = 0xf0;       /* for DC brushless motor control */
  P_MTU34.TCNT_3 = 0x0000;        /* Timer counter3 = 0 */
  P_MTU34.TCNT_4 = 0x0000;        /* Timer counter4 = 0 */
  P_MTU34.TGRA_3 = cycle;         /* Set carrier Period  */
  P_MTU34.TGRB_3 = duk1;          /* Set duty */
  P_MTU34.TGRA_4 = duk2;
  P_MTU34.TGRB_4 = duk3;
  P_MTU34.TOCR.BYTE = 0x03;       /* Set output level */
  P_MTU34.TMDR_3.BYTE = 0xc8;     /* Reset-synchronized pwm mode */
  P_MTU34.TOER.BYTE = 0xff;       /* Enable timer3/4 output */
  P_MTU34.TSTR.BYTE = 0x40;       /* Start timer3 */
  while(1);                       /* Loop */
 }
```

2.10 Start of A/D Conversion by MTU

Start of A/D Conversion by MTU	MCU: SH7144/45	Functions Used: MTU, A/D Converter

Specifications

(1) Four channel voltages are input and subjected to A/D conversion as shown in figure 2.34.

(2) Single-cycle scan mode is used for A/D conversion, with A/D conversion performed consecutively on channels 0 to 3.

(3) A/D converter activation is performed by an MTU/ch0 TGRA_0 compare match.

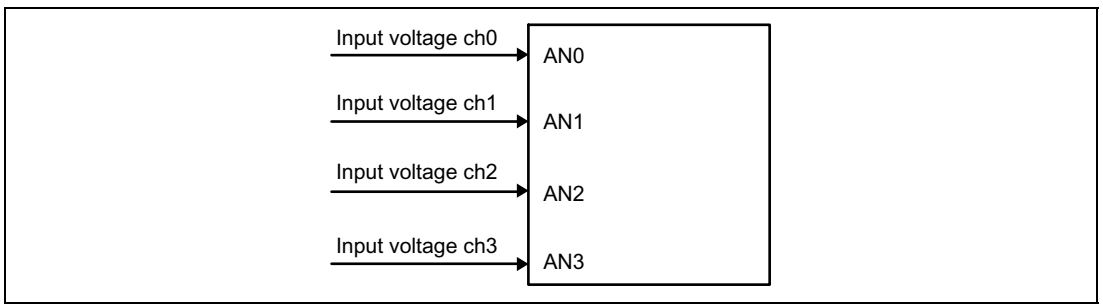


Figure 2.34 Block Diagram of Voltage Measurement by SH7145

Functions Used

(1) In this sample task, the A/D converter is activated by an A/D conversion start request from the MTU.

(a) Figure 2.35 shows a block diagram of ch0. Ch0 A/D conversion is initiated using the following functions.

- A function that starts A/D conversion by means of an MTU compare match, without software intervention
- A function that outputs pulses automatically by hardware without software intervention (output compare)

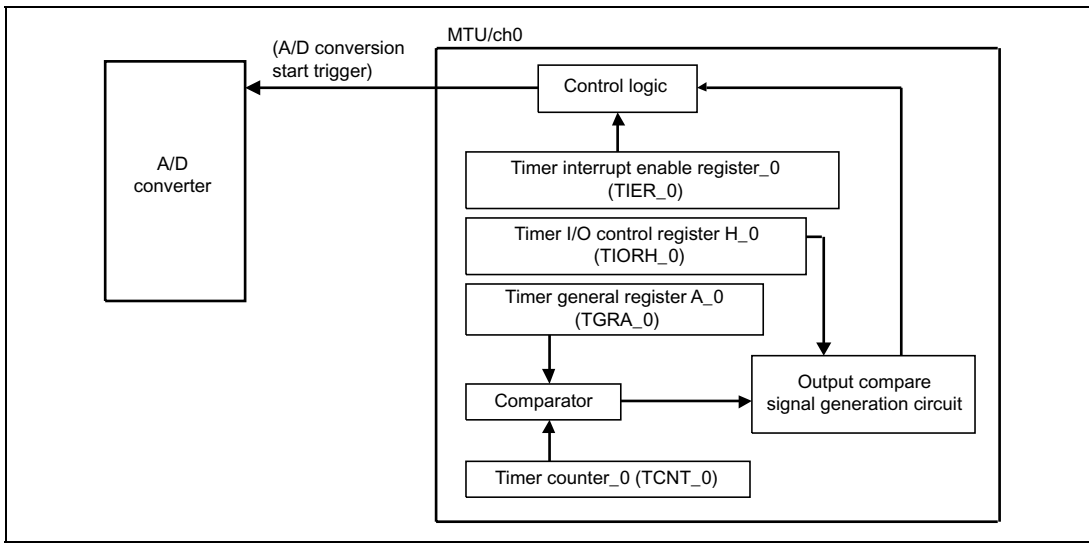


Figure 2.35 Block Diagram of SH7145 ch0

(b) Figure 2.36 shows a block diagram of the A/D converter. The A/D converter performs conversion from analog to digital form using the following function.

- A function that performs A/D conversion once on the specified channels (ch0 to ch3) (single-cycle scan mode)

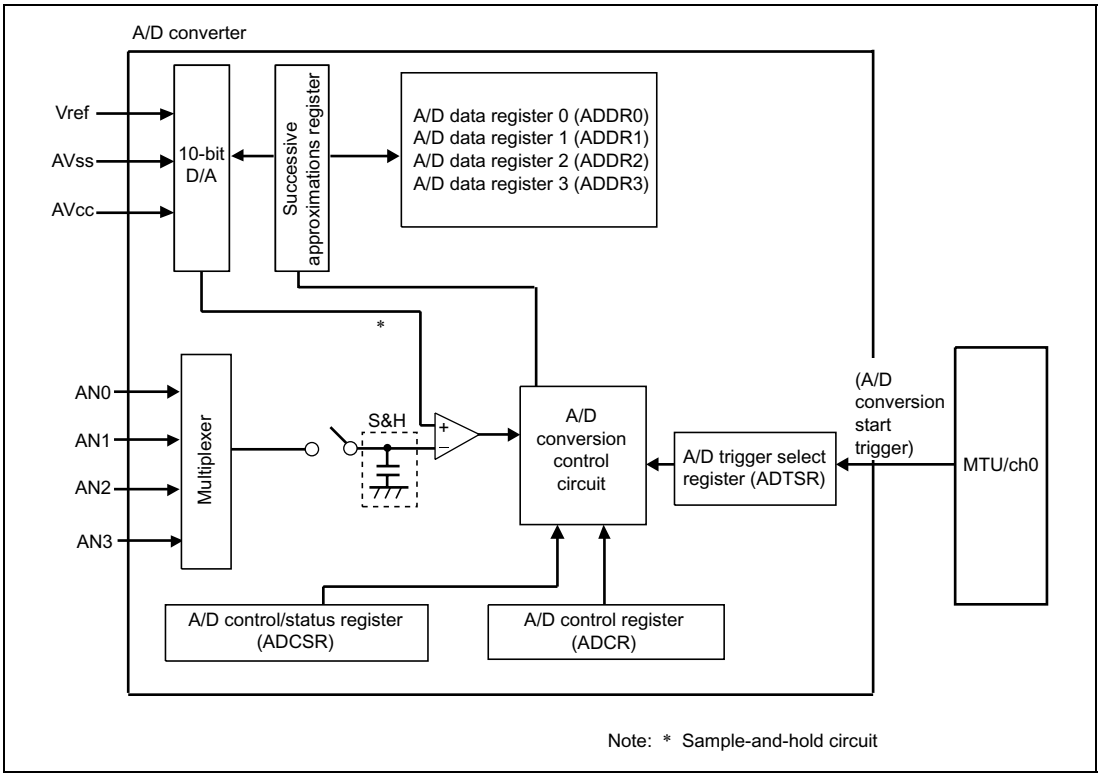


Figure 2.36 Block Diagram of Voltage Measurement by SH7145

(2) Table 2.10 shows the function assignments used in this sample task.

Table 2.10 Function Assignments

Pin or Register Name	Function Assignment
AN0 to AN3	Analog input pins
TCR_0	Selection of counter clearing source
TIER_0	Enables A/D conversion start request generation
TGRA_0	A/D conversion sampling period setting
ADCR	A/D conversion mode and measurement pin setting
ADCSR	Selection of conversion time and activation source
ADDR0 to ADDR3	Storage of A/D conversion results

Principles of Operation

(1) Figure 2.37 illustrates the principles of operation of this sample task. As shown in the figure, the A/D converter is activated by a TGRA_0 compare match and sequentially measures voltages input to AN0 through AN3.

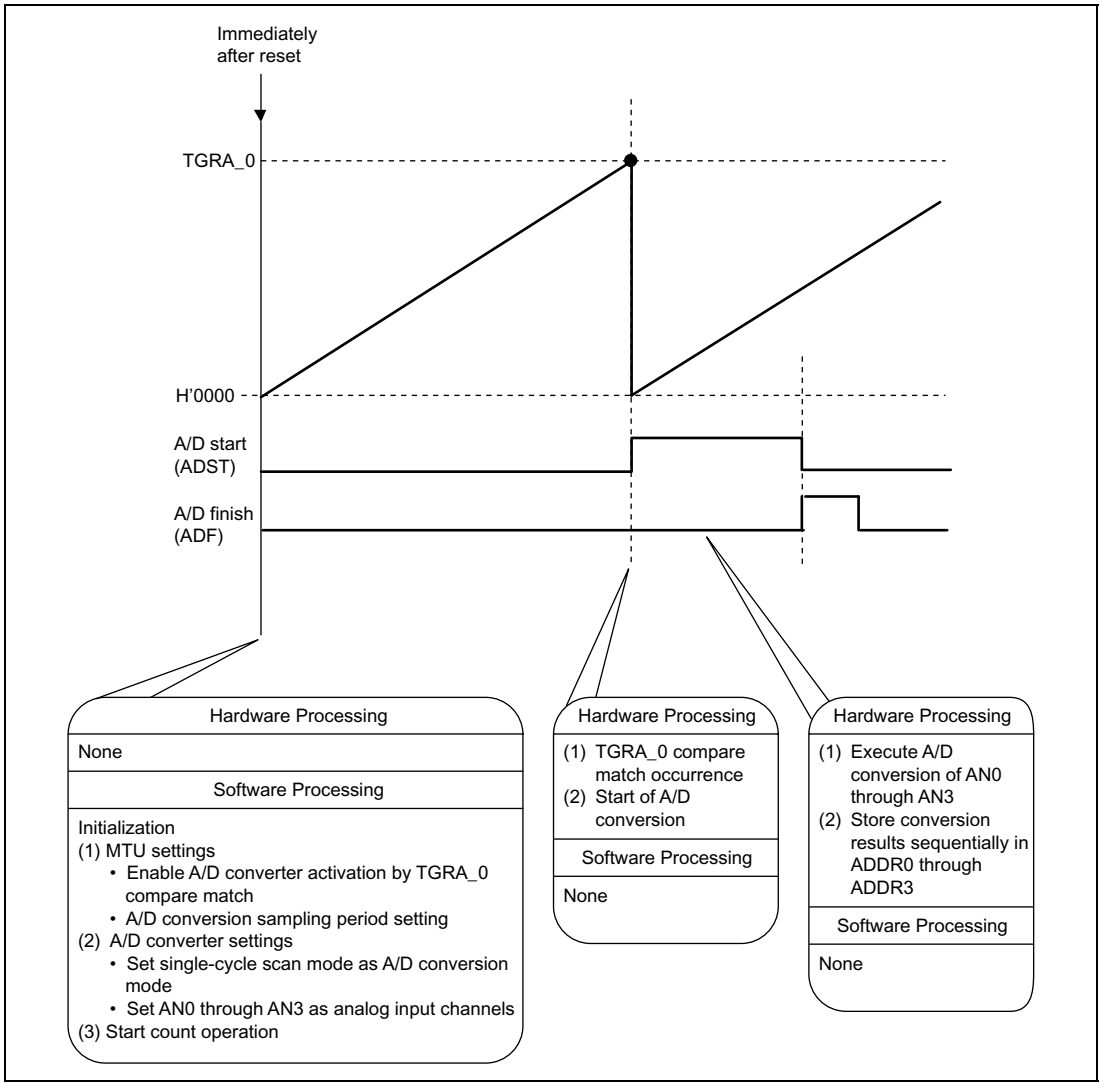


Figure 2.37 Principles of Operation of A/D Conversion Initiation by MTU

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	main	A/D converter activation by MTU

(2) Internal Registers Used

Register Name	Function	Address	Set Value
P_STBY.MSTCR2	Module standby mode clearing (MTU, A/D converter)	H'FFFF861E	H'd0ed
P_MTU0.TCR_0	Selection of TCNT_0 counter clock, and setting of output compare A as TCNT_0 counter clearing source	H'FFFF8260	H'20
P_MTU0.TIORH_0	Sets TGRA_0 for output compare	H'FFFF8262	H'00
P_MTU0.TIER_0	Enables generation of A/D conversion start requests from MTU	H'FFFF8264	H'c1
P_MTU0.TGRA_0	Sets 1 ms as A/D conversion sampling period	H'FFFF8268	H'9c40
P_AD.ADCR_0	Sets MTU conversion start trigger as A/D conversion mode (single-cycle scan mode) activation source	H'FFFF8488	H'87
P_AD.ADCSR_0	Setting of A/D conversion channels (AN0 to AN3) and conversion time	H'FFFF8480	H'1b
P_AD.ADTSR	Sets enabling of start of A/D0 module conversion by MTU A/D conversion start trigger signal	H'FFFF87F4	H'02

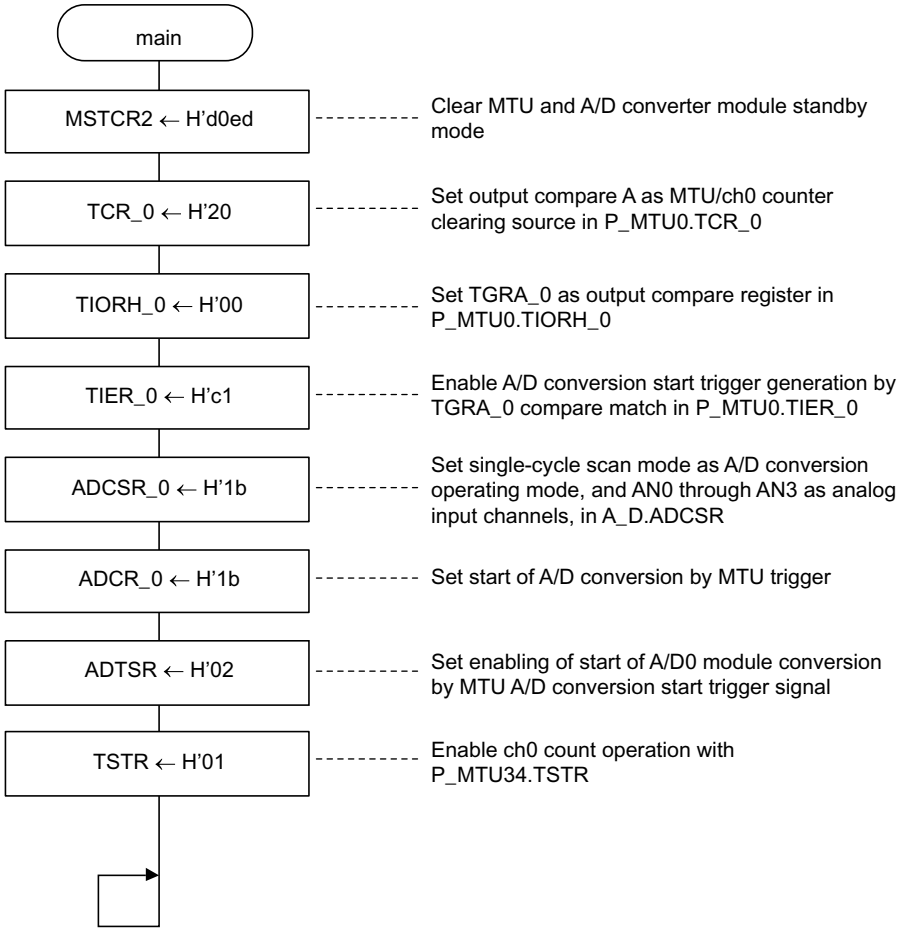
(4) RAM Used

This sample task does not use any RAM apart from the arguments.

Note: SH7145 header file names are used for register label names.

Flowcharts

(1) Main routine



Program Listing

```
/************************************************************************/
/*                              INCLUDE FILE                            */
/************************************************************************/
#include <machine.h>
#include "iodefine_7145F.h"
/************************************************************************/
/*                               PROTOTYPE                              */
/************************************************************************/
void main(void);
/************************************************************************/
/*                              MAIN PROGRAM                            */
/************************************************************************/
void main(void)
 {
    P_STBY.MSTCR2.WORD = 0xd0ed;          /* Clear Module standby mode */

    P_MTU0.TCR_0.BYTE = 0x20;             /* clock=Pϕ/1,counter clear TGRA_0 */
    P_MTU0.TIORH_0.BYTE = 0x00;
    P_MTU0.TIER_0.BYTE = 0xc1;            /* enable TGIA */
    P_MTU0.TCNT_0 = 0x0000;
    P_MTU0.TGRA_0 = 0x9c40;               /* A/D sampling period = 1.0ms */

    P_AD.ADCR_0.BYTE = 0x87;              /* 1cycle scan mode */
    P_AD.ADCSR_0.BYTE = 0x1b;             /* 4channel scan mode */
    P_AD.ADTSR.BYTE = 0x02;
    P_MTU34.TSTR.BYTE = 0x01;             /* Start timer counter */

    set_imask(0x0);
    while(1);
 }
```

2.11 RAM Monitoring Using DMAC

RAM Monitoring Using DMAC	MCU: SH7144/45	Functions Used: SCI, DMAC

Specifications

(1) Using the SH7145's SCI in asynchronous mode, a RAM reference address (4-byte data) transmitted from the console is received, and the contents of that address are fetched from RAM and transmitted to the console via the SCI, as shown in figure 2.38.

(2) The transfer protocol settings used are 19200 bps, 8-bit data, one stop bit, and no parity.

(3) For data transfer from RDR to RAM, DMAC direct addressing mode is used, and RDR receive data is stored in RAM, as shown in figure 2.39.

(4) For data transfer from RAM to TDR, DMAC indirect addressing mode is used and the following operations are executed, as shown in figure 2.40.

(a) Data stored in RAM is stored in a temporary buffer in the DMAC, and data is fetched from RAM using this as the address.

(b) Fetched data is transferred sequentially to TDR one byte at a time.

(5) The DMAC transfer conditions are as shown in tables 2.11 and 2.12.

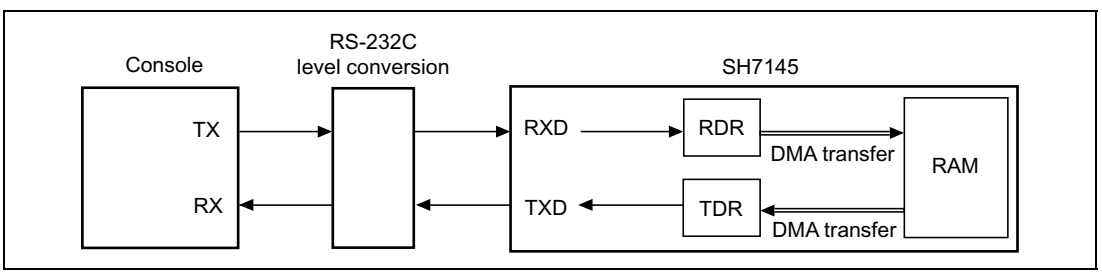


Figure 2.38 Block Diagram of SCI Transfer of Data in RAM by SH7145

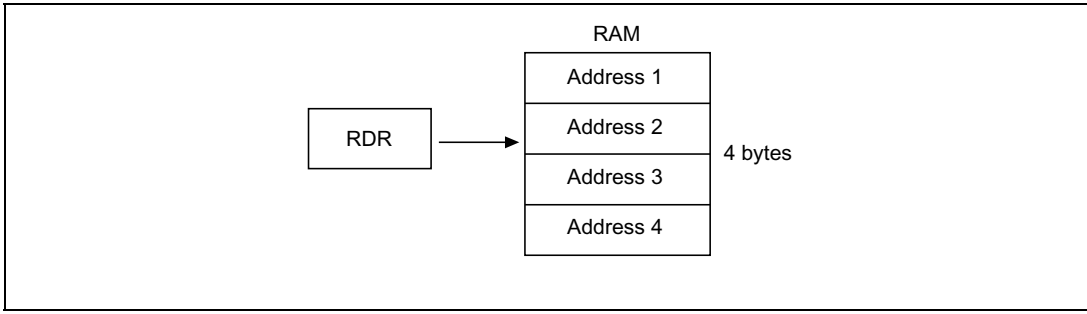


Figure 2.39 Data Transfer Using DMAC (Transfer Source Direct Addressing)

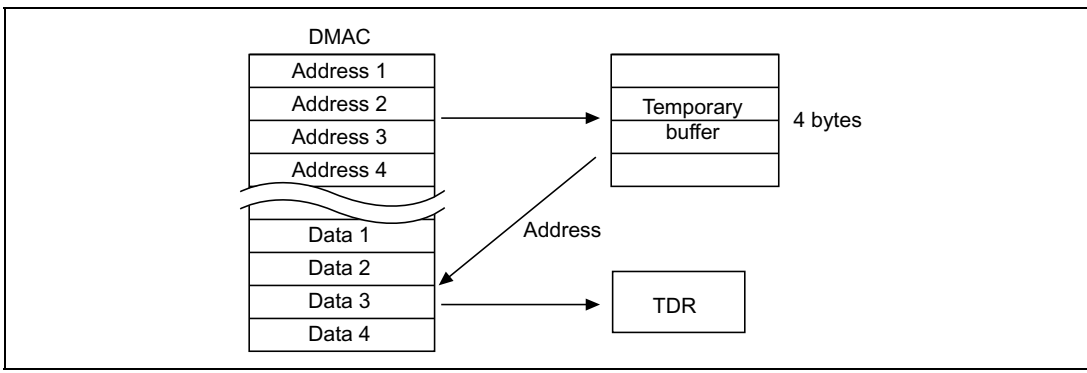


Figure 2.40 Data Transfer Using DMAC (Transfer Source Indirect Addressing)

Table 2.11 DMAC Transfer Conditions for SCI Reception (RDR → RAM)

Condition	Description
DMAC channel	Channel 0
Transfer source	On-chip SCI channel 0
Transfer destination	On-chip RAM
Number of transfers	4
Transfer source address	Fixed
Transfer destination address	Incremented
Transfer request source	SCI channel 0
Bus mode	Cycle-steal
Transfer unit	Byte

Table 2.12 DMAC Transfer Conditions for SCI Reception (RAM → TDR)

Condition	Description
DMAC channel	Channel 3
Transfer source	On-chip RAM
Transfer destination	On-chip SCI channel 0
Number of transfers	1
Transfer source address	Incremented
Transfer destination address	Fixed
Transfer request source	SCI channel 0
Bus mode	Cycle-steal
Transfer unit	Byte

Functions Used

In this sample task, RAM monitoring is performed using the SCI and DMAC.

(a) Figure 2.41 shows a block diagram of SCI transmission circuits. In this task, console data transmission is performed using the following SCI functions.

— A function that performs data communication asynchronously, with synchronization performed character by character (asynchronous mode)

— A function that generates an interrupt on completion of transmission (TEI interrupt)

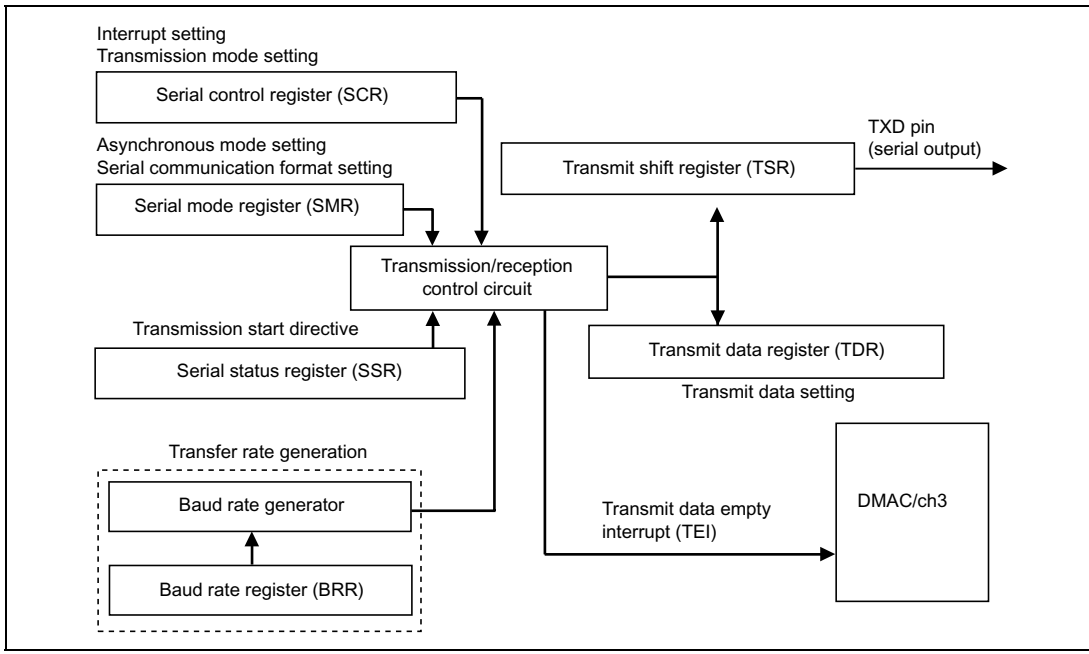


Figure 2.41 SCI Transmission Block Diagram

(b) Figure 2.42 shows a block diagram of SCI reception circuits. In this sample task, data is received from the console using the following SCI functions.

— A function that performs data communication asynchronously, with synchronization performed character by character (asynchronous mode)

— A function that generates an interrupt on completion of reception (RXI interrupt)

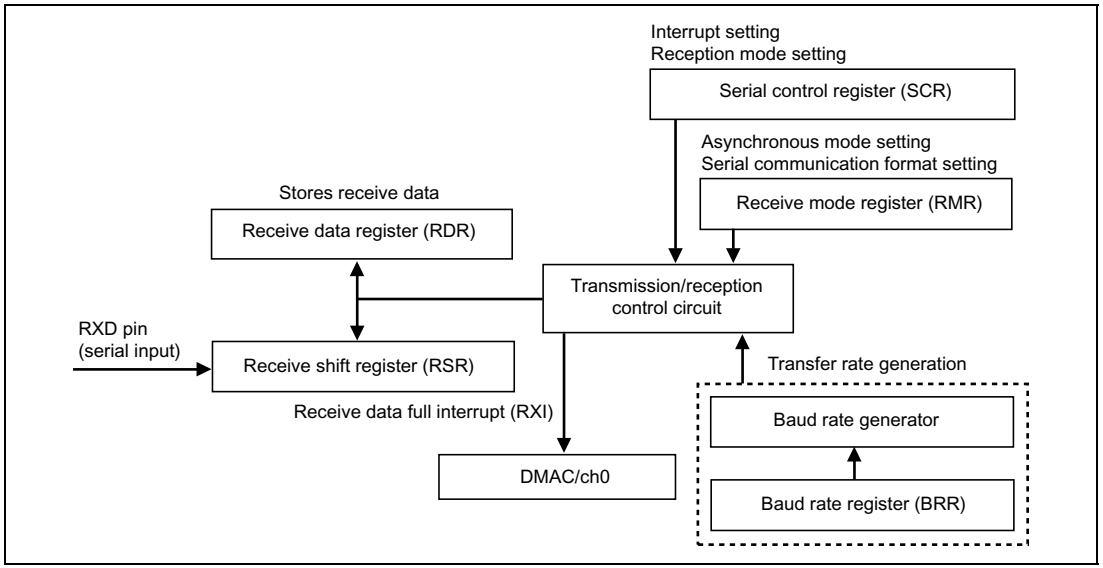


Figure 2.42 SCI Reception Block Diagram

(c) Figure 2.43 shows a block diagram of DMAC/ch0 used in this sample task. In this sample task, block transfer is performed using the following DMAC/ch0 functions.

— A function that performs direct transfer of address data for both the transfer source and transfer destination when the DMAC is activated (direct address transfer mode)

— A function that activates the DMAC by means of the SCI

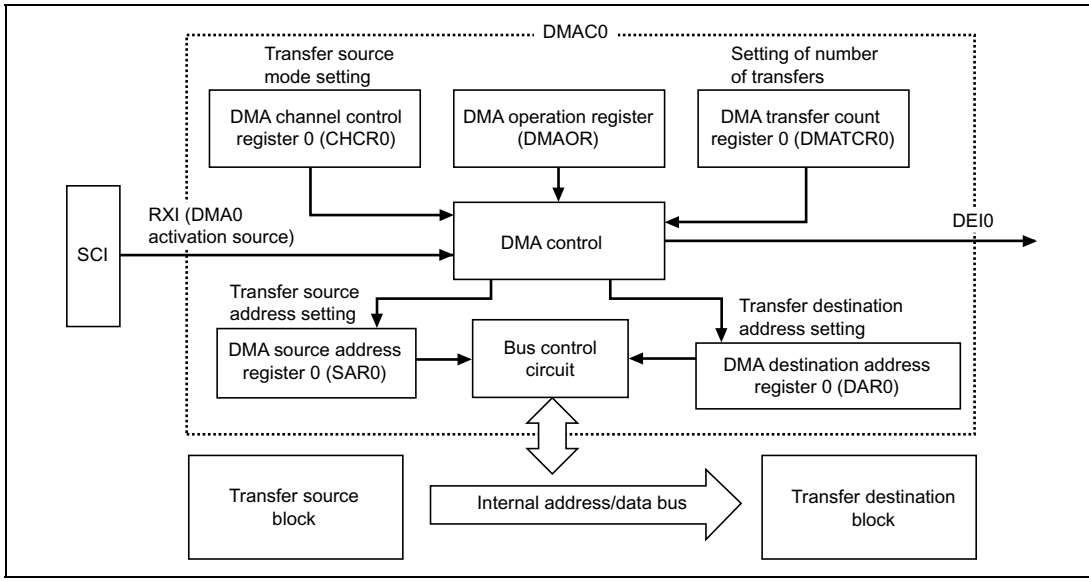


Figure 2.43 Block Diagram of DMAC/ch0

(d) Figure 2.44 shows a block diagram of DMAC/ch3 used in this sample task. In this sample task, block transfer is performed using the following DMAC/ch3 functions.

— A function that transfers data stored in a transfer source register when the DMAC is activated (indirect address transfer mode)

— A function that activates the DMAC by means of the SCI

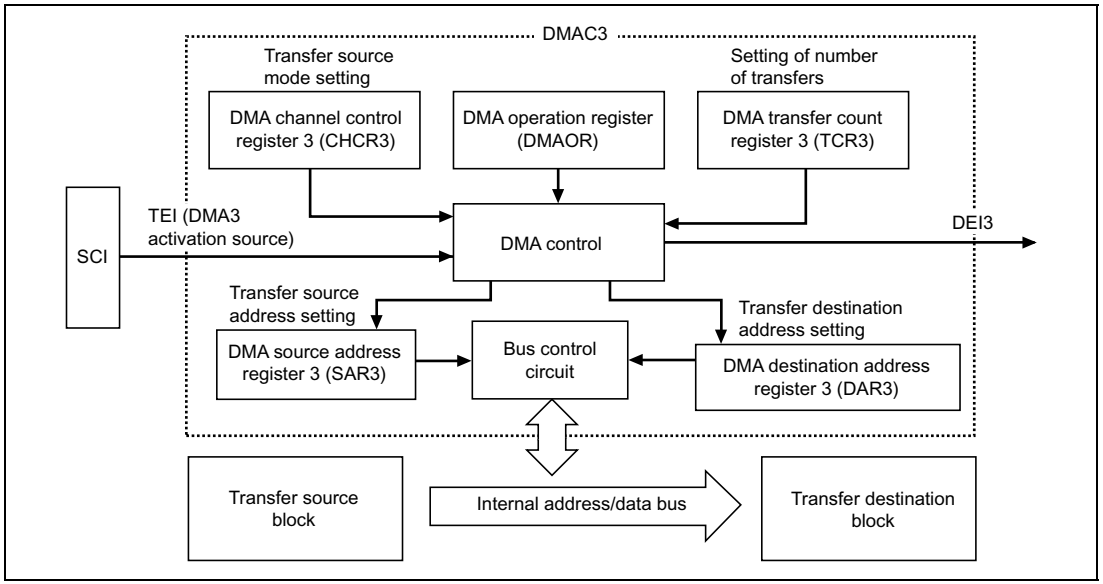


Figure 2.44 Block Diagram of DMAC/ch3

Table 2.13 shows the function assignments used in this sample task. Data transmission/reception is performed via the SCI by assigning DMAC and SCI functions as shown in the table.

Table 2.13 Function Assignments

Pin or Register Name	Function Assignment
SAR0	Transfer source address setting
SAR3	
DAR0	Transfer destination address setting
DAR3	
TCR0	Setting of number of transfers
TCR3	
CHCR0	Setting of DMAC operating mode, transfer method, etc.
CHCR3	
DMAOR	DMAC executing channel priority level setting
RXD	Data reception pin
TXD	Data transmission pin
SMR	SCI transmission format setting
SCR	SCI interrupt enabling/disabling setting
SSR	Interrupt status setting
RDR	Stores receive data from console
TDR	Used to transfer transmit data to console
BRR	Transfer rate setting

Principles of Operation

(1) Figure 2.45 illustrates the principles of operation of this sample task. Data transmission/reception is performed via the SCI by SH7145 hardware and software processing as shown in the figure.

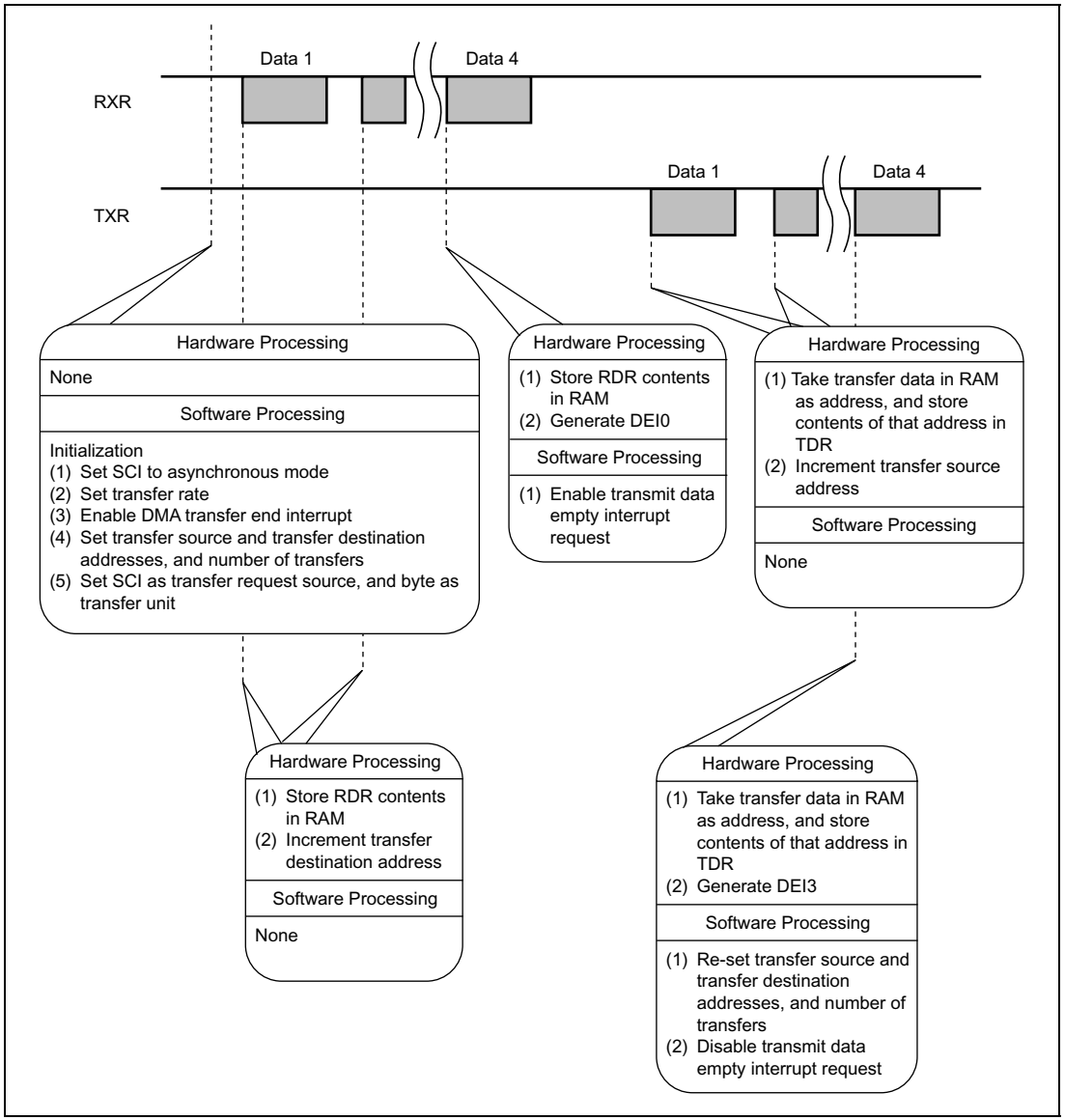


Figure 2.45 Principles of Operation of Data Transfer by SCI

Software

(1) Modules

Module Name	Label	Function Assignment
Main routine	rammon	Performs SCI and DMAC initialization
Receive data transfer	dma_rdr	Initiated by DEI0. Enables transmit data empty interrupt request
Transmit data transfer	dma_tdr	Initiated by DEI3. Re-sets transfer source and transfer destination addresses, and number of transfers. Disables transmit data empty interrupt request

(2) Arguments

Label or Register Name	Function	Data Length	Module	Input/ Output
dat.addr0	Stores RAM reference address	Longword	Main routine	Input
data0	Stores reference data ("H")	Byte	Main routine	Output

(3) Internal Registers Used

Register Name	Function	Address	Set Value
P_STBY.MSTCR1	SCI and DMAC module standby mode clearing	H'FFFF861C	H'f03d
P_DMAC0.SAR0	Used to set RDR address	H'FFFF86C0	&P_SCI1.RDR_1
P_DMAC0.DAR0	Used to set transfer destination RAM start address	H'FFFF86C4	&dat.addr0
P_DMAC0.DMATCR0	Sets 4 as number of transfers	H'FFFF86C8	H'04
P_DMAC0.CHCR0	Setting of DMAC operating mode, transfer method, etc.	H'FFFF86CC	H'00004f05
P_DMAC3.SAR3	Used to set transfer source RAM start address	H'FFFF86F0	&dat.addr0
P_DMAC3.DAR3	Used to set TDR address	H'FFFF86F4	&P_SCI1.TDR_1
P_DMAC3.DMATCR3	Sets 1 as number of transfers	H'FFFF86F8	H'01
P_DMAC3.CHCR3	Setting of DMAC operating mode, transfer method, etc.	H'FFFF86FC	H'00101e04
P_DMAC.DMAOR	Setting of DMAC executing channel priority level	H'FFFF86B0	H'0001
P_PORTA.PAIORL	Sets SCI ch1 input/output pin	H'FFFF8386	H'0010
P_PORTA.PACRL2	Sets pin multiplexing to SCI0	H'FFFF838E	H'0140
P_INTC.IPRC	Sets 14/15 as DMAC ch0/3 interrupt priority levels	H'FFFF834C	H'e00f
P_INTC.IPRF	Sets 10 as SCI/ch1 interrupt priority level	H'FFFF8352	H'000a
P_SCI1.SMR_1	Sets SCI to asynchronous mode	H'FFFF81B0	H'00
P_SCI1.SCR_1	Enables transmission/reception interrupts, transmission/reception operation	H'FFFF81B2	H'50
P_SCI1.BRR_1	Sets transfer rate (19200 bits/s)	H'FFFF81B1	H'40

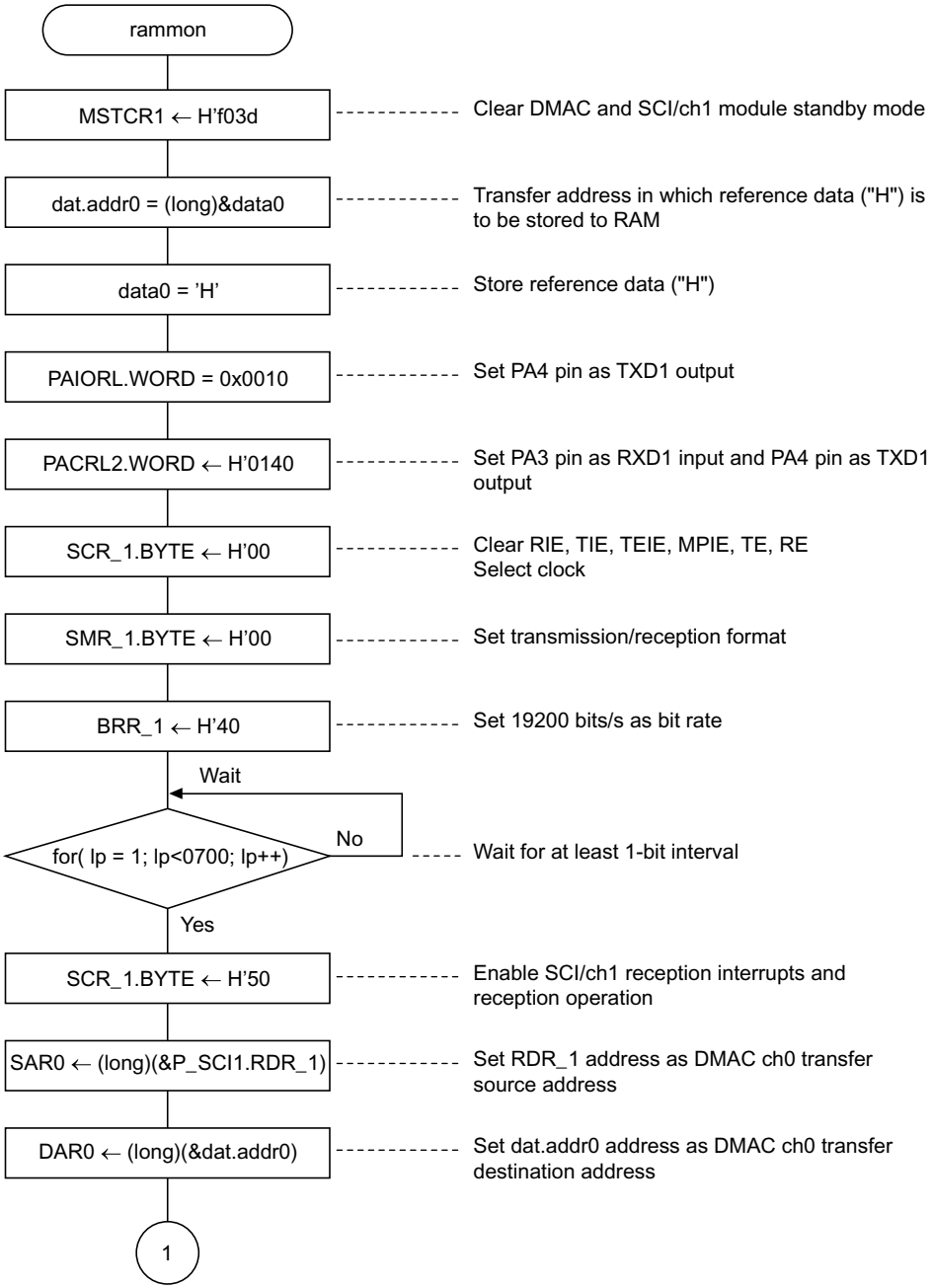
(4) RAM Used

This task does not use any RAM apart from the arguments.

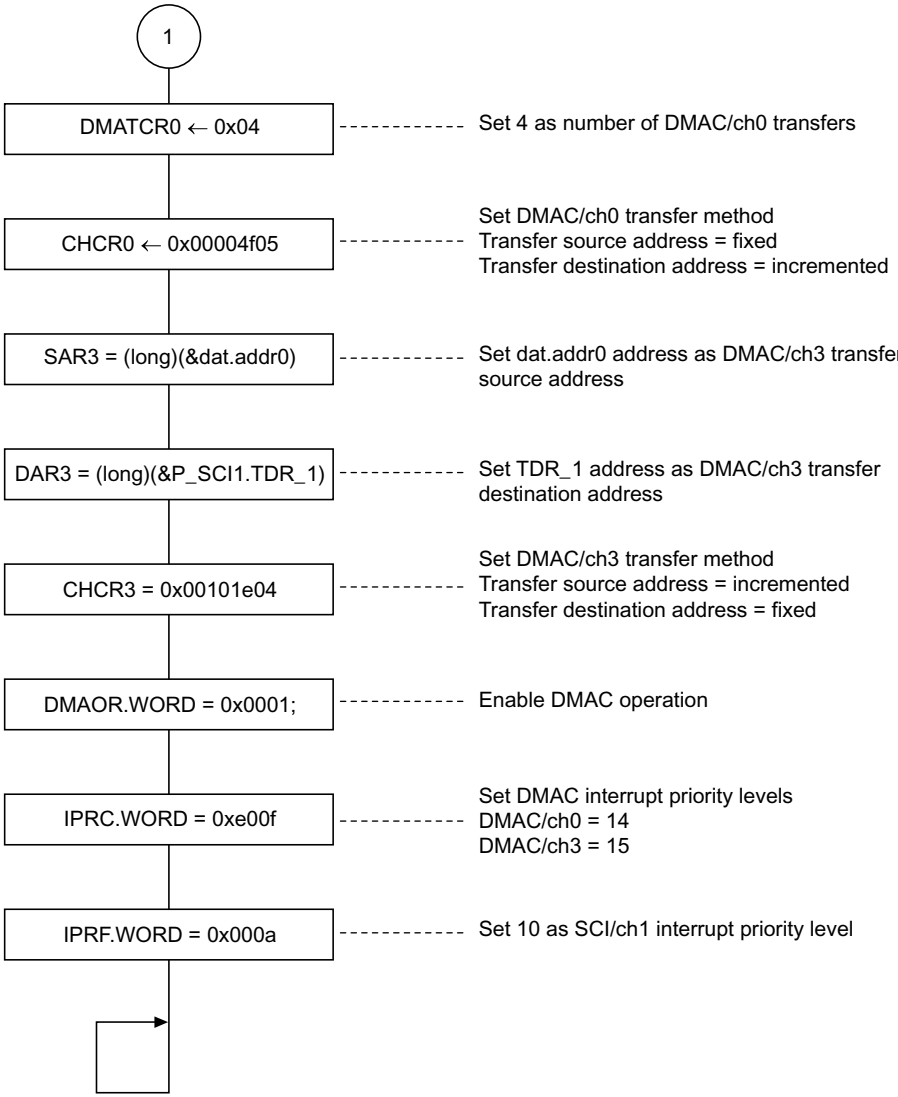
Note: SH7145 header file names are used for register label names.

Flowcharts

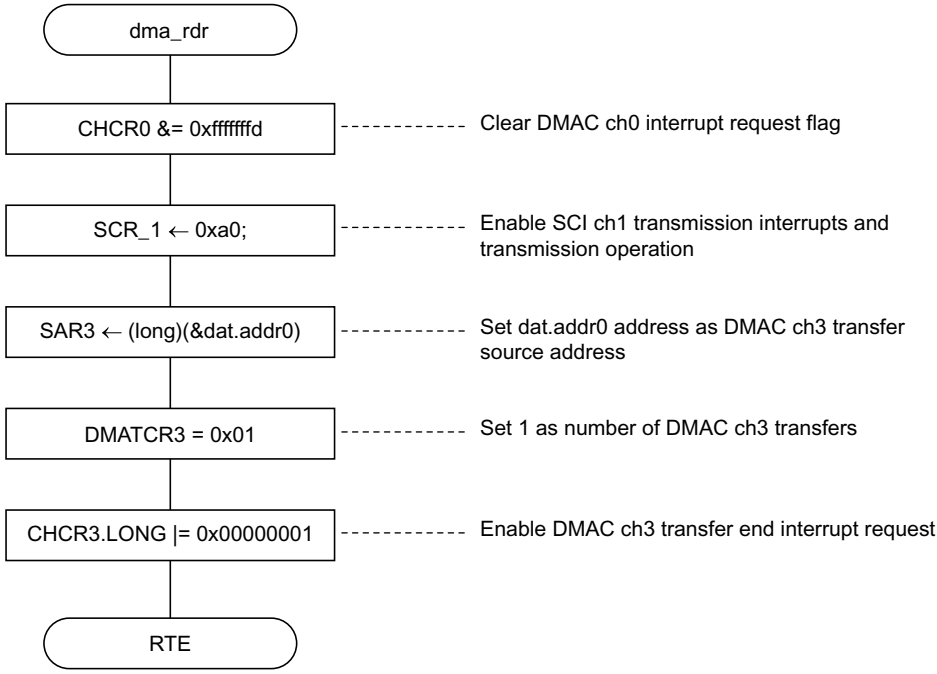
(1) Main routine



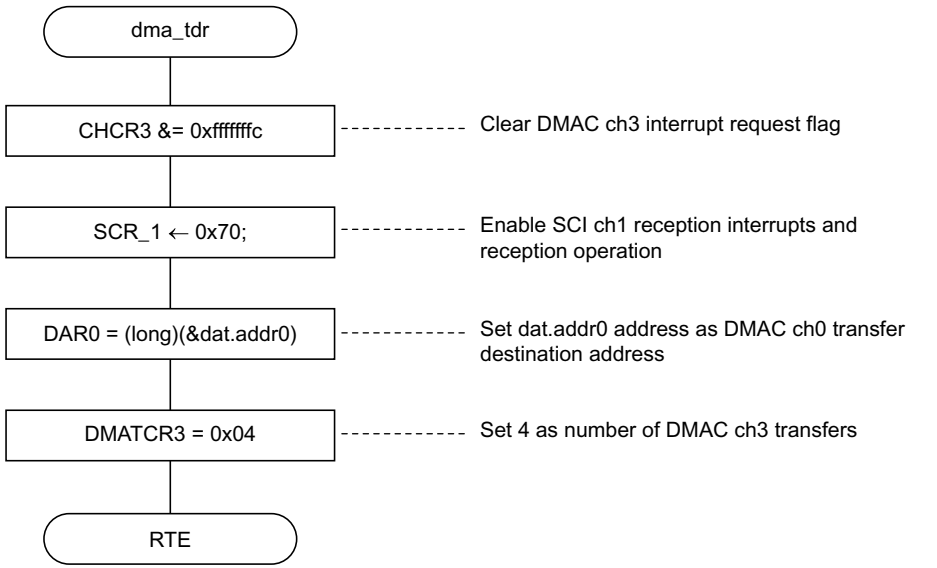
(2) Main routine (cont)



(3) Receive data transfer



(4) Transmit data transfer



Program Listing

```
/*****************************************************************************/
/*                                INCLUDE FILE                               */
/*****************************************************************************/
#include<machine.h>
#include "iodefine_7145F.h"
/*****************************************************************************/
/*                                 PROTOTYPE                                 */
/*****************************************************************************/
void rammon(void);
#pragma interrupt(dma_rdr,dma_tdr)
/*****************************************************************************/
/*                               RAM ALLOCATION                              */
/*****************************************************************************/
#define data0 (*(volatile unsigned char *)0xffffe000)
volatile struct addr
{
     long addr0;
};
#define dat (*(struct addr *)0xffffe010)
/*****************************************************************************/
/*                               MAIN PROGRAM                                */
/*****************************************************************************/
void rammon(void)
 {
  signed int lp;
  P_STBY.MSTCR1.WORD = 0xf03d;    /* DMAC, SCI ch1 module standby mode clear */

  dat.addr0 = (long)&data0;
  data0 = 'H';                            /* reference data = "H" */
  P_PORTA.PAIORL.WORD = 0x0010;
  P_PORTA.PACRL2.WORD = 0X0140;

  P_SCI1.SCR_1.BYTE = 0x00;
  P_SCI1.SMR_1.BYTE = 0x00;               /* Pϕ/1 */
  P_SCI1.BRR_1 = 0x40;                    /* 19200bps */
  for( lp = 1; lp<0300; lp++);            /* for loop = 1bit */
  P_SCI1.SCR_1.BYTE = 0x50;
  P_DMAC0.SAR0 = (long)(&P_SCI1.RDR_1);   /* DMAC ch0 source address */
  P_DMAC0.DAR0 = (long)(&dat.addr0);      /* DMAC ch0 destination address */
  P_DMAC0.DMATCR0 = 0x04;                 /* DMAC ch0 transfer count =4 */
  P_DMAC0.CHCR0.LONG = 0x00004f05;
  P_DMAC3.SAR3 = (long)(&dat.addr0);      /* DMAC ch3 source address */
  P_DMAC3.DAR3 = (long)(&P_SCI1.TDR_1);   /* DMAC ch0 destination address */
  P_DMAC3.CHCR3.LONG = 0x00101e04;
  P_DMAC.DMAOR.WORD = 0x0001;             /* Enable DMAC operation */
  P_INTC.IPRC.WORD = 0xe00f;              /* DMAC ch0 interrupt level =14 ch3
                                             interrupt level =15 */
  P_INTC.IPRF.WORD = 0x000a;              /* SCI ch1 interrupt level =10 */
  set_imask(0x0);
  while(1);                               /* Loop */
 }
```

```
void dma_rdr(void)                          /* DMAC ch1  innterrupt routine */
 {                                          /* DMAC ch1 data taransfer end */
  P_DMAC0.CHCR0.LONG &= 0xfffffffd;         /* Clear flag */
  P_SCI1.SCR_1.BYTE = 0xa0;                 /* Enable SCI ch1 transfer & transfer
                                               data empty interrupt */
  P_DMAC3.SAR3 = (long)(&dat.addr0);        /* DMAC ch3 source address */
  P_DMAC3.DMATCR3 = 0x01;                   /* DMAC ch3 transfer count =1 */
  P_DMAC3.CHCR3.LONG |= 0x00000001;         /* DMAC ch3 enable*/
 }

void dma_tdr(void)                          /* DMAC ch3 interrupt routine*/
 {                                          /* DMAC ch1 data transfer end*/
  P_DMAC3.CHCR3.LONG &= 0xfffffffc;         /* Clear flag */
  P_SCI1.SCR_1.BYTE = 0x70;                 /* Enable SCI ch1 receive data & receive
                                               data full interrupt */
  P_DMAC0.DAR0 = (long)(&dat.addr0);        /* DMAC ch0 destination address */
  P_DMAC0.DMATCR0 = 0x04;                   /* DMAC ch3 transfer count =4 */
 }
```

SH7144 Series Application Note

Publication Date: 1st Edition, March 2003
Published by: Business Operation Division
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.