To our customers,

Old Company Name in Catalogs and Other Documents

On April 1\textsuperscript{st}, 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 1\textsuperscript{st}, 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.
SH7046 Group
On-Chip I/O Volume
Application Note
Renesas 32-Bit RISC
Microcomputer
SuperH™ RISC engine Family/
SH7046 Series

Renesas Electronics
www.renesas.com
Rev.1.00  2003.7
Renesas 32-Bit RISC Microcomputer
SuperH™ RISC engine Family/
SH7046 Series

SH7046 Group
On-Chip I/O Volume

Application Note
### 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.
Preface

The SH7046, SH7048, SH7047, SH7049, and SH7101 are high-performance microcomputers with a 32-bit SH-2 CPU core that uses a RISC (reduced instruction set computer) type instruction set, and comprehensive on-chip peripheral functions.

On-chip peripherals include a CPU, ROM, RAM, a 16-bit multifunction timer pulse unit (MTU), serial communication interface (SCI), port output enable (POE), data transfer controller (DTC) (except for the SH7101), and motor management timer (MMT) (except for the SH7101), enabling these microcomputers to be used for a wide range of applications covering small to large-scale systems.

This Application Note includes sample tasks that use the SH7046 Series' on-chip peripheral functions, which we hope users will find useful as reference material in carrying out software design.

Although the operation of the task programs in this Application Note has been checked, operation should be confirmed again before any of these programs are actually used.
# Contents

**Section 1** Using the SH7046 Series Application Note ........................................... 1  
1.1 Organization of On-Chip I/O Volume ................................................................. 1

**Section 2** SH7046 On-Chip I/O Volume ............................................................... 3  
2.1 Pulse High and Low Width Measurement ......................................................... 3  
2.2 Pulse Output ....................................................................................................... 11  
2.3 PWM 4-Phase Output ......................................................................................... 17  
2.4 PWM 7-Phase Output ......................................................................................... 26  
2.5 Positive-Phase/Negative Phase PWM 3-Phase Output ...................................... 35  
2.6 Complementary PWM 3-Phase Output ............................................................. 43  
2.7 2-Phase Encoder Count .................................................................................... 56  
2.8 Externally Triggered Timer Waveform Cutoff .................................................... 70  
2.9 DC Motor Control Signal Output ....................................................................... 78  
2.10 Start of A/D Conversion by MTU ...................................................................... 86  
2.11 Complementary PWM 3-Phase Output ........................................................... 94  
2.12 Start of A/D Conversion by MMT ................................................................. 109

**Section 3** Appendix ............................................................................................ 117
Section 1 Using the SH7046 Series Application Note

This Application Note illustrates the operation of SH7046 Series’ on-chip peripheral functions based on simple sample tasks.

1.1 Organization of On-Chip I/O Volume

In the On-Chip I/O Volume, the layout shown in figure 1.1 is employed to describe the use of the peripheral functions.

<table>
<thead>
<tr>
<th>On-chip I/O volume</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Functions Used</td>
</tr>
<tr>
<td></td>
<td>Operation</td>
</tr>
<tr>
<td></td>
<td>Software</td>
</tr>
<tr>
<td></td>
<td>Modules</td>
</tr>
<tr>
<td></td>
<td>Arguments</td>
</tr>
<tr>
<td></td>
<td>Internal Registers Used</td>
</tr>
<tr>
<td></td>
<td>RAM Used</td>
</tr>
<tr>
<td></td>
<td>Flowcharts</td>
</tr>
<tr>
<td></td>
<td>Program Listing</td>
</tr>
</tbody>
</table>

Figure 1.1 On-Chip I/O Volume

(1) Specifications
Describes the system specifications for the sample task.

(2) Functions Used
Describes the features of the peripheral function(s) used in the sample task, and peripheral function assignment.

(3) Operation
Describes the operation of the sample task, using a timing chart.

(4) 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 module, and the output arguments
    after execution.
(c) Internal Registers Used
    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 RAM used by the modules.

(5) Flowcharts
    Describes the software that executes the sample task, using general flowcharts.

(6) Program Listing
    Shows a program listing of the software that executes the sample task.
Section 2   SH7046 On-Chip I/O Volume

2.1 Pulse High and Low Width Measurement

<table>
<thead>
<tr>
<th>Pulse High and Low Width Measurement</th>
<th>MCU: SH7046/47</th>
<th>Functions Used: MTU (Input Capture)</th>
</tr>
</thead>
</table>

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 $P\phi = 20.0 \text{ MHz}$, the pulse high width and low width can be measured in a range of 50.0 ns to 3.27 ms in 50.0 ns units.

![Figure 2.1 Pulse Width Measurement Timing](image-url)
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](image)

(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>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCR_0</td>
<td>Register</td>
<td>Counter clearing source selection</td>
</tr>
<tr>
<td>TIORH_0</td>
<td>Register</td>
<td>Selects input edge of input capture signal</td>
</tr>
<tr>
<td>TGRA_0</td>
<td>Register</td>
<td>Stores counter value at pulse rising edge or falling edge</td>
</tr>
<tr>
<td>TGIA_0</td>
<td>Register bit</td>
<td>Initiates pulse high and low width measurement at pulse rising edge or failing edge</td>
</tr>
<tr>
<td>TIOC0A</td>
<td>Pin</td>
<td>Inputs pulse to be measured</td>
</tr>
</tbody>
</table>
Operation

(1) Figure 2.3 illustrates the principles of operation of this sample task. Pulse high width and low width measurement is performed by SH7046 hardware and software processing as shown in the figure.

![Figure 2.3 Principles of Operation of Pulse Width Measurement](image)
### Software

#### (1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>pwhlmn</td>
<td>MTU initialization</td>
</tr>
<tr>
<td>Pulse high width and low width测量</td>
<td>pwhl1</td>
<td>Initiated by TGIA_0. Measures pulse high width and low width based on TGRA_0 value, and stores results in RAM</td>
</tr>
</tbody>
</table>

#### (2) Arguments

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function Assignment</th>
<th>Data Length</th>
<th>Module</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwh_hdata</td>
<td>Used to set timer value for pulse high width</td>
<td>1 word</td>
<td>Pulse high width and low width measurement</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>Pulse high width is calculated using following equation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pulse high width (ns) = timer value × φ period</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(50.0 ns at 20.0 MHz operation)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pwh_ldata</td>
<td>Used to set timer value for pulse low width</td>
<td>1 word</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pulse low width is calculated using following equation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pulse low width (ns) = timer value × φ period</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(50.0 ns at 20.0 MHz operation)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PORTE.PECRL2</td>
<td>Enables pulse input from TIOC0A input pin</td>
<td>H'FFFF83BA</td>
<td>H'0001</td>
</tr>
<tr>
<td>P_MTU0.TCR_0</td>
<td>TCNT counter clock selection, and setting of input capture A as counter clearing source</td>
<td>H'FFFF8260</td>
<td>H'20</td>
</tr>
<tr>
<td>P_MTU0.TIORH_0</td>
<td>Sets transfer from TCNT_0 to TGRA_0 on detection of pulse rise or fall</td>
<td>H'FFFF8262</td>
<td>H'08</td>
</tr>
<tr>
<td>P_MTU0.TIER_0</td>
<td>Enables interrupt by TGIA_0</td>
<td>H'FFFF8264</td>
<td>H'41</td>
</tr>
<tr>
<td>P_MTU0.TGRA_0</td>
<td>TCNT_0 values at time of pulse rising edge and falling edge are stored, and pulse period is calculated from these values</td>
<td>H'FFFF8268</td>
<td>pwh_ldata pwh_hdata</td>
</tr>
<tr>
<td>P_INTC.IPRD</td>
<td>Sets 15 as TGIA_0 interrupt priority level</td>
<td>H'FFFF834E</td>
<td>H'1000</td>
</tr>
<tr>
<td>P_STBY.MSTCR2</td>
<td>MTU module standby mode clearing</td>
<td>H'FFFF861E</td>
<td>H'd2fd</td>
</tr>
</tbody>
</table>

(4) RAM Used

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

Note: SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

- **MSTCR2 ← H'd2fd**
  - Clear MTU module standby mode

- **TCR_0 ← H'20**
  - Set input capture A as ch0 counter clearing source in TCR_0

- **TIORH_0 ← H'08**
  - Set rising edge as pulse detection edge in TIORH_0

- **TIER_0 ← H'41**
  - Enable interrupt by TGIA_0 in TIER_0

- **IPRD ← H'f000**
  - Set 15 as TGIA_0 interrupt priority level in INCT:IPRD

- **PECRL2 ← H'0001**
  - Enable pulse input from TIOC0A input pin in PECRL2

- **CST ← 1**
  - Set CST to 1 to start ch0 count operation
(2) Pulse high and low width measurement

- pwh1
  - TGFA ← 0
  - TIORH_0 & & H'0f = H'08? (Rising edge detection?)
    - No
    - Yes
      - pwh_hdata ← TGRA_0
    - TIORH_0 |= H'01
      - pwh_ldata ← TGRA_0
      - TIORH_0 |= H'fe

- Clear interrupt request flag
- Set pulse width in high width area
- Set falling edge as pulse detection edge in TIORH_0
- Set pulse width in low width area
- Set rising edge as pulse detection edge in TIORH_0

RTE
Program Listing

/******************************************************************************/
/*                                  INCLUDE FILE                              */
/******************************************************************************/
#include <machine.h>
#include "iodefine_7046.h"
/******************************************************************************/
/*                                  PROTOTYPE                                 */
/******************************************************************************/
void  pwhlmn(void);
#pragma interrupt(pwhl1)
/******************************************************************************/
/*                                 RAM ALLOCATION                             */
/******************************************************************************/
#define pwh_hdata  (*(unsigned short  *)0xffffd000)
#define pwh_ldata  (*(unsigned short  *)0xffffd002)
/******************************************************************************/
/*                                  MAIN PROGRAM                              */
/******************************************************************************/
void pwhlmn(void)
{
  set_imask(0xf);
  P_STBY.MSTCR2.WORD = 0xd2fd; /* timer clear input capture with TGRA_0 */
  P_MTU0.TCR_0.BYTE = 0x20;  /* counter clock = φ/1 */
  P_MTU0.TIORH_0.BYTE = 0x08; /* input capture by TIOCOA 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;
    P_MTU0.TIORH_0 |= 0x01;
  }
  else
  {
    pwh_ldata = P_MTU0.TGRA_0.BYTE;
    P_MTU0.TIORH_0 |= 0xfe;
  }
}

Rev. 1.00, 07/03, page 10 of 160
2.2 Pulse Output

| Pulse Output | MCU: SH7046/47 | 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 $P\phi = 20.0$ MHz, the output pulse width can be set arbitrarily in the range 100.0 ns to 3.27 ms.

![Figure 2.4 Pulse Output](image-url)
Functions Used

(1) In this sample task, a pulse with a 50% duty 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

<table>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIOC0A</td>
<td>Pin</td>
<td>Pulse output pin</td>
</tr>
<tr>
<td>TCR_0</td>
<td>Register</td>
<td>Selection of counter clearing source and input clock</td>
</tr>
<tr>
<td>TIORH_0</td>
<td>Register</td>
<td>Pulse output level setting</td>
</tr>
<tr>
<td>TGRA_0</td>
<td>Register</td>
<td>Pulse 1/2 period setting</td>
</tr>
</tbody>
</table>
Operation

(1) Figure 2.6 illustrates the principles of operation of this sample task. Pulses are output by SH7046 hardware and software processing as shown in the figure.

Figure 2.6  Principles of Operation of Pulse Output
Software

(1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>puls_out</td>
<td>PFC and pulse output setting</td>
</tr>
</tbody>
</table>

(2) Arguments

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function Assignment</th>
<th>Data Length</th>
<th>Module</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>pul_cyc</td>
<td>Used to set timer value for pulse 1/2 period</td>
<td>1 word</td>
<td>Main routine</td>
<td>Input</td>
</tr>
<tr>
<td></td>
<td>Pulse period is calculated using following equation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pulse period (ns) = timer value × φ period</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(50.0 ns at 20.0 MHz)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PORTE.PECRL</td>
<td>Sets TIOC0A pin as output</td>
<td>H'FFFF83B4</td>
<td>H'0001</td>
</tr>
<tr>
<td>P_PORTE.PECRL2</td>
<td>Used to set multiplex pin as TIOC0A output</td>
<td>H'FFFF83BA</td>
<td>H'0001</td>
</tr>
<tr>
<td>P_MTU0.TCR_0</td>
<td>Sets TGRA_0 compare match as counter clearing source</td>
<td>H'FFFF8260</td>
<td>H'20</td>
</tr>
<tr>
<td></td>
<td>Sets Pφ/1 as input clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_MTU0.TIORH_0</td>
<td>TIOC0A initial output 0, output toggled on compare match</td>
<td>H'FFFF8262</td>
<td>H'03</td>
</tr>
<tr>
<td>P_MTU0.TGRA_0</td>
<td>Output pulse 1/2 period setting</td>
<td>H'FFFF8268</td>
<td>pul_cyc</td>
</tr>
<tr>
<td>P_MTU0.TMDR_0</td>
<td>Sets MTU/ch0 to normal mode</td>
<td>H'FFFF8261</td>
<td>H'c0</td>
</tr>
<tr>
<td>P_STBY.MSTCR2</td>
<td>MTU module standby mode clearing</td>
<td>H'FFFF861E</td>
<td>H'd2fd</td>
</tr>
</tbody>
</table>

(4) RAM Used

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

Note: SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

- puls_out
- MSTCR2 ← H’2fd
  - Clear MTU module standby mode
- PEIORL ← H’0001
  - PECRL2 ← H’0001
  - Set TIOC0A as timer output
- TCR_0 ← H’20
  - Set counter clearing source in P_MTU0.TCR_0
- TIORH_0 ← H’03
  - Set output toggling by compare match in P_MTU0.TIORH_0
- TGRA_0 ← pul_cyc
  - Set 1/2 pulse period in P_MTU0.TGRA_0
- TMDR_0 ← H’c0
  - Set mode to normal operation in P_MTU0.TMDR_0
- CST ← 1
  - Set CST to 1 to start ch0 counter operation
Program Listing

/**************************************************************************/
/*                                  INCLUDE FILE                             */
/**************************************************************************/
#include<machine.h>
#include"iodefine_7046.h"
/**************************************************************************/
/*                                  PROTOTYPE                                */
/**************************************************************************/
void puls_out(void);
/**************************************************************************/
/*                                 RAM ALLOCATION                            */
/**************************************************************************/
#define  pul_cyc  (*(unsigned short *)0xffffd000)
/**************************************************************************/
/*                                 MAIN PROGRAM                              */
/**************************************************************************/
void puls_out(void)
{
    P_STBY.MSTCR2.WORD = 0xd2fd;
    P_PORTE.PEIORL.WORD = 0x0001; /* TIOC0A = Output */
    P_PORTE.PECRL2.WORD = 0x0001; /* PE0 function = TIOC0A */
    P_MTU0.TCR_0.BYTE = 0x20; /* Counter cleare by TGRA */
    P_MTU0.TIORH_0.BYTE = 0x03; /* toggle output */
    P_MTU0.TGRA_0 = pul_cyc; /* 1/2 period */
    P_MTU0.TCNT_0 = 0x0000; /* Cleara timer counter */
    P_MTU0.TMDR_0.BYTE = 0xc0; /* Set mode */
    P_MTU34.TSTR.CST.BIT = 1; /* Start timer counter */
    while(1);
}
2.3 PWM 4-Phase Output

**Specifications**

(1) Using MTU PWM mode 1, 4-phase PWM output is performed based on a set duty value 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 of 0% to 100% can be set with a 1/65,535 resolution.

![Figure 2.7 Example of PWM Output](image_url)
**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.

![Block Diagram of MTU/ch1, ch2, ch3](image-url)

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>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIOC1A</td>
<td>Pins</td>
<td>PWM pulse output pins</td>
</tr>
<tr>
<td>TIOC2A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC3A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC3C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCR_1</td>
<td>Registers</td>
<td>Selection of ch1 to ch3 timer counter clearing sources and input clocks</td>
</tr>
<tr>
<td>TCR_2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCR_3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMDR_1</td>
<td>Registers</td>
<td>Operation of ch1 to ch3 in PWM mode 1</td>
</tr>
<tr>
<td>TMDR_2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMDR_3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRA_1</td>
<td>Registers</td>
<td>PWM period setting</td>
</tr>
<tr>
<td>TGRA_2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRA_3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRB_1</td>
<td>Registers</td>
<td>Duty value setting</td>
</tr>
<tr>
<td>TGRB_2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRB_3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRC_3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRD_3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 SH7046 hardware and software processing as shown in the figure.

---

**Figure 2.9 Principles of Operation of PWM Waveforms**

Rev. 1.00, 07/03, page 20 of 160
# Software

## (1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>pwm_1</td>
<td>PFC and PWM output setting</td>
</tr>
</tbody>
</table>

## (2) Arguments

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>pul_cyc1</td>
<td>Used to set timer value for pulse period</td>
</tr>
<tr>
<td>pul_cyc2</td>
<td>Pulse period is calculated using following equation:</td>
</tr>
<tr>
<td>pul_cyc3</td>
<td>Pulse period (ns) = timer value × φ period</td>
</tr>
<tr>
<td></td>
<td>(50.0 ns at 20.0 MHz operation)</td>
</tr>
<tr>
<td>pul_duty1b</td>
<td>Used to set TIOC pin output waveform transition timing</td>
</tr>
<tr>
<td>pul_duty2b</td>
<td></td>
</tr>
<tr>
<td>pul_duty3b</td>
<td></td>
</tr>
<tr>
<td>pul_duty3c</td>
<td></td>
</tr>
<tr>
<td>pul_duty3d</td>
<td></td>
</tr>
</tbody>
</table>

## (3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>MTU module standby mode clearing</td>
</tr>
<tr>
<td>P_PORTE.PEIORL</td>
<td>Multiplex pins set as timer output pins</td>
</tr>
<tr>
<td>P_PORTE.PECRL1</td>
<td>TIOC1A, TIOC2A, TIOC3A, TIOC3C</td>
</tr>
<tr>
<td>P_PORTE.PECRL2</td>
<td></td>
</tr>
<tr>
<td>P_MTU1.TCR_1</td>
<td>Timer counter clearing sources cleared by TGRA_1, TGRA_2, TGRA_3</td>
</tr>
<tr>
<td>P_MTU1.TGRA_1</td>
<td>Channel 1 PWM period setting</td>
</tr>
<tr>
<td>P_MTU1.TGRB_1</td>
<td>Used to set timer counter value causing high output from TIOC1A</td>
</tr>
<tr>
<td>P_MTU2.TCR_2</td>
<td></td>
</tr>
<tr>
<td>P_MTU2.TGRA_2</td>
<td>Channel 2 PWM period setting</td>
</tr>
<tr>
<td>P_MTU2.TGRB_2</td>
<td>Used to set timer counter value causing high output from TIOC2A</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Register Name</td>
<td>Function Assignment</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_MTU34.TGRA_3</td>
<td>Channel 3 PWM period setting</td>
</tr>
<tr>
<td>P_MTU34.TGRB_3</td>
<td>Used to set timer counter value causing high output from TIOC3A</td>
</tr>
<tr>
<td>P_MTU34.TGRC_3</td>
<td>Used to set timer counter value causing low output from TIOC3C</td>
</tr>
<tr>
<td>P_MTU34.TGRD_3</td>
<td>Used to set timer counter value causing high output from TIOC3C</td>
</tr>
<tr>
<td>P_MTU1.TIOR_1</td>
<td>Sets TGRA_1 initial output 0, 0 output on output compare, TGRB_1 initial output 0, 1 output on output compare</td>
</tr>
<tr>
<td>P_MTU2.TIOR_2</td>
<td>Sets TGRA_2 initial output 0, 0 output on output compare, TGRB_2 initial output 0, 1 output on output compare</td>
</tr>
<tr>
<td>P_MTU34.TIORH_3</td>
<td>Sets TGRA_3 initial output 0, 0 output on output compare, TGRB_3 initial output 0, 1 output on output compare</td>
</tr>
<tr>
<td>P_MTU34.TIORL_3</td>
<td>Sets TGRC_3 initial output 0, 0 output on output compare, TGRD_3 initial output 0, 1 output on output compare</td>
</tr>
<tr>
<td>P_MTU1.TMDR_1</td>
<td>Used to set PWM mode 1 as operating mode</td>
</tr>
<tr>
<td>P_MTU2.TMDR_2</td>
<td></td>
</tr>
<tr>
<td>P_MTU34.TMDR_3</td>
<td></td>
</tr>
</tbody>
</table>

(4) RAM Used

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

**Note:** SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

```
pwm_1
  MSTCR2 ← H'd2fd  ------- Clear MTU module standby mode
  PEIORL ← H'0550
  PECLRL1 ← H'0011
  PECLRL2 ← H'1100
  TCR_1 ← H'20
  TCR_2 ← H'20
  TCR_3 ← H'20
  TGRA_1 ← pul_cyc1
  TGRA_2 ← pul_cyc2
  TGRA_3 ← pul_cyc3
  TGRB_1 ← pul_duty1b
  TGRB_2 ← pul_duty2b
  TGRB_3 ← pul_duty3b
  TGRB_3 ← pul_duty3c
  TGRD_3 ← pul_duty3d
  TIOR_1 ← H'02
  TIOR_2 ← H'02
  TIORL_3 ← H'21
  TIORH_3 ← H'02
  TMDR_1 ← H'c2
  TMDR_2 ← H'c2
  TMDR_3 ← H'c2
  TSTR ← H'46
  ------- Set PWM mode 1 in TMDR_1/2/3
  ------- Enable ch1/2/3 count operation with P_MTU34.TSTR
```

Perform TIOC1A, 2A, 3A, 3C output settings in P_PORTE.PECRL1/2
Set TGRA_1/2/3 as TCR counter clearing sources
Set pulse periods in P_MTU1.TGRA_1, P_MTU2.TGRA_2, P_MTU3.TGRA_3
Set duty values in P_MTU1.TGRB_1, P_MTU2.TGRB_2, P_MTU34.TGRB_3, P_MTU34.TGRD_3
Set waveform output values in TIOR
Program Listing

/******************************************************************************/
/*                                  INCLUDE FILE                              */
/******************************************************************************/
#include<machine.h>
#include"iodefine_7046.h"
/******************************************************************************/
/*                                    PROTOTYPE                               */
/******************************************************************************/
void pwm_1(void);
/******************************************************************************/
/*                                  RAM ALLOCATION                            */
/******************************************************************************/
#define pul_cyc1     (*(unsigned short  *)0xffffd000)
#define pul_duty1b   (*(unsigned short  *)0xffffd002)
#define pul_cyc2     (*(unsigned short  *)0xffffd004)
#define pul_duty2b   (*(unsigned short  *)0xffffd006)
#define pul_cyc3     (*(unsigned short  *)0xffffd008)
#define pul_duty3b   (*(unsigned short  *)0xffffd00a)
#define pul_duty3c   (*(unsigned short  *)0xffffd00c)
#define pul_duty3d   (*(unsigned short  *)0xffffd00e)
/******************************************************************************/
/*                                  MAIN PROGRAM                              */
/******************************************************************************/
void pwm_1(void)
{
  P_STBY.MSTCR2.WORD = 0xd2fd;    /* 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 = 0x20;      /* Counter clear by TGRA */
  P_MTU1.TGRA_1 = pul_cyc1;      /* set period */
  P_MTU1.TGRB_1 = pul_duty1b;    /* set duty */
  P_MTU1.TIOR_1.BYTE = 0x02;
  P_MTU1.TMDR_1.BYTE = 0xc2;     /* PWM mode */
  P_MTU1.TCNT_1 = 0x0000;
  P_MTU2.TCR_2.BYTE = 0x20;
  P_MTU2.TGRA_2 = pul_cyc2;
  P_MTU2.TGRB_2 = pul_duty2b;
  P_MTU2.TIOR_2.BYTE = 0x02;
  P_MTU2.TMDR_2.BYTE = 0xc2;
  P_MTU2.TCNT_2 = 0x0000;
  P_MTU34.TCR_3.BYTE = 0x20;
  P_MTU34.TGRA_3 = pul_cyc3;
  P_MTU34.TGRB_3 = pul_duty3b;
  P_MTU34.TGRC_3 = pul_duty3c;
  P_MTU34.TGRD_3 = pul_duty3d;

Rev. 1.00, 07/03, page 24 of 160
P_MTU34.TIORL_3.BYTE = 0x21;
P_MTU34.TIORH_3.BYTE = 0x02;
P_MTU34.TMDR_3.BYTE = 0xc2;
P_MTU34.TCNT_3 = 0x0000;

P_MTU34.TSTR.BYTE = 0x46;    /* Timer counter start */
while(1);
}
2.4 PWM 7-Phase Output

Specifications

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

(2) When operating with on-chip peripheral clock \( P\phi = 20.0 \text{ MHz} \), the output PWM period can be set arbitrarily in the range 100 ns to 3.27 ms.

![Figure 2.10 Example of PWM Output](image-url)
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)

![Block Diagram of Synchronous Clearing](image-url)

Figure 2.11  Block Diagram of Synchronous Clearing
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>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIOC0B</td>
<td>Pins</td>
<td>PWM pulse output pins</td>
</tr>
<tr>
<td>TIOC0C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC0D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC1A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC1B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC2A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC2B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSYR</td>
<td>Register</td>
<td>Ch0/1/2 synchronous operation</td>
</tr>
<tr>
<td>TCR_0/1/2</td>
<td>Register</td>
<td>Selection of ch0/1/2 timer counter clearing sources and input clocks</td>
</tr>
<tr>
<td>TGRA_0</td>
<td>Register</td>
<td>PWM period setting</td>
</tr>
<tr>
<td>TGRB_0</td>
<td>Registers</td>
<td>Duty value setting</td>
</tr>
<tr>
<td>TGRB_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRC_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRD_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRA_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRB_2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRB_2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMDR_0/1/2</td>
<td>Register</td>
<td>Operation of ch0/1/2 in PWM Mode 2</td>
</tr>
</tbody>
</table>
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 SH7046 hardware and software processing as shown in the figure.
**Software**

(1) **Modules**

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>pwm_2</td>
<td>PFC and PWM output setting</td>
</tr>
</tbody>
</table>

(2) **Arguments**

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function Assignment</th>
<th>Data Length</th>
<th>Module</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>pul_cyc0a</td>
<td>Used to set timer value for pulse period</td>
<td>1 word</td>
<td>Main routine</td>
<td>Input</td>
</tr>
<tr>
<td></td>
<td>Pulse period is calculated using following equation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pulse period (ns) = timer value × period (50.0 ns at 20.0 MHz operation)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty0b</td>
<td>Used to set TIOC pin output waveform transition timing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty0c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty0d</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty1a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty1b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty2a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty2b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(3) **Internal Registers Used**

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function Assignment</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>Module standby mode clearing</td>
<td>H'FFFF861E</td>
<td>H'd2fd</td>
</tr>
<tr>
<td>P_PORTE.PEIORL</td>
<td>Used to set multiplex pins as timer output pins TIOC0B/C/D, TIOC1A/B, TIOC2A/B</td>
<td>H'FFFF83B4</td>
<td>H'00fe</td>
</tr>
<tr>
<td>P_PORTE.PECRL2</td>
<td></td>
<td>H'FFFF83BA</td>
<td>H'5554</td>
</tr>
<tr>
<td>P_MTU34.TSYR</td>
<td>Synchronous operation set for timer counters 0/1/2</td>
<td>H'FFFF8241</td>
<td>H'07</td>
</tr>
<tr>
<td>P_MTU0.TCR_0</td>
<td>Used to select TGRA_0 compare match set as timer counter clearing source, and P/1 as input clock</td>
<td>H'FFFF8260</td>
<td>H'20</td>
</tr>
<tr>
<td>P_MTU1.TCR_1</td>
<td></td>
<td>H'FFFF8280</td>
<td>H'60</td>
</tr>
<tr>
<td>P_MTU2.TCR_2</td>
<td></td>
<td>H'FFFF82A0</td>
<td>H'60</td>
</tr>
<tr>
<td>P_MTU0.TGRA_0</td>
<td>PWM period setting</td>
<td>H'FFFF8268</td>
<td>pul_cyc0a</td>
</tr>
<tr>
<td>P_MTU0.TGRB_0</td>
<td>Used to set timer counter value causing high output from TIOC0B</td>
<td>H'FFFF826A</td>
<td>pul_duty0b</td>
</tr>
<tr>
<td>P_MTU0.TGRC_0</td>
<td>Used to set timer counter value causing high output from TIOC0C</td>
<td>H'FFFF826C</td>
<td>pul_duty0c</td>
</tr>
</tbody>
</table>

Rev. 1.00, 07/03, page 30 of 160
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function Assignment</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_MTU0.TGRD_0</td>
<td>Used to set timer counter value causing high output from TIOC0D</td>
<td>H'FFFF826E</td>
<td>pul_duty0d</td>
</tr>
<tr>
<td>P_MTU1.TGRA_1</td>
<td>Used to set timer counter value causing high output from TIOC1A</td>
<td>H'FFFF8288</td>
<td>pul_duty1a</td>
</tr>
<tr>
<td>P_MTU1.TGRB_1</td>
<td>Used to set timer counter value causing high output from TIOC1B</td>
<td>H'FFFF828A</td>
<td>pul_duty1b</td>
</tr>
<tr>
<td>P_MTU2.TGRA_2</td>
<td>Used to set timer counter value causing high output from TIOC2A</td>
<td>H'FFFF82A8</td>
<td>pul_duty2a</td>
</tr>
<tr>
<td>P_MTU2.TGRB_2</td>
<td>Used to set timer counter value causing high output from TIOC2B</td>
<td>H'FFFF82AA</td>
<td>pul_duty2b</td>
</tr>
<tr>
<td>P_MTU0.TIORH_0</td>
<td>Sets TGRA_0 initial output 0, 0 output on output compare, TGRB_0 initial output 0, 1 output on output compare</td>
<td>H'FFFF8262</td>
<td>H'20</td>
</tr>
<tr>
<td>P_MTU0.TIORL_0</td>
<td>Sets TGRC_0 initial output 0, 1 output on output compare, TGRD_0 initial output 0, 1 output on output compare</td>
<td>H'FFFF8263</td>
<td>H'22</td>
</tr>
<tr>
<td>P_MTU1.TIOR_1</td>
<td>Sets TGRA_1 initial output 0, 1 output on output compare, TGRB_1 initial output 0, 1 output on output compare</td>
<td>H'FFFF8282</td>
<td>H'22</td>
</tr>
<tr>
<td>P_MTU1.TIOR_2</td>
<td>Sets TGRA_2 initial output 0, 1 output on output compare, TGRB_2 initial output 0, 1 output on output compare</td>
<td>H'FFFF82A2</td>
<td>H'22</td>
</tr>
<tr>
<td>P_MTU0.TMDR_0</td>
<td>Used to set PWM Mode 2 as operating mode of each channel</td>
<td>H'FFFF8261</td>
<td>H'c3</td>
</tr>
<tr>
<td>P_MTU1.TMDR_1</td>
<td></td>
<td>H'FFFF8281</td>
<td>H'c3</td>
</tr>
<tr>
<td>P_MTU2.TMDR_2</td>
<td></td>
<td>H'FFFF82A1</td>
<td>H'c3</td>
</tr>
</tbody>
</table>

(4) RAM Used

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

Note: SH7046 header file names are used for register label names.
Clear MTU module standby mode

Set TIOC0B/C/D, TIOC1A/B, TIOC2A/B as outputs in P_PORTE.PECLR2

Set TGRA_0 as counter clearing source in P_MTU0.TCR_0, set synchronous clearing in P_MTU1.TCR_1, P_MTU2.TCR_2

Set pulse period in P_MTU0.TGRA_0

Set duty values in P_MTU0.TGRB/C/D_0, P_MTU1.TGRA/B_1, P_MTU2.TGRA/B_2

Set PWM Mode 2 in TMDR_0/1/2

Set synchronous operation for ch0/1/2

Enable ch0/1/2 count operation with P_MTU34.TSTR
# Program Listing

```c
/****************************************************************************/
/*                                INCLUDE  FILE                             */
/****************************************************************************/
#include<machine.h>
#include"iodefine_7046.h"
/****************************************************************************/
/*                                 PROTOTYPE                                */
/****************************************************************************/
void pwm_2(void);
/****************************************************************************/
/*                               RAM ALLOCATION                             */
/****************************************************************************/
#define pul_cyc0  (*(unsigned short *)0xffffd000)
#define pul_duty0b  (*(unsigned short *)0xffffd002)
#define pul_duty0c  (*(unsigned short *)0xffffd004)
#define pul_duty0d  (*(unsigned short *)0xffffd006)
#define pul_dutyla  (*(unsigned short *)0xffffd008)
#define pul_duty1b  (*(unsigned short *)0xffffd00a)
#define pul_duty2a  (*(unsigned short *)0xffffd00c)
#define pul_duty2b  (*(unsigned short *)0xffffd00e)
/****************************************************************************/
/*                                MAIN PROGRAM                               */
/****************************************************************************/
void pwm_2(void)
{
    P_STBY.MSTCR2.WORD =0xd2fd;      /* 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 = 0x20;     /* Counter clear by TGRA_0 */
    P_MTU0.TIORH_0.BYTE = 0x20;
    P_MTU0.TIORL_0.BYTE = 0x22;
    P_MTU0.TCNT_0 = 0x0000;
    P_MTU0.TGRA_0 = pul_cyc0;      /* Set general register */
    P_MTU0.TGRB_0 = pul_duty0b;
    P_MTU0.TGRC_0 = pul_duty0c;
    P_MTU0.TGRD_0 = pul_duty0d;
    P_MTU0.TMDR_0.BYTE = 0xc3;      /* PWM mode2 */
    P_MTU1.TCR_1.BYTE = 0x60;     /* Counter clear by TGRA_0 */
    P_MTU1.TIORH_1.BYTE = 0x20;
    P_MTU1.TIORL_1.BYTE = 0x22;
    P_MTU1.TCNT_1 = 0x0000;
    P_MTU1.TGRA_1 = pul_dutyla;
    P_MTU1.TGRB_1 = pul_duty1b;
    P_MTU1.TMDR_1.BYTE = 0xc3;      /* PWM mode2 */
    P_MTU2.TCR_2.BYTE =0x60;     /* Counter clear by TGRA_0 */
    P_MTU2.TIORH_2.BYTE = 0x20;
    P_MTU2.TIORL_2.BYTE = 0x22;
    P_MTU2.TCNT_2 = 0x0000;
```
P_MTU2.TGRA_2 = pul_duty2a;  /* Set general register */
P_MTU2.TGRB_2 = pul_duty2b;
P_MTU2.TMDR_2.BYTE = 0xc3;  /* PWM mode2 */
P_MTU34.TSYR.BYTE = 0x07;
P_MTU34.TSTR.BYTE = 0x07;  /* Start timer counter */
while(1);
}
2.5 Positive-Phase/Negative Phase PWM 3-Phase Output

Specifications

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

(2) When operating with on-chip peripheral clock $P\phi = 20.0 \text{ MHz}$, the output pulse period can be set arbitrarily in the range $100.0 \text{ ns}$ to $3.27 \text{ ms}$.

![Diagram of Positive-Phase/Negative-Phase PWM 3-Phase Output Waveforms]

**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.

![Block Diagram of MTU/ch3, ch4](image-url)
(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>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIOC3B</td>
<td>Pin</td>
<td>PWM output 1</td>
</tr>
<tr>
<td>TIOC3D</td>
<td>Pin</td>
<td>Negative-phase waveform of PWM output 1</td>
</tr>
<tr>
<td>TIOC4A</td>
<td>Pin</td>
<td>PWM output 2</td>
</tr>
<tr>
<td>TIOC4B</td>
<td>Pin</td>
<td>PWM output 3</td>
</tr>
<tr>
<td>TIOC4C</td>
<td>Pin</td>
<td>Negative-phase waveform of PWM output 2</td>
</tr>
<tr>
<td>TIOC4D</td>
<td>Pin</td>
<td>Negative-phase waveform of PWM output 3</td>
</tr>
<tr>
<td>TCR_3</td>
<td>Register</td>
<td>Selection of ch3 timer counter clearing source and input clock</td>
</tr>
<tr>
<td>TMDR_3</td>
<td>Register</td>
<td>Ch3 set to operate in reset-synchronized PWM mode</td>
</tr>
<tr>
<td>TGRA_3</td>
<td>Register</td>
<td>PWM period setting</td>
</tr>
<tr>
<td>TGRB_3</td>
<td>Registers</td>
<td>Duty value setting</td>
</tr>
<tr>
<td>TGRA_4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRB_4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 SH7046 hardware and software processing as shown in the figure.

Figure 2.15   Principles of Operation of Reset-Synchronized PWM Waveforms
Software

(1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>rst_pwm</td>
<td>PFC and PWM output setting</td>
</tr>
</tbody>
</table>

(2) Arguments

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function</th>
<th>Data Length</th>
<th>Module</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>pul_cyc1</td>
<td>Used to set timer value for pulse period</td>
<td>1 word</td>
<td>Main routine</td>
<td>Input</td>
</tr>
<tr>
<td></td>
<td>Pulse period is calculated using following equation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pulse period (ns) = timer value × φ period</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(50.0 ns at 20.0 MHz operation)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty3b</td>
<td>Used to set TIOC pin output waveform transition timing</td>
<td>1 word</td>
<td>Main routine</td>
<td>Input</td>
</tr>
<tr>
<td>pul_duty4a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty4b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### (3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PORTE.PEIORL</td>
<td>Used to set multiplex insert as timer output pins TIOC3B/D, TIOC4A/B/C/D</td>
<td>H'FFFF83B4</td>
<td>H'fa00</td>
</tr>
<tr>
<td>P_PORTE.PECRL1</td>
<td></td>
<td>H'FFFF83B8</td>
<td>H'5544</td>
</tr>
<tr>
<td>P_MTU34.TCR_3</td>
<td>Used to select TGRA_3 compare match as timer counter clearing source, and Pφ/1 as input clock</td>
<td>H'FFFF8200</td>
<td>H'20</td>
</tr>
<tr>
<td>P_MTU34.TOCR</td>
<td>Enabling of toggle output synchronized with PWM period, and positive-phase/negative-phase output level setting</td>
<td>H'FFFF820B</td>
<td>H'43</td>
</tr>
<tr>
<td>P_MTU34.TGRA_3</td>
<td>PWM period setting</td>
<td>H'FFFF8218</td>
<td>pul_cyc1</td>
</tr>
<tr>
<td>P_MTU34.TGRB_3</td>
<td>Used to set timer counter value for toggle output from TIOC3B/D</td>
<td>H'FFFF821A</td>
<td>pul_duty3b</td>
</tr>
<tr>
<td>P_MTU34.TGRA_4</td>
<td>Used to set timer counter value for toggle output from TIOC4A/C</td>
<td>H'FFFF821C</td>
<td>pul_duty4a</td>
</tr>
<tr>
<td>P_MTU34.TGRB_4</td>
<td>Used to set timer counter value for toggle output from TIOC4B/D</td>
<td>H'FFFF821E</td>
<td>pul_duty4b</td>
</tr>
<tr>
<td>P_MTU34.TOER</td>
<td>Sets enabling of reset-synchronized PWM output</td>
<td>H'FFFF821E</td>
<td>H'ff</td>
</tr>
<tr>
<td>P_MTU34.TMDR_3</td>
<td>Sets reset-synchronized PWM mode</td>
<td>H'FFFF8202</td>
<td>H'c8</td>
</tr>
<tr>
<td>P_STBY.MSTCR2</td>
<td>Module standby mode clearing</td>
<td>H'FFFF861E</td>
<td>H'd2fd</td>
</tr>
</tbody>
</table>

### (4) RAM Used

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

**Note:** SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

```
rst_pwm

MSTCR2 ← H'd2fd
-------- Clear MTU module standby mode

PEIORL ← H'fa00
PECRL1 ← H'5544

TSTR ← H'00
-------- Stop timer counter

TCR_3 ← H'20
-------- Set TGRA_3 as counter clearing source in TCR

TCNT_3 ← H'0000
TCNT_4 ← H'0000
-------- Clear timer counters 3/4

TGRA_3 ← pul_cyc1
-------- Set pulse period in P_MTU34.TGRA_3

TGRB_3 ← pul_duty3b
TGRA_4 ← pul_duty4a
TGRB_4 ← pul_duty4b
-------- Set duty values in P_MTU34.TGRB_3,
-------- TGRA/B_4

TOCR ← H'43
-------- Set enabling of toggle output synchronized with
-------- PWM period, and positive-phase/negative-phase
-------- output level, in P_MTU34.TOCR

TMDR_3 ← H'c8
-------- Set P_MTU34.TMDR to reset-synchronized
-------- PWM mode

TOER ← H'ff
-------- Enable reset-synchronized PWM output with
-------- P_MTU34.TOER

TSTR ← H'40
-------- Enable ch3 count operation with
-------- P_MTU34.TSTR
```
Program Listing

****************************************************************************
/*                                INCLUDE FILE                              */
****************************************************************************
#include<machine.h>
#include"iodefine_7046.h"
****************************************************************************
/*                                 PROTOTYPE                                */
****************************************************************************
void rst_pwm(void);
****************************************************************************
/*                               RAM ALLOCATION                             */
****************************************************************************
#define pul_cyc1  (*(unsigned short *)0xffffd000)
#define pul_duty3b  (*(unsigned short *)0xffffd002)
#define pul_duty4a  (*(unsigned short *)0xffffd004)
#define pul_duty4b  (*(unsigned short *)0xffffd006)
****************************************************************************
/*                                 MAIN PROGRAM                             */
****************************************************************************
void rst_pwm(void)
{
    P_STBY.MSTCR2.WORD = 0xd2fd;    /* Clear 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 = 0x00;
    P_MTU34.TCNT_3 = 0x0000;    /* Clear timer counter3 */
    P_MTU34.TCR_4.BYTE = 0x0000;  /* Clear timer counter4 */
    P_MTU34.TGRA_3 = pul_cyc1;    /* Set 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 = 0x43;    /* Set timer output control register */
    P_MTU34.TMDR_3.BYTE = 0xc8;  /* Reset synchronized PWM mode */
    P_MTU34.TOER.BYTE = 0xff;   /* Timer output enable */
    P_MTU34.TSTR = 0x40;    /* Start timer counter */
    while(1);
}

Rev. 1.00, 07/03, page 42 of 160
2.6 Complementary PWM 3-Phase Output

| Complementary PWM 3-Phase Output | MCU: SH7046/47 | 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 can be changed between 0% and 100% by setting an arbitrary value in RAM.

\[
\text{Duty} = \frac{\text{Pulse high width}}{\text{Pulse period}} \times 100 \% 
\]

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

(4) When operating with on-chip peripheral clock \( P\phi = 20.0 \, \text{MHz} \), the output pulse period can be set arbitrarily in the range 100.0 ns to 3.27 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 MTU/ch3 and 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

![Block Diagram of MTU/ch3, ch4](image)

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>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIOC3A</td>
<td>Pin</td>
<td>Toggle output synchronized with PWM period</td>
</tr>
<tr>
<td>TIOC3C</td>
<td>Pin</td>
<td>PWM output 1</td>
</tr>
<tr>
<td>TIOC3D</td>
<td>Pin</td>
<td>Negative-phase waveform in non-overlapping relationship with PWM output 1</td>
</tr>
<tr>
<td>TIOC4A</td>
<td>Pin</td>
<td>PWM output 2</td>
</tr>
<tr>
<td>TIOC4B</td>
<td>Pin</td>
<td>PWM output 3</td>
</tr>
<tr>
<td>TIOC4C</td>
<td>Pin</td>
<td>Negative-phase waveform in non-overlapping relationship with PWM output 2</td>
</tr>
<tr>
<td>TIOC4D</td>
<td>Pin</td>
<td>Negative-phase waveform in non-overlapping relationship with PWM output 3</td>
</tr>
<tr>
<td>TOCR</td>
<td>Register</td>
<td>Enabling/disabling of toggle output synchronized with PWM period</td>
</tr>
<tr>
<td>TOER</td>
<td>Register</td>
<td>Complementary PWM output pin signal output enabling/disabling</td>
</tr>
<tr>
<td>TCR_3</td>
<td>Register</td>
<td>Selection of ch3 timer counter clearing source and input clock</td>
</tr>
<tr>
<td>TMDR_3</td>
<td>Register</td>
<td>Ch3, ch4 set to complementary PWM mode operation</td>
</tr>
<tr>
<td>TGRA_3</td>
<td>Register</td>
<td>Used to set value of 1/2 PWM period + dead time</td>
</tr>
<tr>
<td>TGRC_3</td>
<td>Register</td>
<td>TGRA_3 buffer register</td>
</tr>
<tr>
<td>TGRB_3</td>
<td>Registers</td>
<td>Output pulse transition point setting (compare register)</td>
</tr>
<tr>
<td>TGRA_4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRB_4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRD_3</td>
<td>Register</td>
<td>TGRB_3 buffer register</td>
</tr>
<tr>
<td>TGRC_4</td>
<td>Register</td>
<td>TGRA_4 buffer register</td>
</tr>
<tr>
<td>TGRD_4</td>
<td>Register</td>
<td>TGRB_4 buffer register</td>
</tr>
<tr>
<td>TDDR</td>
<td>Register</td>
<td>Dead time setting</td>
</tr>
<tr>
<td>TCDR</td>
<td>Register</td>
<td>Setting of 1/2 period</td>
</tr>
<tr>
<td>TCBR</td>
<td>Register</td>
<td>TCDR buffer register</td>
</tr>
</tbody>
</table>
Operation

(1) Figure 2.18 illustrates the principles of operation. Complementary PWM waveform output is performed by SH7046 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 (at the point at which data is set in TGRD_4) is 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.
- When period Tb ends, a value transferred to a temporary register is transferred to a compare match register. This transfer timing can be selected with timer mode register (TMDR) bits MD3 to MD0.

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)).
(3) Figure 2.20 illustrates the principles of operation. Complementary PWM waveform output is performed by SH7046 hardware and software processing. The transfer mode selected in this sample task is the mode in which data is changed at a peak.

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

Rev. 1.00, 07/03, page 48 of 160
(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 SH7046 hardware and software processing as shown in the figure.

Figure 2.21   Principles of Operation of PWM Waveforms

Rev. 1.00, 07/03, page 49 of 160
(5) Figure 2.22 illustrates the principles of operation. Toggle output synchronized with the PWM period is performed by SH7046 hardware and software processing.

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

#### (1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>comple</td>
<td>Complementary PWM output setting</td>
</tr>
<tr>
<td>Data setting</td>
<td>setdata</td>
<td>Sets waveform transition timing in buffer register</td>
</tr>
</tbody>
</table>

#### (2) Arguments

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function</th>
<th>Data Length</th>
<th>Module</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>pul_cyc1</td>
<td>Used to set pulse 1/2 period + dead time value</td>
<td>1 word</td>
<td>Main routine</td>
<td>Input</td>
</tr>
<tr>
<td></td>
<td>Pulse period is calculated using following equation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pulse period (ns) = timer value × φ period</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(50.0 ns at 20.0 MHz operation)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty3d</td>
<td>Used to set TIOC pin output waveform transition timing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty4c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pul_duty4d</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c_cyc</td>
<td>PWM carrier period register value setting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dead_time</td>
<td>Non-overlap time setting</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### (3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>MTU module standby mode clearing, and setting of MTU to operational status</td>
<td>H'FFFF861E</td>
<td>H'd2fd</td>
</tr>
<tr>
<td>P_PORTE.PECRH</td>
<td>Used to set multiplex pins as MTU timer output pins TIOC3A, TIOC3B, TIOC3D, TIOC4A, TIOC4B, TIOC4C, TIOC4D</td>
<td>H'FFFF83BC</td>
<td>H'0000</td>
</tr>
<tr>
<td>P_PORTE.PECRL1</td>
<td></td>
<td>H'FFFF83B8</td>
<td>H'55545</td>
</tr>
<tr>
<td>P_PORTE.PECRL2</td>
<td></td>
<td>H'FFFF83BA</td>
<td>H'0000</td>
</tr>
<tr>
<td>P_PORTE.PEIORH</td>
<td></td>
<td>H'FFFF83B6</td>
<td>H'0000</td>
</tr>
<tr>
<td>P_PORTE.PEIORL</td>
<td></td>
<td>H'FFFF83B4</td>
<td>H'fb00</td>
</tr>
<tr>
<td>P_MTU34.TCR_3</td>
<td>Selects timer counter clearing source and input clock</td>
<td>H'FFFF8200</td>
<td>H'00</td>
</tr>
<tr>
<td>P_MTU34.TCR_4</td>
<td>Selects timer counter clearing source and input clock</td>
<td>H'FFFF8201</td>
<td>H'00</td>
</tr>
<tr>
<td>P_MTU34.TIER_3</td>
<td>Enables TGR3A interrupt</td>
<td>H'FFFF8208</td>
<td>H'01</td>
</tr>
<tr>
<td>Register Name</td>
<td>Function</td>
<td>Address</td>
<td>Set Value</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------------------------------</td>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>P_MTU34.TGRA_3</td>
<td>Used to set 1/2 carrier period + dead time register value</td>
<td>H'FFFF8218</td>
<td>pul_cyc1</td>
</tr>
<tr>
<td>P_MTU34.TGRC_3</td>
<td>Used to set 1/2 carrier period + dead time register value</td>
<td>H'FFFF8224</td>
<td>pul_cyc1</td>
</tr>
<tr>
<td>P_MTU34.TGRB_3</td>
<td>Setting of PWM duty value of waveform output from TIOC3B, TIOC3D</td>
<td>H'FFFF821A</td>
<td>pul_duty3d</td>
</tr>
<tr>
<td>P_MTU34.TGRD_3</td>
<td>Setting of PWM duty value of waveform output from TIOC3B, TIOC3D</td>
<td>H'FFFF8226</td>
<td>pul_duty3d</td>
</tr>
<tr>
<td>P_MTU34.TGRA_4</td>
<td>Setting of PWM duty value of waveform output from TIOC4A, TIOC4C</td>
<td>H'FFFF821C</td>
<td>pul_duty4c</td>
</tr>
<tr>
<td>P_MTU34.TGRC_4</td>
<td>Setting of PWM duty value of waveform output from TIOC4A, TIOC4C</td>
<td>H'FFFF821C</td>
<td>pul_duty4c</td>
</tr>
<tr>
<td>P_MTU34.TGRB_4</td>
<td>Setting of PWM duty value of waveform output from TIOC4B, TIOC4D</td>
<td>H'FFFF821E</td>
<td>pul_duty4d</td>
</tr>
<tr>
<td>P_MTU34.TGRD_4</td>
<td>Setting of PWM duty value of waveform output from TIOC4B, TIOC4D</td>
<td>H'FFFF821E</td>
<td>pul_duty4d</td>
</tr>
<tr>
<td>P_MTU34.TCNT_3</td>
<td>Dead time value setting</td>
<td>H'FFFF8210</td>
<td>dead_time</td>
</tr>
<tr>
<td>P_MTU34.TDDR</td>
<td>Dead time value setting</td>
<td>H'FFFF8216</td>
<td>dead_time</td>
</tr>
<tr>
<td>P_MTU34.TCDR</td>
<td>Setting of upper limit value of timer counter TCNT_4 (1/2 carrier period)</td>
<td>H'FFFF8214</td>
<td>c_cyc</td>
</tr>
<tr>
<td>P_MTU34.TCBR</td>
<td>Setting of upper limit value of timer counter TCNT_4 (1/2 carrier period)</td>
<td>H'FFFF8222</td>
<td>c_cyc</td>
</tr>
<tr>
<td>P_MTU34.TOCR</td>
<td>Enabling of toggle output synchronized with PWM period, and positive-phase/negative phase output level setting</td>
<td>H'FFFF820B</td>
<td>H'43</td>
</tr>
<tr>
<td>P_MTU34.TOER</td>
<td>Complementary PWM output enabling setting</td>
<td>H'FFFF820A</td>
<td>H'ff</td>
</tr>
<tr>
<td>P_MTU34.TMDR_3</td>
<td>Complementary PWM mode setting</td>
<td>H'FFFF8202</td>
<td>H'ff</td>
</tr>
<tr>
<td>P_INTC.IPRE</td>
<td>Sets 15 as MTU channel 3 interrupt priority level</td>
<td>H'FFFF8350</td>
<td>H'00f0</td>
</tr>
</tbody>
</table>

(4) RAM Used

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

Note: SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

- **comple**
  - MSTCR2 ← H\textquotesingle d2fd
    - Clear MTU module standby mode
  - PEORH ← H'0000
    - Set TIOC3B/D, TIOC4A/B/C/D as MTU output pins in PEORH, PEORL
  - PECRL1 ← H'5545
  - PECRL2 ← H'0000
  - PECRH ← H'0000
  - PEIORH ← H'0000
  - PEIORL ← HFB00
  - TSTR ← H'00
    - Stop timer counter operation
  - TCR_3 ← H'00
    - Set counter clearing disabling in P_MTU34.TCR_3/4
  - TCR_4 ← H'00
  - TCNT_3 ← dead_time
    - Set dead time value in TCNT_3
  - TCNT_4 ← H'0000
    - Clear TCNT_4
  - TGRD_3/TGRB_3 ← pul_duty3d
    - Set duty value in duty register and buffer register
  - TGRC_4/TGRA_4 ← pul_duty4c
  - TGRD_4/TGRB_4 ← pul_duty4d
  - TDDR ← dead_time
    - Set dead time value in P_MTU34.TDDR, set 1/2 period value in P_MTU34.TCDR and P_MTU34.TCBR
  - TCDR/TCBR ← c_cyc
    - Set 1/2 period + non-overlap value in P_MTU34.TGRA/C_3
  - TGRA_3/TGRC_3 ← pul_cyc1
    - Set enabling of toggle output synchronized with PWM period, and positive-phase/negative phase output level, in P_MTU34.TOCR
  - TOCR ← H'43
    - Set complementary PWM mode in P_MTU34.TMDR_3
  - TMDR_3 ← H'ff
    - Enable complementary PWM waveform output with P_MTU34.TOER
  - TOER ← H'ff
  - TIER_3 ← H'01
    - Enable TGIA_3 interrupt request with P_MTU34.TIER_3
  - IPRE ← H'0080
    - Set 15 as TGIA_0 interrupt priority level in P_INTC.IPRE
  - TSTR ← H'c0
    - Enable ch3/4 count operation with P_MTU34.TSTR
Program Listing

```c
/*---------------------------------------------*/
/* INCLUDE FILE */
/*---------------------------------------------*/
#include <machine.h>
#include "iodefine_7046.h"
/*---------------------------------------------*/
/* PROTOTYPE */
/*---------------------------------------------*/
extern void comple(void);
#pragma interrupt(setdata)
/*---------------------------------------------*/
/* RAM ALLOCATION */
/*---------------------------------------------*/
#define pul_cyc1           (*(unsigned short *)0xffffd000)
#define pul_duty3d        (*(unsigned short *)0xffffd002)
#define pul_duty4c  (*(unsigned short *)0xffffd004)
#define pul_duty4d  (*(unsigned short *)0xffffd006)
#define c_cyc  (*(unsigned short *)0xffffd008)
#define dead_time  (*(unsigned short *)0xffffd00a)
/*---------------------------------------------*/
/* MAIN PROGRAM */
/*---------------------------------------------*/
void comple(void)
{
    P_STBY.MSTCR2 = 0xd2fd;  /* MTU module stop mode clear */
    P_PORTE.PECRH.WORD = 0x0000;  /* TIOC3A/B,D,TIOC4A/B/C/D output */
    P_PORTE.PECRL1.WORD = 0x5545;
    P_PORTE.PECRL2.WORD = 0x0000;
    P_PORTE.PEIORH.WORD = 0x0000;
    P_PORTE.PEIORL.WORD = 0xFB00;
    P_MTU34.TSTR.BYTE = 0x00;
    P_MTU34.TCR_3.BYTE = 0x00;  /* not clear */
    P_MTU34.TCR_4.BYTE = 0x00;  /* not clear */
    P_MTU34.TCNT_3 = dead_time;  /* initial data */
    P_MTU34.TCNT_4 = 0x0000;
    P_MTU34.TGRD_3 = pul_duty3d;  /* PWM output1 compare register */
    P_MTU34.TGRB_3 = pul_duty3d;  /* PWM output1 compare register */
    P_MTU34.TGRA_4 = pul_duty4c;  /* PWM output2 compare register */
    P_MTU34.TGRB_4 = pul_duty4c;  /* PWM output2 compare register */
    P_MTU34.TGRD_4 = pul_duty4d;  /* PWM output3 compare register */
    P_MTU34.TGRD_4 = pul_duty4d;  /* PWM output3 compare register */
    P_MTU34.TDDR = dead_time;  /* dead time set */
    P_MTU34.TCRD = c_cyc;  /* 1/2 carrier period */
    P_MTU34.TCRB = 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.TMCR.BYTE = 0xff;  /* complementary-pwm mode */
    P_MTU34.TNR.BYTE = 0xff;  /* timer output enable register */
}
```

Rev. 1.00, 07/03, page 54 of 160
/* timer interrupt enable register */
P_MTU34.TIER_3.BYTE = 0x01;

/* set interrupt level = 15 */
P_INTC.IPRE.WORD = 0x00f0;
set_imask(0x0);

/* set imask level = 0 */
P_MTU34.TSTR.BYTE = 0xc0;

/* timer start */
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: SH7046/47 | 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 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](image)

Rev. 1.00, 07/03, page 56 of 160
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)
(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 the timer value at that point 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
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>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCLKA</td>
<td>Pin</td>
<td>External clock input pins</td>
</tr>
<tr>
<td>TCLKB</td>
<td>Pin</td>
<td></td>
</tr>
<tr>
<td>TSTR</td>
<td>Register</td>
<td>Enabling/disabling of ch0, ch1 timer counter operation</td>
</tr>
<tr>
<td>TCR_0</td>
<td>Register</td>
<td>Selection of counter clock and counter clearing source</td>
</tr>
<tr>
<td>TIORH_0</td>
<td>Register</td>
<td>TIOC0A output compare setting. Setting of TIOC0B for input capture on ch0 output compare occurrence</td>
</tr>
<tr>
<td>TIORL_0</td>
<td>Register</td>
<td>TIOC0C output compare setting</td>
</tr>
<tr>
<td>TGRA_0</td>
<td>Register</td>
<td>Measurement time 1 setting</td>
</tr>
<tr>
<td>TGRB_0</td>
<td>Register</td>
<td>Count result stored on input capture B</td>
</tr>
<tr>
<td>TGRC_0</td>
<td>Register</td>
<td>Measurement time 2 setting</td>
</tr>
<tr>
<td>TMDR_1</td>
<td>Register</td>
<td>Phase counting mode setting</td>
</tr>
<tr>
<td>TCR_1</td>
<td>Register</td>
<td>Selection of counter clock and counter clearing source</td>
</tr>
<tr>
<td>TIOR_1</td>
<td>Register</td>
<td>Setting of TIOC1A/C for input capture on ch1 output compare occurrence</td>
</tr>
<tr>
<td>TIER_1</td>
<td>Register</td>
<td>Enables TIOC1A/B, TCIU_1, TCIV_1 interrupts</td>
</tr>
<tr>
<td>TGRA_1</td>
<td>Register</td>
<td>Count result storage on input capture A</td>
</tr>
<tr>
<td>TGRB_1</td>
<td>Register</td>
<td></td>
</tr>
</tbody>
</table>
Operation

(1) Figure 2.26 illustrates the principles of operation. A counter is incremented or decremented by SH7046 hardware and software processing.

![Figure 2.26 Principles of Operation in Phase Counting Mode (1)](image_url)

Rev. 1.00, 07/03, page 60 of 160
(2) Interrupt handling is executed on external event occurrence by means of SH7046 hardware and software processing as shown in figure 2.27.

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

### (1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>en2</td>
<td>Initialization of MTU, etc.</td>
</tr>
<tr>
<td>Counter value</td>
<td>phacnt1</td>
<td>Initiated by TGIA_1. Sets up/down-count result in RAM based on TGRA value. Sets counter period result in RAM based on TGRC value</td>
</tr>
<tr>
<td>measurement 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Counter value</td>
<td>phacnt2</td>
<td>Initiated by TGIB_1. Sets up/down-count result in RAM based on TGRB value</td>
</tr>
<tr>
<td>measurement 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>ovf1</td>
<td>Initiated by TCIV_1. Software counter incrementing</td>
</tr>
<tr>
<td>Underflow</td>
<td>unf1</td>
<td>Initiated by TCIU_1. Software counter decrementing</td>
</tr>
</tbody>
</table>

### (2) Arguments

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function Assignment</th>
<th>Data Length</th>
<th>Module</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>msr_tim1</td>
<td>Used to set timer value for counter measurement time</td>
<td>Word</td>
<td>Main routine</td>
<td>Input</td>
</tr>
<tr>
<td>msr_tim2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Measurement time is calculated using following equation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Measurement time (ns) = timer value × φ period</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(50.0 ns at 20.0 MHz operation)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cnt_data1</td>
<td>Used to set up/down-count results</td>
<td>Longword</td>
<td>Counter value measurement 1</td>
<td>Output</td>
</tr>
<tr>
<td>cnt_data2</td>
<td></td>
<td></td>
<td>Counter value measurement 2</td>
<td></td>
</tr>
<tr>
<td>p_cycle</td>
<td>Used to set count period result</td>
<td>Word</td>
<td>Counter value measurement 2</td>
<td></td>
</tr>
</tbody>
</table>

---

Rev. 1.00, 07/03, page 62 of 160
### (3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>MTU module standby mode clearing, and setting of MTU to operational status</td>
<td>H'FFFF861E</td>
<td>H'd2fd</td>
</tr>
<tr>
<td>P_PORTA.PACRL2</td>
<td>Used to set multiplex pins as timer pins</td>
<td>H'FFFF838E</td>
<td>H'5000</td>
</tr>
<tr>
<td>P_PORTA.PACRL3</td>
<td>TCLKA, TCLKB</td>
<td>H'FFFF838A</td>
<td>H'0000</td>
</tr>
<tr>
<td>P_MTU0.TCR_0</td>
<td>Selection of counter clock and counter clearing source</td>
<td>H'FFFF8260</td>
<td>H'20</td>
</tr>
<tr>
<td>P_MTU0.TIORH_0</td>
<td>TIOC0A output compare setting. Setting of TIOBC0B for input capture on ch0 output compare</td>
<td>H'FFFF8262</td>
<td>H'f0</td>
</tr>
<tr>
<td>P_MTU0.TIORL_0</td>
<td>TIOC0C output compare setting</td>
<td>H'FFFF8263</td>
<td>H'00</td>
</tr>
<tr>
<td>P_MTU0.TGRA_0</td>
<td>Measurement time 1 setting</td>
<td>H'FFFF8268</td>
<td>msr_tim1</td>
</tr>
<tr>
<td>P_MTU0.TGRC_0</td>
<td>Measurement time 2 setting</td>
<td>H'FFFF826C</td>
<td>msr_tim2</td>
</tr>
<tr>
<td>P_MTU1.TMDR_1</td>
<td>Phase counting mode setting</td>
<td>H'FFFF8281</td>
<td>H'04</td>
</tr>
<tr>
<td>P_MTU0.TMDR_0</td>
<td>Sets buffer operation for GRD</td>
<td>H'FFFF8261</td>
<td>H'20</td>
</tr>
<tr>
<td>P_MTU1.TIOR_1</td>
<td>Setting of TIOC0A/C for input capture on ch1 output compare occurrence</td>
<td>H'FFFF8282</td>
<td>H'ff</td>
</tr>
<tr>
<td>P_MTU1.TIER_1</td>
<td>Enables interrupts by TGI/B_1, TCIU_1, TCIV_1</td>
<td>H'FFFF8284</td>
<td>H'33</td>
</tr>
<tr>
<td>P_MTU34.TSTR</td>
<td>Starts ch0, ch1 timer count</td>
<td>H'FFFF8240</td>
<td>H'03</td>
</tr>
<tr>
<td>P_INTC.IPRD</td>
<td>Sets 15 as MTU0, MTU1 interrupt priority level</td>
<td>H'FFFF834E</td>
<td>H'00ff</td>
</tr>
</tbody>
</table>

### (4) RAM Used

<table>
<thead>
<tr>
<th>Module</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter value measurement 1, 2</td>
<td>wrk</td>
<td>Used as work area for data setting</td>
</tr>
<tr>
<td>All modules</td>
<td>cnt</td>
<td>Software counter</td>
</tr>
</tbody>
</table>

**Note:** SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

```
<table>
<thead>
<tr>
<th>Block</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>en2</td>
<td>Clear MTU module standby mode</td>
</tr>
<tr>
<td>MSTCR2 ← H'2fd</td>
<td></td>
</tr>
<tr>
<td>TCR_0 ← H'20</td>
<td>Set Pφ/1 as TCNT_0 input clock and compare match A as counter clearing source in P_MTU0.TCR_0</td>
</tr>
<tr>
<td>TIORH_0 ← H'f0</td>
<td>Set TIOCA for output compare operation and TIOCB for input capture operation in P_MTU0.TIORH_0</td>
</tr>
<tr>
<td>TIORL_0 ← H'00</td>
<td>TIOCB input capture is initiated by ch1 increment/decrement</td>
</tr>
<tr>
<td>TIOR_1 ← H'ff</td>
<td>Set TIOCC for output compare operation in P_MTU0.TIORL_0</td>
</tr>
<tr>
<td>TIER_1 ← H'33</td>
<td>Set TIOCA/B for input capture operation in P_MTU0.TIOR_1</td>
</tr>
<tr>
<td>TGRA_0 ← msr_tim1</td>
<td>TIOC1A input capture is initiated by ch0 output compare</td>
</tr>
<tr>
<td>TGRC_0 ← msr_tim2</td>
<td>TIOC1B input capture is initiated by ch0 output compare</td>
</tr>
<tr>
<td>IPRD ← H'00ff</td>
<td>Enable TGI/A/B_1, TCIU_1, TCIV_1 interrupts with P_MTU1.TIER_1</td>
</tr>
<tr>
<td>PACRL2 ← 5000</td>
<td>Set counter measurement times in P_MTU0.TGRA/C_0</td>
</tr>
<tr>
<td>PACRL3 ← 0000</td>
<td></td>
</tr>
<tr>
<td>TMDR_1 ← H'04</td>
<td>Set 15 as interrupt priority level in INTC.IPRD</td>
</tr>
<tr>
<td>TMDR_0 ← H'20</td>
<td>Enable pulse input from TCLKA/B pins</td>
</tr>
<tr>
<td>TSTR ← H'03</td>
<td>Set phase counting mode in P_MTU1.TMDR_1</td>
</tr>
<tr>
<td></td>
<td>Set buffer operation for TGRD in P_MTU0.TMDR_0</td>
</tr>
<tr>
<td></td>
<td>Enable ch0/1 count operation</td>
</tr>
</tbody>
</table>
```

Rev. 1.00, 07/03, page 64 of 160
(2) Counter value measurement 1 (ch1 input capture B interrupt)

```
TSR_1 &= H'fe
wrk ← TGRB_1

cnt < 0?

Yes

cnt_data1 ← (unsigned long)wrk - 0x010000 + cnt * 0x010000

No

Ch1 down-count?

Set ch1 count result in RAM

Set pulse width in RAM

Capture data capture from buffer register
TGRD_0
TGRD_0 is TGRB_0 buffer register
```

```
Cnt_data1 ← (unsigned long)wrk + cnt * 0x010000

phacnt1

Clear interrupt request flag

Capture data capture

p_cycle ← TGRD_0

RTE

Set ch1 count result in RAM

Set ch1 count result in RAM

Set ch1 count result in RAM
```
(3) Counter value measurement 2 (ch1 input capture B interrupt)

```
phacnt2

TSR_1 &= H'fd

wrk ← TGRA_1

cnt < 0?

Yes

cnt_data2(unsignedlong)wrk +
cnt * 0x010000

RTE

No

Ch1 up-count?

Set ch1 count result in RAM

cnt_data2 ← (unsignedlong)wrk -
0x010000 + cnt * 0x010000

Clear interrupt request flag

Capture data capture
```

(4) Ch1 overflow interrupt

```
ovf1

TSR_1 &= H'ef

Clear interrupt request flag

cnt++

Increment software counter

RTE
```
(5) Ch1 underflow interrupt

```
unf1

TSR_1 &= H'df  Clear interrupt request flag

cnt − −  Decrement software counter

RTE
```
Program Listing

/*----------------------------------------------------------------*/
/*                           INCLUDE FILE                         */
 /*----------------------------------------------------------------*/
#include <machine.h>
#include "iodefine_7046.h"
 /*----------------------------------------------------------------*/
/*                            PROTOTYPE                           */
 /*----------------------------------------------------------------*/
void en2(void);
#pragma interrupt(phacnt1,phacnt2,ovf1,unf1)
 /*----------------------------------------------------------------*/
/*                          RAM ALLOCATION                        */
 /*----------------------------------------------------------------*/
#define msr_tim1  (*(unsigned short *)0xffffd000)
#define msr_tim2  (*(unsigned short *)0xffffd002)
#define cnt_data2  (*(signed long *)0xffffd004)
#define cnt_data1  (*(signed long *)0xffffd008)
#define p_cycle  (*(unsigned long *)0xffffd00c)
#define cnt  (*(signed long *)0xffffd010)
#define wrk  (*(unsigned short *)0xffffd014)
 /*----------------------------------------------------------------*/
/*                         MAIN PROGRAM                           */
 /*----------------------------------------------------------------*/
void en2(void)
{"P_STBY.MSTCR2.WORD = 0xd2fd; /* MTU module stop mode clear */
P_MTU0.TCR_0.BYTE = 0x20; /* timer clear output compare TGRA_0 */
P_MTU0.TIORH_0.BYTE = 0xf0; /* output compare TIOC0A */
P_MTU0.TIORL_0.BYTE = 0x00; /* output compare TIOC0C */
P_MTU1.TIOR_1.BYTE = 0xff; /* input capture TIOC1A,B */
P_MTU1.TIER_1.BYTE = 0x33; /* interrupt TIOC1A,TIOC1B,TCIU1,TCIV1 */
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 sellect */
P_PORTA.PACRL3.WORD = 0x0000;
P_MTU1.TMDR_1.BYTE = 0x04; /* set phase counting mode */
P_MTU0.TMDR_0.BYTE = 0x20; /* TGRD buffer mode */
P_MTU34.TSTR.BYTE = 0x03; /* start timer 0,1 */
set_imask(0x0); /* set imask level=0 */
while(1); /* loop*/
}

Rev. 1.00, 07/03, page 68 of 160
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 */
}

void phacnt1(void)
{
    P_MTU1.TSR_1.BYTE &= 0xfe;    /* clear flag */
    wrk = P_MTU1.TGRB_1;
    if(cnt < 0)                   /* count < 0 */
        cnt_data1 = (unsigned long)wrk-0x010000+cnt*0x010000; /* set sp */
    else
        cnt_data1 = (unsigned long)wrk+cnt*0x010000;           /* set sp */
    p_cycle = P_MTU0.TGRD_0;     /* set width pulse */
}

void phacnt2(void)
{
    P_MTU1.TSR_1.BYTE &= 0xfd;    /* clear flag */
    wrk = P_MTU1.TGRA_1;
    if(cnt < 0)                   /* count < 0 */
        cnt_data2 = (unsigned long)wrk+cnt*0x010000;           /* set po */
    else
        cnt_data2 = (unsigned long)wrk-0x010000+cnt*0x010000;     /* set po */
}
2.8 Externally Triggered Timer Waveform Cutoff

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](image-url)
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.

![Block Diagram of MTU/ch3, ch4, and POE](image-url)
(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>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIOC3B</td>
<td>Pins</td>
<td>Pulse output pins</td>
</tr>
<tr>
<td>TIOC3D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC4A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC4C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIOC4D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POE0</td>
<td>Pin</td>
<td>Waveform cutoff external signal input pin</td>
</tr>
<tr>
<td>TSTR_3</td>
<td>Register</td>
<td>Enabling/disabling of ch3 timer counter operation</td>
</tr>
<tr>
<td>TCR_3</td>
<td>Register</td>
<td>Selection of ch3 timer counter clearing source and input clock</td>
</tr>
<tr>
<td>TMDR_3</td>
<td>Register</td>
<td>Sets reset-synchronized PWM mode for ch3, ch4</td>
</tr>
<tr>
<td>TGRA_3</td>
<td>Register</td>
<td>PWM period setting</td>
</tr>
<tr>
<td>TGRB_3</td>
<td>Registers</td>
<td>Output waveform transition timing setting</td>
</tr>
<tr>
<td>TGRA_4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGRB_4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOER</td>
<td>Register</td>
<td>Enabling/disabling of TIOC3B/D and TIOC4A/B/C/D pin timer output</td>
</tr>
<tr>
<td>ICSR</td>
<td>Register</td>
<td>POE input mode selection</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>down</td>
<td>DC motor control waveform generation</td>
</tr>
</tbody>
</table>

#### (2) Arguments

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function Assignment</th>
<th>Data Length</th>
<th>Module</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>cycle</td>
<td>PWM period setting</td>
<td>1 word</td>
<td>Main routine</td>
<td>Input</td>
</tr>
<tr>
<td>duk1</td>
<td>Used to set TIOC3B/D output waveform transition timing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>duk2</td>
<td>Used to set TIOC4A/C output waveform transition timing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>duk3</td>
<td>Used to set TIOC4B/D output waveform transition timing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### (3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>MTU module standby mode clearing, and setting of MTU to operational status</td>
<td>H'FFFF861E</td>
<td>H'd2fd</td>
</tr>
<tr>
<td>P_PORTE.PEIORL</td>
<td>Sets TIOC3B/D, TIOC4A/B/C/D as output pins</td>
<td>H'FFFF83B4</td>
<td>H'fa00</td>
</tr>
<tr>
<td>P_PORTE.PECRL1</td>
<td>Sets TIOC3B/D, TIOC4A/B/C/D as MTU output pins</td>
<td>H'FFFF83B8</td>
<td>H'5544</td>
</tr>
<tr>
<td>P_PORTB.PBCR2</td>
<td>Sets POE0 pin</td>
<td>H'FFFF839A</td>
<td>H'0020</td>
</tr>
<tr>
<td>P_MTU34.TCR_3</td>
<td>Selection of timer counter clearing source and input clock</td>
<td>H'FFFF8200</td>
<td>H'20</td>
</tr>
<tr>
<td>P_MTU34.TOCR</td>
<td>Enabling of toggle output synchronized with PWM period, and positive-phase/negative-phase output level setting</td>
<td>H'FFFF820B</td>
<td>H'00</td>
</tr>
<tr>
<td>P_MTU34.TGRA_3</td>
<td>PWM period setting</td>
<td>H'FFFF8218</td>
<td>cycle</td>
</tr>
<tr>
<td>P_MTU34.TGRB_3</td>
<td>Used to set TIOC3B, TIOC3D output waveform transition timing</td>
<td>H'FFFF821A</td>
<td>duk1</td>
</tr>
<tr>
<td>P_MTU34.TGRA_4</td>
<td>Used to set TIOC4A, TIOC4C output waveform transition timing</td>
<td>H'FFFF821C</td>
<td>duk2</td>
</tr>
<tr>
<td>P_MTU34.TGRB_4</td>
<td>Used to set TIOC4B, TIOC4D output waveform transition timing</td>
<td>H'FFFF821E</td>
<td>duk3</td>
</tr>
<tr>
<td>P_MTU34.TOER</td>
<td>Sets TIOC3B/D, TIOC4A/B/C/D as MTU output pins</td>
<td>H'FFFF820A</td>
<td>H'ff</td>
</tr>
<tr>
<td>P_MTU34.TMDR_3</td>
<td>Sets reset-synchronized PWM mode</td>
<td>H'FFFF8202</td>
<td>H'c8</td>
</tr>
<tr>
<td>P_MTU34.ICSR1</td>
<td>Sets high-impedance output synchronized with falling edge of POE0 pin input signal</td>
<td>H'FFFF83C0</td>
<td>H'0000</td>
</tr>
</tbody>
</table>

#### (4) RAM Used

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

**Note:** SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

- **down**
- MSTCR2 ← H'd2fd  
  Clear MTU module standby mode
- ICSR1 ← H'0000  
  Set high-impedance output in synchronization with fall of POE0 pin input signal in ICSR1
- PEIORL ← H'fa00  
  Set TIOC3B/D, TIOC4A/B/C/D as output pins in PEIORL
- PECRL1 ← H'5544  
  Set TIOC3B/D, TIOC4A/B/C/D as MTU output pins in PECRL1
- PBCR2 ← H'0020  
  Set POE0 enabling in PBCR2
- TSTR ← H'00  
  Stop TCNT_3/4
- TCR_3 ← H'20  
  Set TGRA_3 as counter clearing source in TCR_3
- TCNT_3 ← H'0000  
  TCNT_4 ← H'0000  
  Clear TCNT_3/4
- TGRA_3 ← cycle  
  Set pulse period in TGRA_3
- TGRB_3 ← duk1  
  TGRA_4 ← duk2  
  TGRB_4 ← duk3  
  Set duty values in TGRB_3, TGRA_4, TGRB_4
- TOCR ← H'00  
  Set enabling of toggle output synchronized with PWM period, and positive-phase/negative-phase output levels, in TOCR
- TMDR_3 ← H'c8  
  Set reset-synchronized PWM mode in TMDR_3
- TOER ← H'ff  
  Set TIOC3B/D, TIOC4A/B/C/D as MTU output pins in TOER
- TSTR ← H'40  
  Enable MTU/ch3 count operation with TSTR
Program Listing

/*----------------------------------------------------------------*/
/*                             INCLUDE FILE                       */
 /*----------------------------------------------------------------*/
#include <machine.h>
#include "iodefine_7046.h"
 /*----------------------------------------------------------------*/
/*                             PROTOTYPE                          */
 /*----------------------------------------------------------------*/
void down(void);
 /*----------------------------------------------------------------*/
/*                          RAM ALLOCATION                        */
 /*----------------------------------------------------------------*/
#define cycle   (*(unsigned short *)0xffffd000)
#define duk1    (*(unsigned short *)0xffffd002)
#define duk2    (*(unsigned short *)0xffffd004)
#define duk3    (*(unsigned short *)0xffffd006)
 /*----------------------------------------------------------------*/
/*                            MAIN PROGRAM                        */
 /*----------------------------------------------------------------*/
void down(void)
{
    P_STBY.MSTCR2.WORD = 0xd2fd;  /* MTU module stop 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;  /* POE enable */
    P_PORTB.PBCR1.WORD = 0x0000;  /* POE enable */
    P_PORTB.PBCR2.WORD = 0x0020;  /* POE enable */
    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 = 0x20;  /* timer clear input capture TGRA_3 */
    P_MTU34.TCNT_3 = 0x0000;  /* set timer counter3 0x0000 */
    P_MTU34.TCNT_4 = 0x0000;  /* set timer counter4 0x0000 */
    P_MTU34.TGRA_3 = cycle;  /* period set */
    P_MTU34.TGRB_3 = duk1;  /* duty set */
    P_MTU34.TGRA_4 = duk2;
    P_MTU34.TGRB_4 = duk3;
    P_MTU34.TOCR.BYTE = 0x00;  /* set output level */
    P_MTU34.TMDR_3.BYTE = 0x00;  /* reset-synchronized pwm mode */
    P_MTU34.TOER.BYTE = 0xff;  /* set timer3,4 output */
    P_MTU34.TSTR.BYTE = 0x00;
    while(1);  /* loop */
}
2.9 DC Motor Control Signal Output

<table>
<thead>
<tr>
<th>DC Motor Control Signal Output</th>
<th>MCU: SH7046/47</th>
<th>Functions Used: MTU (Gate Control Register)</th>
</tr>
</thead>
</table>

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](image-url)
(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

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

![Principles of Operation of DC Motor Control Signal Output](image-url)

**Figure 2.33  Principles of Operation of DC Motor Control Signal Output**
# Software

## (1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>dc_3</td>
<td>DC motor control waveform generation</td>
</tr>
</tbody>
</table>

## (2) Arguments

<table>
<thead>
<tr>
<th>Label or Register Name</th>
<th>Function Assignment</th>
<th>Data Length</th>
<th>Module</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>cycle</td>
<td>Used to set timer value for PWM pulse period</td>
<td>1 word</td>
<td>Main routine</td>
<td>Input</td>
</tr>
<tr>
<td>duk1</td>
<td>Used to set TIOC3B/3D output waveform transition timing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>duk2</td>
<td>Used to set TIOC4A/4C output waveform transition timing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>duk3</td>
<td>Used to set TIOC4B/4D output waveform transition timing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>MTU module standby mode clearing, and setting of MTU to operational status</td>
<td>H'FFFF861E</td>
<td>H'd2fd</td>
</tr>
<tr>
<td>P_PORTE.PEIORL</td>
<td>Sets TIOC3B/D, TIOC4A/B/C/D as output pins</td>
<td>H'FFFF83B4</td>
<td>H'fa00</td>
</tr>
<tr>
<td>P_PORTE.PECRL1</td>
<td>Sets TIOC3B/D, TIOC4A/B/C/D as MTU input/output pins</td>
<td>H'FFFF83B8</td>
<td>H'5544</td>
</tr>
<tr>
<td>P_MTU34.TCR_3</td>
<td>Selection of timer counter clearing source and input clock</td>
<td>H'FFFF8200</td>
<td>H'20</td>
</tr>
<tr>
<td>P_MTU34.TOCR</td>
<td>Enabling of toggle output synchronized with PWM period, and positive-phase/negative-phase output level setting</td>
<td>H'FFFF820B</td>
<td>H'03</td>
</tr>
<tr>
<td>P_MTU34.TGRA_3</td>
<td>PWM period setting</td>
<td>H'FFFF8218</td>
<td>cycle</td>
</tr>
<tr>
<td>P_MTU34.TGRB_3</td>
<td>Used to set TIOC3B, TIOC3D output waveform transition timing</td>
<td>H'FFFF821A</td>
<td>duk1</td>
</tr>
<tr>
<td>P_MTU34.TGRA_4</td>
<td>Used to set TIOC4A, TIOC4C output waveform transition timing</td>
<td>H'FFFF821C</td>
<td>duk2</td>
</tr>
<tr>
<td>P_MTU34.TGRB_4</td>
<td>Used to set TIOC4B, TIOC4D output waveform transition timing</td>
<td>H'FFFF821E</td>
<td>duk3</td>
</tr>
<tr>
<td>P_MTU34.TOER</td>
<td>Sets TIOC3B/3D, TIOC4A/4B/4C/4D as MTU output pins</td>
<td>H'FFFF820A</td>
<td>H'ff</td>
</tr>
<tr>
<td>P_MTU34.TMDR_3</td>
<td>Sets reset-synchronized PWM mode</td>
<td>H'FFFF8202</td>
<td>H'c8</td>
</tr>
<tr>
<td>P_MTU34.TGCR</td>
<td>Enables DC motor control waveform output</td>
<td>H'FFFF820D</td>
<td>H'70</td>
</tr>
</tbody>
</table>

(4) RAM Used

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

Note: SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

- **dc_3**
  - MSTCR2 ← H'd2fd  Clear MTU module standby mode
  - PEIORL ← H'fa00  Set TIOC3B/D, TIOC4A/B/C/D as output pins in PEIORL
  - PECRL1 ← H'5544
  - PECRL2 ← H'0015  Set TIOC3B/D, TIOC4A/B/C/D as MTU output pins in PECRL1, PECRL2
  - TSTR ← H'00  Stop TCNT_3/4
  - TCR_3 ← H'20  Set TGRA_3 as counter clearing source in TCR_3
  - TGCR ← H'70  Enable DC motor control output with TGCR
  - TCNT_3 ← H'0000
  - TCNT_4 ← H'0000  Clear TCNT_3/4
  - TGRA_3 ← cycle  Set pulse period in TGRA_3
  - TGRB_3 ← duk1
  - TGRA_4 ← duk2
  - TGRB_4 ← duk3  Set duty values in TGRB_3, TGRA/B_4
  - TOCR_3 ← H'03  Set enabling of toggle output synchronized with PWM period, and positive-phase/negative-phase output levels, in TOCR_3
  - TMDR_3 ← H'c8  Set reset-synchronized PWM mode in TMDR_3
  - TOER ← H'ff  Set TIOC3B/D, TIOC4A/B/C/D as MTU output pins in TOER
  - TSTR ← H'40  Enable ch3 count operation with TSTR
Program Listing

/*-----------------------------------------------*/
/* INCLUDE FILE */
/*-----------------------------------------------*/
#include <machine.h>
#include "iodefine_7046.h"
/*-----------------------------------------------*/
/* PROTOTYPE */
/*-----------------------------------------------*/
void dc_3(void);

/*-----------------------------------------------*/
/* RAM ALLOCATION */
/*-----------------------------------------------*/
#define cycle          (*(unsigned short *)0xffffd000)
#define duk1           (*(unsigned short *)0xffffd002)
#define duk2           (*(unsigned short *)0xffffd004)
#define duk3           (*(unsigned short *)0xffffd006)

/*-----------------------------------------------*/
/* MAIN PROGRAM */
/*-----------------------------------------------*/
void dc_3(void)
{
    P_STBY.MSTCR2.WORD = 0xd2fd;  /* MTU module stop 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_PORTE.PECRL2.WORD = 0x0015;   /* TIOC0A/B/C input */
    P_MTU34.TSRT.BYTE = 0x00;
    P_MTU34.TCR_3.BYTE = 0x20;  /* timer clear input capture TGRA_3 */
    P_MTU34.TGRB_3 = duk1;   /* duty set */
    P_MTU34.TGRA_3 = duk2;
    P_MTU34.TGRB_4 = duk3;
    P_MTU34.TOCR.BYTE = 0x03;  /* set output level */
    P_MTU34.TOCR_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 */
}
2.10 Start of A/D Conversion by MTU

| Start of A/D Conversion by MTU | MCU: SH7046/47 | 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 and 4-channel scan mode are used for A/D conversion, with A/D conversion performed consecutively on channels 8 to 11.

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

Figure 2.34 Block Diagram of Voltage Measurement by SH7046
Functions Used

(1) In this sample task, A/D conversion is started by an MTU compare match.

(a) Figure 2.35 shows a block diagram of ch0. In ch0, the A/D converter is activated 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)

![Block Diagram of SH7046 ch0](image)

Figure 2.35   Block Diagram of SH7046 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 a number of channels (ch8 to ch11) (4-channel, single-cycle scan mode)

![Block Diagram of Voltage Measurement by SH7046](image-url)

**Figure 2.36** Block Diagram of Voltage Measurement by SH7046
Table 2.10 Function Assignments

<table>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN8 to AN11 Pins</td>
<td>Analog measurement pins</td>
<td></td>
</tr>
<tr>
<td>TCR_0 Register</td>
<td>Selection of counter clearing source</td>
<td></td>
</tr>
<tr>
<td>TIER_0 Register</td>
<td>Enables A/D conversion start request generation</td>
<td></td>
</tr>
<tr>
<td>TIORH_0 Register</td>
<td>Timer pin function setting</td>
<td></td>
</tr>
<tr>
<td>TGRA_0 Register</td>
<td>Sampling period setting</td>
<td></td>
</tr>
<tr>
<td>ADCR Register</td>
<td>A/D conversion mode and measurement pin setting</td>
<td></td>
</tr>
<tr>
<td>ADCSR Register</td>
<td>Selection of conversion time and activation source</td>
<td></td>
</tr>
<tr>
<td>ADTSR Register</td>
<td>Enables start of A/D0 module conversion by MTU trigger signal</td>
<td></td>
</tr>
<tr>
<td>ADDR8 to ADDR11 Registers</td>
<td>Storage of A/D conversion results</td>
<td></td>
</tr>
</tbody>
</table>
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 AN8 through AN11.

Figure 2.37  Principles of Operation of A/D Converter Activation by MTU
Software

(1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>main</td>
<td>A/D converter activation by MTU</td>
</tr>
</tbody>
</table>

(2) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>Module standby mode clearing (MTU, A/D)</td>
<td>H'FFFF861E</td>
<td>H'd2ed</td>
</tr>
<tr>
<td>P_MTU0.TCR_0</td>
<td>Selection of TCNT counter clock, and setting of output compare A as TCNT_0 counter clearing source</td>
<td>H'FFFF8260</td>
<td>H'00</td>
</tr>
<tr>
<td>P_MTU0.TIORH_0</td>
<td>Sets TGRA_0 for output compare</td>
<td>H'FFFF8262</td>
<td>H'00</td>
</tr>
<tr>
<td>P_MTU0.TIER_0</td>
<td>Enables A/D conversion start request generation</td>
<td>H'FFFF8264</td>
<td>H'c1</td>
</tr>
<tr>
<td>P_MTU0.TGRA_0</td>
<td>Sets A/D conversion sampling period</td>
<td>H'FFFF8268</td>
<td>H'1000</td>
</tr>
<tr>
<td>P_AD.ADCR_0</td>
<td>Sets MTU conversion start trigger as A/D conversion mode (single-cycle scan mode) activation source</td>
<td>H'FFFF8488</td>
<td>H'87</td>
</tr>
<tr>
<td>P_AD.ADCSR_0</td>
<td>Setting of A/D conversion mode (4ch scan mode), conversion channels (AN8 to AN11), and conversion time, and enabling of A/D conversion end interrupt</td>
<td>H'FFFF8480</td>
<td>H'5f</td>
</tr>
<tr>
<td>P_AD.ADTSR</td>
<td>Enables start of A/D0 module conversion by MTU conversion start trigger signal</td>
<td>H'FFFF87F4</td>
<td>H'02</td>
</tr>
</tbody>
</table>

(3) RAM Used

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

Note: SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

- **MSTCR2 ← H'd2ed**: Clear MTU and A/D converter module standby mode
- **TCR_0 ← H'00**: Set output compare A as ch0 counter clearing source in P_MTU0.TCR_0
- **TIORH_0 ← H'00**: Set TGRA_0 as output compare register in P_MTU0.TIORH_0
- **TIER_0 ← H'c1**: Enable A/D conversion start request generation by TGRA_0 compare match in P_MTU0.TIER_0
- **TGRA_0 ← H'1000**: Set A/D conversion sampling period (compare match period)
- **TCNT_0 ← H'0000**: Clear timer counter
- **ADCR ← H'87**: Set A/D conversion 4ch scan mode and single-cycle scan mode in P_AD.ADCR and P_AD.ADCSR
- **ADCSR ← H'5f**: Enable start of A/D0 module conversion by MTU conversion start trigger signal with P_AD.ADTSR
- **ADTSR ← H'02**: Enable MTU/ch0 count operation with P_MTU34.TSTR
- **TSTR ← H'01**:
Program Listing

/******************************************************************************/
/*                                    INCLUDE FILE                            */
/******************************************************************************/
#include <machine.h>
#include "iodefine_7046.h"
/******************************************************************************/
/*                                     PROTOTYPE                              */
/******************************************************************************/
void main(void);
/******************************************************************************/
/*                                    MAIN PROGRAM                            */
/******************************************************************************/

void main(void)
{
    P_STBY.MSTCR2.WORD = 0xd2ed;    /* Clear Module standby mode */
    P_MTU0.TCR_0.BYTE = 0x00;    /* clock=Pφ/1 */
    P_MTU0.TIORH_0.BYTE = 0x00;
    P_MTU0.TIER_0.BYTE = 0xc1;    /* enable TGIA interrupt */
    P_MTU0.TGRA_0 = 0x1000;
    P_MTU0.TCNT_0 = 0x0000;
    P_AD.ADCR_0.BYTE = 0x87;      /* 1-cycle scan mode */
    P_AD.ADCSR_0.BYTE = 0x5f;    /* 4-channel scan mode */
    P_AD.ADTSR.BYTE = 0x02;      /* A/D start by MTU */
    P_MTU34.TSTR.BYTE = 0x01;    /* Start timer counter */

    set_imask(0x00);
    while(1);
}
2.11 Complementary PWM 3-Phase Output

Specifications

(1) Three-phase complementary PWM waveform output is performed with a non-overlap time (dead time) between positive and negative phases, as shown in figure 2.38.

(2) In this task, duty values are set in a data table, and the duty ratio can be changed by interrupt handling.

(3) In this task, 2.0 ms is set for the period, and 0.1 ms for the dead time.

\[
\text{Duty} = \frac{\text{Pulse high width}}{\text{Pulse period}} \times 100 \, (\%)
\]

![Figure 2.38 Complementary PWM 3-Phase Output Waveforms](image-url)
Functions Used

(1) In this sample task, 3-phase PWM waveforms with a non-overlap time are output from pins PU0A, PU0B, PV0A, PV0B, PW0A, and PW0B, using the complementary PWM waveform output function.

(a) Figure 2.39 shows a block diagram of the complementary PWM waveform output function for the U-phase.

The block diagram of the U-phase complementary PWM waveform output function is described below.

- The timer counter (MMT_TCNT) is a 16-bit counter that counts up/down on an input clock.
- The timer period buffer register (TPBR) is a 16-bit readable/writable register that functions as a buffer register for the timer period data register. A register value of 1/2 PWM carrier period is set. In this task, a setting of 2.0 ms is used.
- The timer period data register (TPDR) is a 16-bit read-only register that is compared with MMT_CNT in the operating mode. When the TPDR value matches the MMT_CNT value, MMT_CNT switches direction from up-counting to down-counting, and the TGFM bit in MMT_TSR is set to 1. The TPDR value is \([TPBR \text{ value} + 2Td]\).
- The timer dead time data register (MMT_TDDR) is a 16-bit readable/writable register that is used to set the non-overlap time (dead time) between positive and negative phases. In this task, a dead time setting of 0.1 ms is used.
- The timer mode register (MMT_TMDR) is an 8-bit readable/writable register that is used for positive-phase/negative-phase output level selection and operating mode setting.
- The timer control register (TCNR) is an 8-bit readable/writable register that selects operation/halting of MMT_CNT, and, when the TGFM bit in MMT_TSR is set to 1, enables/disables interrupt requests.
- Timer buffer register U is a 16-bit readable/writable register that functions as the TGR buffer register. A value written to TBR is transferred to TGR at the timing set by MD1 and MD0 in MMT_TMDR. However, a value written to the TBR free-operation register is transferred to TGR immediately. In this task, the free-operation register is used.
- Timer general register UD (TGRUD) is a 16-bit read-only register to which the TBRU value is transferred. MMT_CNT is compared with TGRUD when counting down.
- Timer general register U (TGRU) is a 16-bit read-only register to which the value of TBRU+Td is transferred. TGRU is constantly compared with MMT_CNT.
- Timer general register UU (TGRUU) is a 16-bit read-only register to which the value of TBRU+2Td is transferred. MMT_CNT is compared with TGRUU when counting up.
- The PWM U-phase (positive-phase) output pin (PU0A) outputs the U-phase positive-phase waveform.
The PWM U-phase (negative-phase) output pin (PU0B) outputs the U-phase negative-phase waveform.

Figure 2.39 MTU U-Phase Block Diagram
Table 2.11  Function Assignments

<table>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUOA</td>
<td>Pin</td>
<td>PWM U-phase output (positive phase)</td>
</tr>
<tr>
<td>PUOB</td>
<td>Pin</td>
<td>PWM U-phase output (negative phase)</td>
</tr>
<tr>
<td>TBRU</td>
<td>Register</td>
<td>TGRUD, TGRU, TGRUU buffer register</td>
</tr>
<tr>
<td>MMT_TDDR</td>
<td>Register</td>
<td>Setting of non-overlap time (Td: dead time) between positive and negative phases</td>
</tr>
<tr>
<td>TPBR</td>
<td>Register</td>
<td>TPDR buffer register. Value of 1/2 PWM carrier period is set</td>
</tr>
<tr>
<td>TPDR</td>
<td>Register</td>
<td>1/2 PWM period + 2Td</td>
</tr>
<tr>
<td>MMT_TSR</td>
<td>Register</td>
<td>Indicates TCNT/TPDR, 2Td compare match occurrence</td>
</tr>
<tr>
<td>TCNR</td>
<td>Register</td>
<td>Interrupt request enabling/disabling control</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Register access enabling/disabling selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Counter operation/halting selection</td>
</tr>
<tr>
<td>MMT_TMDR</td>
<td>Register</td>
<td>Operating mode setting, PWM output level selection</td>
</tr>
</tbody>
</table>
### Operation

(1) Figure 2.40 illustrates the principles of operation of complementary PWM waveform output by SH7046 hardware and software processing.

#### Hardware Processing

- (1) Set $2T_d$ ($T_d$: dead time) in MMT_CNT
- (2) Set output PWM duty in TBR
- (3) Set dead time ($T_d$) in MMT_TDDR
  - Set $1/2$ carrier period in TPBR
  - Set TPBR value + $2T_d$ in TPDR
- (4) Set PWM output level in MMT_TMDR
- (5) Set CST bit to 1 in TCNR to start count operation

#### Software Processing

- (1) Transfer value from buffer register to compare register

#### Initialization

- (1) Clear TGFM bit
- (2) Transfer value from data table to buffer register

---

**Figure 2.40  Principles of Operation of Complementary PWM Waveform Output**
(2) Figure 2.41 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 the operating mode, a buffer register is used when updating compare register data. The update data is fetched from a data table.
• Regarding the timing of data transfer from the data table to the buffer register, transfer is performed by interrupt handling when TGFM is set to 1 by a compare match between MMT_TCNT and TPDR.
• In this sample task TBRU free operation addresses are used, and therefore buffer register data is transferred to the compare register immediately.

Compare Output Waveform (for U-Phase)
Regarding the compare output waveform, MMT_TCNT is compared with TGRU, TGRUU, and TGRUD, and a PWM waveform is generated.

U-phase A
• In period T1 (during TCNT up-counting), MMT_TCNT and TGRUU are compared.
• In period T2 (during TCNT down-counting), MMT_TCNT and TGRU are compared.

U-phase B
• In period T1 (during TCNT up-counting), MMT_TCNT and TGRU are compared.
• In period T2 (during TCNT down-counting), MMT_TCNT and TGRUD are compared.

Period Setting
In case of 20 MHz operation:
Set while 1/2 period (TPBR) = H'0000 to H'FFFF (3.27675 ms) – 4Td.
Figure 2.41 Principles of Operation of PWM Waveform Output Method
(3) Figure 2.42 illustrates the principles of operation. Complementary PWM waveform output is performed by SH7046 hardware and software processing. In this sample task TBRU free operation addresses are used.

Figure 2.42 Principles of Operation of Complementary PWM U-Phase Waveform
Software

(1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>mmt</td>
<td>MMT initialization</td>
</tr>
<tr>
<td>Duty change routine call</td>
<td>UP</td>
<td>Calls U-phase/V-phase/W-phase duty switching routine when TGFM interrupt occurs</td>
</tr>
<tr>
<td>U-phase duty change</td>
<td>set_u</td>
<td>Changes U-phase duty ratio each time TGFM interrupt occurs</td>
</tr>
<tr>
<td>V-phase duty change</td>
<td>set_v</td>
<td>Changes V-phase duty ratio each time TGFM interrupt occurs</td>
</tr>
<tr>
<td>W-phase duty change</td>
<td>set_w</td>
<td>Changes W-phase duty ratio each time TGFM interrupt occurs</td>
</tr>
</tbody>
</table>

(2) Arguments

This sample task does not use any arguments.

(3) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function Description</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>MTU module standby mode clearing</td>
<td>H'FFFF861E</td>
<td>H'b2fd</td>
</tr>
<tr>
<td>P_PORTE.PECRH</td>
<td>Sets port E pins as MMT output pins</td>
<td>H'FFFF83BC</td>
<td>H'0555</td>
</tr>
<tr>
<td>P_PORTE.PEIORH</td>
<td>Sets port E pins as output pins</td>
<td>H'FFFF83B6</td>
<td>H'003f</td>
</tr>
<tr>
<td>P_MMT.MMT_TCNT</td>
<td>2Td (Td: dead time) is set</td>
<td>H'FFFF8A06</td>
<td>H'0fa0</td>
</tr>
<tr>
<td>P_MMT.TBRU_F</td>
<td>Used to set U-phase PWM duty (PWM duty – Td)</td>
<td>H'FFFF8A1C</td>
<td>H'2710</td>
</tr>
<tr>
<td>P_MMT.TBRV_F</td>
<td>Used to set V-phase PWM duty (PWM duty – Td)</td>
<td>H'FFFF8A2C</td>
<td>H'55f0</td>
</tr>
<tr>
<td>P_MMT.TBRW_F</td>
<td>Used to set W-phase PWM duty (PWM duty – Td)</td>
<td>H'FFFF8A3C</td>
<td>H'84b0</td>
</tr>
<tr>
<td>P_MMT.MMT_TDDR</td>
<td>Dead time setting</td>
<td>H'FFFF8A0C</td>
<td>H'07d0</td>
</tr>
<tr>
<td>P_MMT.TPBR</td>
<td>Setting of 1/2 PWM carrier period</td>
<td>H'FFFF8A0A</td>
<td>H'9c40</td>
</tr>
<tr>
<td>P_MMT.MMT_TMDR</td>
<td>Operating mode setting</td>
<td>H'FFFF8A00</td>
<td>H'0e</td>
</tr>
<tr>
<td>P_MMT.TCNR</td>
<td>Enables TGFM interrupts</td>
<td>H'FFFF8A02</td>
<td>H'41</td>
</tr>
</tbody>
</table>
(4) RAM Used

<table>
<thead>
<tr>
<th>Label</th>
<th>Function</th>
<th>Address</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>U-phase duty change counter</td>
<td>H'FFFFD000</td>
<td>set_u</td>
</tr>
<tr>
<td>Y</td>
<td>V-phase duty change counter</td>
<td>H'FFFFD001</td>
<td>set_v</td>
</tr>
<tr>
<td>Y</td>
<td>W-phase duty change counter</td>
<td>H'FFFFD002</td>
<td>set_w</td>
</tr>
</tbody>
</table>

(5) Data Table

In this task, a data table (t_data) is referenced and the PWM duty ratio of each phase is changed by interrupt handling.
Flowcharts

(1) Main routine

```
P_PORTE.PEIORH.WORD = 0x003F;
P_PORTE.PEIORH.WORD = 0x003F;

mmt

X ← 4
Y ← 10
Z ← 16

Assign initial values to software counters

MSTCR2 ← H'2fd

Clear MTT module standby mode

MSTCR2 ← H'2fd

MMT_TCNT ← H'0fa0

Set 2Td as timer count (P_MMT.MMT_CNT)
(P_MMT.MMT_CNT = 0.2 ms)

TBRU_F ← H'2710
TBRV_F ← H'55f0
TBRW_F ← H'84b0

Set output PWM duty initial values - Td in buffer
registers (TBRU_B, TBRV_B, TBRW_B)

MSTCR2 ← H'2fd

MSTCR2 ← H'2fd

MSTCR2 ← H'2fd

MSTCR2 ← H'2fd

MMT_TDDR ← H'07d0

Set dead time (Td = 0.1 ms) in dead time register
(P_MMT.MMT_TDDR)

TPBR ← H'9c40

Set timer period data register (P_MMT.TPDR)
(When 1/2 PWM carrier period is set in timer
period buffer register (P_MMT.TPBR) (TPBR = 2.0 ms),
TPDR = TPBR + 2TD is set)

MMT_TMDR ← H'0e

Set positive-phase/negative-phase output levels
in operating mode in OLSN and OLSP bits of
timer mode register (P_MMT.MMT_TMDR)

IPRI ← H'000f

Set 15 as TGFM interrupt priority level

PECRH ← H'0555
PEIORH ← H'003F

Setting of MMT timer pins PU0A/B, PV0A/B,
PWOA/B

TCNR ← H'41

Set TGIEM bit to 1 in timer control register
(P_MMT.TCNR) to enable TGFM interrupts
Set CST bit to 1 to start count operation
```
(2) Interrupt routine

UP

Yes

MMT_TSR & 0xfe

--- Clear interrupt flag TGFM

set_u()

--- Go to U-phase duty change routine

set_v()

--- Go to V-phase duty change routine

set_w()

--- Go to W-phase duty change routine

RTE

(3) U-phase output

set_u

X++

--- Increment software counter "X"

X <= 19?

Yes

TBRU_F ← t_data[X]

No

X ← 0

--- X less than or equal to 19?

TBRU_F ← t_data[X]

--- Transfer X'th data in data table (t_data) to TBRU_F

RTS

--- Transfer X'th data in data table (t_data) to TBRU_F
(4) V-phase output

set_v

Y++

Yes

Y <= 19?

TBRV_F ← t_data[Y]

NO

Y <= 19?

Y ← 0

TBRV_F ← t_data[Y]

Increment software counter "Y"

Transfer Y'th data in data table (t_data) to TBRV_F

Y++

RTS

(5) W-phase output

set_w

Z++

Yes

Z <= 19?

TBRW_F ← t_data[Z]

NO

Z <= 19?

Z ← 0

TBRW_F ← t_data[Z]

Increment software counter "Z"

Transfer Z'th data in data table (t_data) to TBRW_F

Z++

RTS
Program Listing

/*----------------------------------------------------------------*/
/*                           INCLUDE FILE                         */
/*----------------------------------------------------------------*/
#include <machine.h>
#include "iodefine_7046.h"

/*-----------------------------------------------*/
/*                                      PROTOTYPE    */
/*-----------------------------------------------*/

#include "iodefine_7046.h"

/*----------------------------------------------------------------*/
/*                             PROTOTYPE                          */
/*----------------------------------------------------------------*/

void mmt(void);
void set_u(void);
void set_v(void);
void set_w(void);

void mmt(void)
{
  X=4 ;
  Y=10 ;
  Z=16 ;
  P_STBY.MSTCR2.WORD = 0xb2fd; /* MMT module stop mode clear */
  P_MMT.MMT_TCNT = 0x0FA0;
  P_MMT.TBRU_F = 0x2710;
  P_MMT.TBRV_F = 0x55F0;
  P_MMT.TBRW_F = 0x84B0;
  P_MMT.MMT_TDDR = 0x07D0;
  P_MMT.TBPR = 0x94C0;
  P_MMT.MMT_TMDR.BYTE = 0x0E; /* output level High, mode2 */
  P_INTC.IPRI.WORD = 0x000f; /* set interrupt level=15 */
  P_PORTE.PECRH.WORD = 0x0555; /* PUOA/B,PVOA/B,PWOA/B output */
  P_PORTE.PEIORH.WORD = 0x003f; /* PUOA/B,PVOA/B,PWOA/B output */
  P_MMT.TCNR.BYTE = 0x41; /* timer counter start, TGFM interrupt enable */
  set_imask(0x0); /* set imask level=0 */
  while(1); /* loop */
}

unsigned char X ;
unsigned char Y ;
unsigned char Z ;
void UP()
{
    P_MMT.MMT_TSR.BYTE &= 0xfe; /* TGFM flag clear */
    set_u(); /* change duty Phase U */
    set_v(); /* change duty Phase V */
    set_w(); /* change duty Phase W */
}

void set_u()
{
    X++; /* increment software counter X */
    if(X <= 19){ /* X<=19? */
        P_MMT.TBRU_F = t_data[X]; /* Phase U duty = t_data[X] */
    } else{
        X = 0; /* Clear software counter X */
        P_MMT.TBRU_F = t_data[X]; /* Phase U duty = t_data[X] */
    }
}

void set_v()
{
    Y++; /* increment software counter Y */
    if(Y <= 19){ /* Y<=19? */
        P_MMT.TBRV_F = t_data[Y]; /* Phase V duty = t_data[Y] */
    } else{
        Y = 0; /* Clear software counter Y */
        P_MMT.TBRV_F = t_data[Y]; /* Phase V duty = t_data[Y] */
    }
}

void set_w()
{
    Z++; /* increment software counter Z */
    if(Z <= 19){ /* Z<=19? */
        P_MMT.TBRW_F = t_data[Z]; /* Phase W duty = t_data[Z] */
    } else{
        Z = 0; /* Clear software counter Z */
        P_MMT.TBRW_F = t_data[Z]; /* Phase W duty = t_data[Z] */
    }
}
2.12 Start of A/D Conversion by MMT

<table>
<thead>
<tr>
<th>Start of A/D Conversion by MMT</th>
<th>MCU: SH7046/47</th>
<th>Functions Used: MMT, A/D Converter</th>
</tr>
</thead>
</table>

Specifications

(1) Four channel voltages are input and subjected to A/D conversion as shown in figure 2.43.

(2) Single-cycle scan mode and 4-channel scan mode are used for A/D conversion, with A/D conversion performed consecutively on channels 8 to 11.

(3) A/D converter activation is performed by a compare match between MTT TCNT and TPDR.

![Block Diagram of Voltage Measurement by SH7046](image)
Functions Used

(1) In this sample task, A/D conversion is started by an MMT compare match.

(a) Figure 2.44 shows a block diagram of ch0. In ch0, the A/D converter is activated using the following functions.
   • A function that starts A/D conversion by means of an MMT compare match, without software intervention
   • A function that outputs pulses automatically by hardware without software intervention (output compare)

![Figure 2.44 Block Diagram of SH7046 ch0](image-url)
(b) Figure 2.45 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 a number of channels (ch8 to ch11) (4-channel, single-cycle scan mode)

![Block Diagram of Voltage Measurement by SH7046](image)

Figure 2.45  Block Diagram of Voltage Measurement by SH7046

(2) Table 2.12 shows the function assignments used in this sample task.

<table>
<thead>
<tr>
<th>Pin or Register Name</th>
<th>Function</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN8 to AN11</td>
<td>Pins</td>
<td>Analog measurement pins</td>
</tr>
<tr>
<td>ADDR8 to ADDR11</td>
<td>Registers</td>
<td>Storage of A/D conversion results</td>
</tr>
<tr>
<td>TCNR</td>
<td>Register</td>
<td>Enables A/D conversion start request generation</td>
</tr>
<tr>
<td>TPDR</td>
<td>Register</td>
<td>Sampling period setting</td>
</tr>
<tr>
<td>ADCR</td>
<td>Register</td>
<td>A/D conversion mode and measurement pin setting</td>
</tr>
<tr>
<td>ADCSR</td>
<td>Register</td>
<td>Selection of conversion time and activation source</td>
</tr>
</tbody>
</table>
Operation

(1) Figure 2.46 illustrates the principles of operation of this sample task. As shown in the figure, the A/D converter is activated by a compare match between MMT_TCNT and TPDR, and sequentially measures voltages input to AN8 through AN11.

---

![Figure 2.46 Principles of Operation of A/D Converter Activation by MMT](image-url)
Software

(1) Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Label</th>
<th>Function Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main routine</td>
<td>main</td>
<td>A/D converter activation by MMT</td>
</tr>
</tbody>
</table>

(2) Internal Registers Used

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_STBY.MSTCR2</td>
<td>Module standby mode clearing (MMT, A/D)</td>
<td>H'FFFF861E</td>
<td>H'b2ed</td>
</tr>
<tr>
<td>P_MMT.TCNR</td>
<td>Enables A/D conversion start request generation by compare match between TPBR and MMT_TCNT</td>
<td>H'FFFF8A02</td>
<td>H'c1</td>
</tr>
<tr>
<td>P_MMT.TPBR</td>
<td>Sets A/D conversion sampling period</td>
<td>H'FFFF8A0A</td>
<td>H'61a8</td>
</tr>
<tr>
<td>P_AD.ADCR_0</td>
<td>Sets MMT conversion start trigger as A/D conversion mode (single-cycle scan mode) activation source</td>
<td>H'FFFF8488</td>
<td>H'87</td>
</tr>
<tr>
<td>P_AD.ADCSR_0</td>
<td>Setting of A/D conversion mode (4ch scan mode), conversion channels (AN8 to AN11), and conversion time, and enabling of A/D conversion end interrupt</td>
<td>H'FFFF8480</td>
<td>H'5f</td>
</tr>
<tr>
<td>P_AD.ADTSR</td>
<td>Selects start of A/D conversion by MMT trigger signal</td>
<td>H'FFFF87F4</td>
<td>H'3f</td>
</tr>
</tbody>
</table>

(3) RAM Used

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

Note: SH7046 header file names are used for register label names.
Flowcharts

(1) Main routine

```
main

MSTCR2 ← H'2ed
  Clear MMT and A/D converter module standby mode

ADTSR ← H'3f
  Select start of A/D conversion by MMT trigger signal

ADCR_0 ← H'87
  Set single-cycle scan mode as A/D conversion mode, and MMT A/D conversion start trigger as activation source

ADCSR_0 ← H'5f
  Setting of A/D conversion mode (4ch scan mode), conversion channels (AN8 to AN11), and conversion time

TPBR ← H'4e20
  Set A/D conversion sampling period

TCNR ← H'c1
  Set enabling of A/D conversion start request generation by compare match between TPBR and MMT_TCNT
```
Program Listing

#include <machine.h>
#include "iodefine_7046.h"

void main(void);

void main(void)
{
    P_STBY.MSTCR2.WORD = 0xb2ed;   /* Clear Module standby mode */
    P_AD.ADTSR.BYTE = 0x3f;    /* A/D start by MMT */
    P_AD.ADCR_0.BYTE = 0x87;   /* 1-cycle scan mode */
    P_MMT.TCNR.BYTE = 0x5f;    /* 4ch scan mode */
    P_AD.ADCSR_0.BYTE = 0x3f;
    P_MMT.TPBR = 0x4e20;       /* Sampling period=1ms */
    P_MMT.TCNR.BYTE = 0x00;
    set_imask(0x0);
    while(1);
}

/*                                  INCLUDE FILE                         */
/******************************************************************************/
/*                                    PROTOTYPE                               */
/******************************************************************************/
/*                                   MAIN PROGRAM                             */
/******************************************************************************/
Section 3  Appendix

Program Listing

/* 7046/47 Include File */
//----------------------------------------------------------------------------
struct st_sci {            /* struct SCI   */
    union {                /* SMR          */
        unsigned char BYTE; /* Byte Access */
        struct {            /* Bit Access */
            unsigned char CA:1; /* C/A */
            unsigned char CHR:1; /* CHR */
            unsigned char PE:1; /* PE */
            unsigned char OE:1; /* O/E */
            unsigned char STOP:1; /* STOP */
            unsigned char MP:1; /* MP */
            unsigned char CKS:2; /* CKS */
        } BIT;
    } SMR;
    unsigned char BRR;      /* BRR */
    union {                /* SCR          */
        unsigned char BYTE; /* Bit Access */
        struct {            /* Bit Access */
            unsigned char TIE:1; /* TIE */
            unsigned char RIE:1; /* RIE */
            unsigned char TE:1; /* TE */
            unsigned char RE:1; /* RE */
            unsigned char MPIE:1; /* MPIE */
            unsigned char TEIE:1; /* TEIE */
            unsigned char CKE:2; /* CKE */
        } BIT;
    } SCR;
    unsigned char TDR;      /* TDR */
    union {                /* SSR          */
        unsigned char BYTE; /* Bit Access */
        struct {            /* Bit Access */
            unsigned char TDRE:1; /* TDRE */
            unsigned char RDRF:1; /* RDRF */
            unsigned char ORER:1; /* ORER */
            unsigned char FER:1; /* FER */
            unsigned char PER:1; /* PER */
            unsigned char TEND:1; /* TEND */
            unsigned char MPB:1; /* MPB */
            unsigned char MPBT:1; /* MPBT */
        } BIT;
    } SSR;
} st_sci;
union {
    unsigned char BYTE;
    struct {
        unsigned char :4;
        unsigned char DIR:1;
        unsigned char :3;
    } BIT;
} SDCR;
};

struct st_mtu34 {
    union {
        unsigned char BYTE;
        struct {
            unsigned char CCLR:3;
            unsigned char CKEG:2;
            unsigned char TPSC:3;
        } BIT;
    } TCR_3;
    union {
        unsigned char BYTE;
        struct {
            unsigned char CCLR:3;
            unsigned char CKEG:2;
            unsigned char TPSC:3;
        } BIT;
    } TCR_4;
    union {
        unsigned char BYTE;
        struct {
            unsigned char :2;
            unsigned char BFB:1;
            unsigned char BFA:1;
            unsigned char MD:4;
        } BIT;
    } TMDR_3;
    union {
        unsigned char BYTE;
        struct {
            unsigned char :2;
            unsigned char BFB:1;
            unsigned char BFA:1;
            unsigned char MD:4;
        } BIT;
    } TMDR_4;
    union {
        unsigned char BYTE;
        struct {
            unsigned char IOB:4;
            unsigned char IOA:4;
        } BIT;
    } TIORH_3;
};

union { /* TIORH_3 */
    unsigned char BYTE; /* Byte Access */
    struct {
        unsigned char IOD:4; /* IOD */
        unsigned char IOC:4; /* IOC */
    } BIT; /* */
} TIORL_3; /* */

union { /* TIORH_4 */
    unsigned char BYTE; /* Byte Access */
    struct {
        unsigned char IOB:4; /* IOB */
        unsigned char IOA:4; /* IOA */
    } BIT; /* */
} TIORL_4; /* */

union { /* TIER_3 */
    unsigned char BYTE; /* Byte Access */
    struct {
        unsigned char TTGE:1; /* TTGE */
        unsigned char :2; /* */
        unsigned char TCIEV:1; /* TCIEV */
        unsigned char TGIEID:1; /* TGIEID */
        unsigned char TGIEC:1; /* TGIEC */
        unsigned char TGIEEB:1; /* TGIEEB */
        unsigned char TGIEEA:1; /* TGIEEA */
    } BIT; /* */
} TIER_3; /* */

union { /* TIER_4 */
    unsigned char BYTE; /* Byte Access */
    struct {
        unsigned char TTGE:1; /* TTGE */
        unsigned char :2; /* */
        unsigned char TCIEV:1; /* TCIEV */
        unsigned char TGIEID:1; /* TGIEID */
        unsigned char TGIEC:1; /* TGIEC */
        unsigned char TGIEEB:1; /* TGIEEB */
        unsigned char TGIEEA:1; /* TGIEEA */
    } BIT; /* */
} TIER_4; /* */

union { /* TOER */
    unsigned char BYTE; /* Byte Access */
    struct {
        unsigned char :2; /* */
    }
}
unsigned char OE4D:1;              /*    OE4D      */
unsigned char OE4C:1;              /*    OE4C      */
unsigned char OE3D:1;              /*    OE3D      */
unsigned char OE4B:1;              /*    OE4B      */
unsigned char OE4A:1;              /*    OE4A      */
unsigned char OE3B:1;              /*    OE3B      */
} BIT;
/* */
} TOER;
/* */
union {
  /* TOCR         */
    unsigned char BYTE;                       /*  Byte Access */
    struct {                                  /*  Bit Access  */
      unsigned char :1;                  /*              */
      unsigned char PSYE:1;              /*    PSYE      */
      unsigned char :4;                  /*              */
      unsigned char OLSN:1;              /*    OLSN      */
      unsigned char OLSP:1;              /*    OLSP      */
    } BIT;                             /*              */
} TOCR;
/* */
unsigned char wk0[1];                           /* */
union {
  /* TGCR         */
    unsigned char BYTE;                       /*  Byte Access */
    struct {                                  /*  Bit Access  */
      unsigned char :1;                  /*              */
      unsigned char BDC:1;               /*    BDC       */
      unsigned char N:1;                 /*    N         */
      unsigned char P:1;                 /*    P         */
      unsigned char FB:1;                /*    FB        */
      unsigned char WF:1;                /*    WF        */
      unsigned char VF:1;                /*    VF        */
      unsigned char UF:1;                /*    UF        */
    } BIT;                             /*              */
} TGCR;
/* */
unsigned char wk1[2];                           /* */
unsigned short TCNT_3;                          /* TCNT_3   */
unsigned short TCNT_4;                          /* TCNT_4   */
unsigned short TCDR;                            /* TCDR     */
unsigned short TDDR;                            /* TDDR     */
unsigned short TGRA_3;                          /* TGRA_3   */
unsigned short TGRB_3;                          /* TGRB_3   */
unsigned short TGRA_4;                          /* TGRA_4   */
unsigned short TGRB_4;                          /* TGRB_4   */
unsigned short TCNTS;                           /* TCNTS    */
unsigned short TCBR;                            /* TCBR     */
unsigned short TGRC_3;                          /* TGRC_3   */
unsigned short TGRD_3;                          /* TGRD_3   */
unsigned short TGRC_4;                          /* TGRC_4   */
unsigned short TGRD_4;                          /* TGRD_4   */
union {
    unsigned char BYTE;                       /*  Byte Access */
    struct {                                  /*  Bit Access  */
      unsigned char TDFD:1;                /*    TDFD      */
    } TSR_3   */
}
unsigned char :2;                  /*              */
unsigned char TCFV:1;              /*    TCFV      */
unsigned char TGFD:1;              /*    TGFD      */
unsigned char TGFC:1;              /*    TGFC      */
unsigned char TGFB:1;              /*    TGFB      */
unsigned char TGFA:1;              /*    TGFA      */
} BIT;                             /*              */
} TSR_3;
union {                                         /* TSR_4        */
    unsigned char BYTE;                       /*  Byte Access */
    struct {                                  /*  Bit Access  */
        unsigned char TDFD:1;              /*    TDFD      */
        unsigned char :2;                  /*              */
        unsigned char TCFV:1;              /*    TCFV      */
        unsigned char TGFD:1;              /*    TGFD      */
        unsigned char TGFC:1;              /*    TGFC      */
        unsigned char TGFB:1;              /*    TGFB      */
        unsigned char TGFA:1;              /*    TGFA      */
    } BIT;                             /*              */
} TSR_4;                                  /*              */
unsigned char wk2[18];                          /*              */
union {                                         /* TSTR         */
    unsigned char BYTE;                       /*  Byte Access */
    struct {                                  /*  Bit Access  */
        unsigned char CST4:1;              /*    CST4      */
        unsigned char CST3:1;              /*    CST3      */
        unsigned char :3;                  /*              */
        unsigned char CST:3;               /*    CST       */
    } BIT;                             /*              */
} TSTR;                                   /*              */
union {                                         /* TSYR         */
    unsigned char BYTE;                       /*  Byte Access */
    struct {                                  /*  Bit Access  */
        unsigned char SYNC4:1;             /*    SYNC4     */
        unsigned char SYNC3:1;             /*    SYNC3     */
        unsigned char :3;                  /*              */
        unsigned char SYNC2:1;             /*    SYNC2     */
        unsigned char SYNC1:1;             /*    SYNC1     */
        unsigned char SYNC0:1;             /*    SYNC0     */
    } BIT;                             /*              */
} TSYR;                                   /*              */
};                                                     /*              */
struct st_mtu0 {                                       /* struct MTU0  */
    union {                                         /* TCR_0        */
        unsigned char BYTE;                       /*  Byte Access */
        struct {                                  /*  Bit Access  */
            unsigned char CCLR:3;              /*    CCLR      */
            unsigned char CKEG:2;              /*    CKEG      */
            unsigned char TPSC:3;              /*    TPSC      */
        } BIT;                             /*              */
    } TCR_0;                                  /*              */
};
union {
    unsigned char BYTE;
    struct {
        unsigned char :2;
        unsigned char BFB:1;
        unsigned char BFA:1;
        unsigned char MD:4;
    } BIT;
} TMDR_0;
union {
    unsigned char BYTE;
    struct {
        unsigned char IOB:4;
        unsigned char IOA:4;
    } BIT;
} TIORH_0;
union {
    unsigned char BYTE;
    struct {
        unsigned char IOD:4;
        unsigned char IOC:4;
    } BIT;
} TIORL_0;
union {
    unsigned char BYTE;
    struct {
        unsigned char TTGE:1;
        unsigned char :2;
        unsigned char TCIEV:1;
        unsigned char TGIED:1;
        unsigned char TGIEC:1;
        unsigned char TGIEB:1;
        unsigned char TGIEA:1;
    } BIT;
} TIER_0;
union {
    unsigned char BYTE;
    struct {
        unsigned char :3;
        unsigned char TCFV:1;
        unsigned char TGFD:1;
        unsigned char TGFC:1;
        unsigned char TGFB:1;
        unsigned char TGFA:1;
    } BIT;
} TSR_0;
unsigned short TCNT_0; /* TCNT_0 */
unsigned short TGRA_0; /* TGRA_0 */
unsigned short TGRB_0; /* TGRB_0 */
unsigned short TGRC_0; /* TGRC_0 */
unsigned short TGRD_0; /* TGRD_0 */
struct st_mtu1 {
    union {
        unsigned char BYTE;
        struct {
            unsigned char :1;
            unsigned char CCLR:2;
            unsigned char CKEG:2;
            unsigned char TPSC:3;
        } BIT;
    } TCR_1;
    union {
        unsigned char BYTE;
        struct {
            unsigned char :4;
            unsigned char MD:4;
        } BIT;
    } TMDR_1;
    union {
        unsigned char BYTE;
        struct {
            unsigned char IOB:4;
            unsigned char IOA:4;
        } BIT;
    } TIOR_1;
    unsigned char wk0[1];
    union {
        unsigned char BYTE;
        struct {
            unsigned char TTGE:1;
            unsigned char :1;
            unsigned char TCIEU:1;
            unsigned char TCIEV:1;
            unsigned char :2;
            unsigned char TGIEB:1;
            unsigned char TGIEA:1;
        } BIT;
    } TIER_1;
    union {
        unsigned char BYTE;
        struct {
            unsigned char TCFD:1;
            unsigned char :1;
            unsigned char TCFU:1;
            unsigned char TCFV:1;
            unsigned char :2;
            unsigned char TGFB:1;
            unsigned char TGFA:1;
        } BIT;
    } TSR_1;
    unsigned short TCNT_1;
}; /* struct MTU1 */ /* TCR_1 */ /* Byte Access */ /* Bit Access */ /* TMDR_1 */ /* TIER_1 */ /* TSR_1 */ /* TCNT_1 */
unsigned short TGRA_1;                          /* TGRA_1       */
unsigned short TGRB_1;                          /* TGRB_1       */
};                                                     /*              */
struct st_mtu2 {                                       /* struct MTU2  */
union {                                         /* TCR_2        */
  unsigned char BYTE;                       /*  Byte Access */
  struct {                                  /*  Bit Access  */
    unsigned char :1;                  /*              */
    unsigned char CCLR:2;              /*    CCLR      */
    unsigned char CKEG:2;              /*    CKEG      */
    unsigned char TPSC:3;              /*    TPSC      */
  } BIT;                             /*              */
} TCR_2;                                  /*              */
union {                                         /* TMDR_2       */
  unsigned char BYTE;                       /*  Byte Access */
  struct {                                  /*  Bit Access  */
    unsigned char :4;                  /*              */
    unsigned char MD:4;                /*    MD        */
  } BIT;                             /*              */
} TMDR_2;                                 /*              */
union {                                         /* TIOR_2       */
  unsigned char BYTE;                       /*  Byte Access */
  struct {                                  /*  Bit Access  */
    unsigned char IOB:4;               /*    IOB       */
    unsigned char IOA:4;               /*    IOA       */
  } BIT;                             /*              */
} TIOR_2;                                 /*              */
unsigned char wk0[1];                           /*              */
union {                                         /* TIER_2       */
  unsigned char BYTE;                       /*  Byte Access */
  struct {                                  /*  Bit Access  */
    unsigned char TTGE:1;              /*    TTGE      */
    unsigned char :1;                  /*              */
    unsigned char TCIEU:1;             /*    TCIEU     */
    unsigned char TCIEV:1;             /*    TCIEV     */
    unsigned char :2;                  /*              */
    unsigned char TGIEB:1;             /*    TGIEB     */
    unsigned char TGIEA:1;             /*    TGIEA     */
  } BIT;                             /*              */
} TIER_2;                                 /*              */
union {                                         /* TSR_2        */
  unsigned char BYTE;                       /*  Byte Access */
  struct {                                  /*  Bit Access  */
    unsigned char TCFD:1;              /*    TCFD      */
    unsigned char :1;                  /*              */
    unsigned char TCFU:1;              /*    TCFU      */
    unsigned char TCFV:1;              /*    TCFV      */
    unsigned char :2;                  /*              */
    unsigned char TGFB:1;              /*    TGFB      */
    unsigned char TGFA:1;              /*    TGFA      */
  } BIT;                             /*              */
} TSR_2;                                 /*              */
struct st_intc {
    union {
        unsigned short WORD;    /* Word Access */
        struct {
            unsigned short IRQ0:4;    /* IRQ0 */
            unsigned short IRQ1:4;    /* IRQ1 */
            unsigned short IRQ2:4;    /* IRQ2 */
            unsigned short IRQ3:4;    /* IRQ3 */
        } BIT;
    } IPRA;

    unsigned char wk0[4];    /* */
    union {
        unsigned short WORD;    /* Word Access */
        struct {
            unsigned short TGI_0:4;    /* TGI_0 */
            unsigned short TCI_0:4;    /* TCI_0 */
            unsigned short TGI_1:4;    /* TGI_1 */
            unsigned short TCI_1:4;    /* TCI_1 */
        } BIT;
    } IPRD;

    union {
        unsigned short WORD;    /* Word Access */
        struct {
            unsigned short TGI_2:4;    /* TGI_2 */
            unsigned short TCI_2:4;    /* TCI_2 */
            unsigned short TGI_3:4;    /* TGI_3 */
            unsigned short TCI_3:4;    /* TCI_3 */
        } BIT;
    } IPRE;

    union {
        unsigned short WORD;    /* Word Access */
        struct {
            unsigned short TGI_4:4;    /* TGI_4 */
            unsigned short TCI_4:4;    /* TCI_4 */
            unsigned short :8;        /* */
        } BIT;
    } IPRF;

    union {
        unsigned short WORD;    /* Word Access */
        struct {
            unsigned short AD01:4;    /* A/D0,1 */
            unsigned short DTC:4;     /* DTC */
            unsigned short CMT0:4;    /* CMT0 */
            unsigned short CMT1:4;    /* CMT1 */
        } BIT;
    } IPRG;
union {
    unsigned short WORD;
    struct {
        unsigned short WDT:4; /* WDT */
        unsigned short IOMTU:4; /* I/O(MTU) */
        unsigned short :8;    /* */
    } BIT;
} IPRH;
union {
    unsigned short WORD;
    struct {
        unsigned short NMIL:1;  /* NMIL */
        unsigned short :6;     /* */
        unsigned short NMIE:1; /* NMIE */
        unsigned short IRQ0S:1; /* IRQ0S */
        unsigned short IRQ1S:1; /* IRQ1S */
        unsigned short IRQ2S:1; /* IRQ2S */
        unsigned short IRQ3S:1; /* IRQ3S */
        unsigned short :4;     /* */
    } BIT;
} ICR1;
union {
    unsigned short WORD;
    struct {
        unsigned short :8;    /* */
        unsigned short IRQ0F:1; /* IRQ0F */
        unsigned short IRQ1F:1; /* IRQ1F */
        unsigned short IRQ2F:1; /* IRQ2F */
        unsigned short IRQ3F:1; /* IRQ3F */
        unsigned short :4;     /* */
    } BIT;
} ISR;
union {
    unsigned short WORD;
    struct {
        unsigned short SCI2:4;  /* SCI2 */
        unsigned short SCI3:4;  /* SCI3 */
        unsigned short SCI4:4;  /* SCI4 */
        unsigned short MMT:4;   /* MMT */
    } BIT;
} IPRI;
union {
    unsigned short WORD;
    struct {
        unsigned short AD2:4;  /* A/D2 */
        unsigned short :12;   /* */
    } BIT;
} IPRJ;
union {
    unsigned short WORD;
    struct {
        unsigned short :8;    /* */
    } BIT;
} IPRK;

unsigned short IOMMT:4;            /*    I/O(MMT)  */
unsigned short :4;                 /*              */
unsigned short HCAN2:4;            /*    HCAN1     */
unsigned short :4;                 /*              */
} BIT;
} IPRK;
unsigned char wk1[4];             /*              */
union {
  /* ICR2         */
    unsigned short WORD;
    struct {
      unsigned short IRQ0ES:2;   /*    IRQ0ES    */
      unsigned short IRQ1ES:2;   /*    IRQ1ES    */
      unsigned short IRQ2ES:2;   /*    IRQ2ES    */
      unsigned short IRQ3ES:2;   /*    IRQ3ES    */
      unsigned short :8;        /*              */
    } BIT;
  } ICR2;
}
struct st_porta {
  /* struct PORTA */
    union {
      unsigned short WORD;
      struct {
        unsigned short PA15DR:1;   /*    PA15DR    */
        unsigned short PA14DR:1;   /*    PA14DR    */
        unsigned short PA13DR:1;   /*    PA13DR    */
        unsigned short PA12DR:1;   /*    PA12DR    */
        unsigned short PA11DR:1;   /*    PA11DR    */
        unsigned short PA10DR:1;   /*    PA10DR    */
        unsigned short PA9DR:1;    /*    PA9DR     */
        unsigned short PA8DR:1;    /*    PA8DR     */
        unsigned short PA7DR:1;    /*    PA7DR     */
        unsigned short PA6DR:1;    /*    PA6DR     */
        unsigned short PA5DR:1;    /*    PA5DR     */
        unsigned short PA4DR:1;    /*    PA4DR     */
        unsigned short PA3DR:1;    /*    PA3DR     */
        unsigned short PA2DR:1;    /*    PA2DR     */
        unsigned short PA1DR:1;    /*    PA1DR     */
        unsigned short PA0DR:1;    /*    PA0DR     */
      } BIT;
    } PADRL;
}
unsigned char wk0[2];             /*              */
union {
  /* PAIORL       */
    unsigned short WORD;
    struct {
      unsigned short PA15IOR:1;  /*    PA15IOR   */
      unsigned short PA14IOR:1;  /*    PA14IOR   */
      unsigned short PA13IOR:1;  /*    PA13IOR   */
      unsigned short PA12IOR:1;  /*    PA12IOR   */
      unsigned short PA11IOR:1;  /*    PA11IOR   */
      unsigned short PA10IOR:1;  /*    PA10IOR   */
      unsigned short PA9IOR:1;   /*    PA9IOR    */
  } BIT;
} PADRL;
unsigned short PA8IOR:1;           /*    PA8IOR    */
unsigned short PA7IOR:1;           /*    PA7IOR    */
unsigned short PA6IOR:1;           /*    PA6IOR    */
unsigned short PA5IOR:1;           /*    PA5IOR    */
unsigned short PA4IOR:1;           /*    PA4IOR    */
unsigned short PA3IOR:1;           /*    PA3IOR    */
unsigned short PA2IOR:1;           /*    PA2IOR    */
unsigned short PA1IOR:1;           /*    PA1IOR    */
unsigned short PA0IOR:1;           /*    PA0IOR    */
} BIT;                             /*              */
}

union {
    unsigned short WORD;                      /*  Word Access */
    struct {
        unsigned short PA15MD2:1;          /*    PA15MD2   */
        unsigned short PA14MD2:1;          /*    PA14MD2   */
        unsigned short PA13MD2:1;          /*    PA13MD2   */
        unsigned short PA12MD2:1;          /*    PA12MD2   */
        unsigned short PA11MD2:1;          /*    PA11MD2   */
        unsigned short PA10MD2:1;          /*    PA10MD2   */
        unsigned short PA9MD2:1;           /*    PA9MD2    */
        unsigned short PA8MD2:1;           /*    PA8MD2    */
        unsigned short PA7MD2:1;           /*    PA7MD2    */
        unsigned short PA6MD2:1;           /*    PA6MD2    */
        unsigned short PA5MD2:1;           /*    PA5MD2    */
        unsigned short PA4MD2:1;           /*    PA4MD2    */
        unsigned short PA3MD2:1;           /*    PA3MD2    */
        unsigned short PA2MD2:1;           /*    PA2MD2    */
        unsigned short PA1MD2:1;           /*    PA1MD2    */
        unsigned short PA0MD2:1;           /*    PA0MD2    */
    } BIT;                             /*              */
}

union {
    unsigned short WORD;                      /*  Word Access */
    struct {
        unsigned short PA15MD:2;           /*    PA15MD    */
        unsigned short PA14MD:2;           /*    PA14MD    */
        unsigned short PA13MD:2;           /*    PA13MD    */
        unsigned short PA12MD:2;           /*    PA12MD    */
        unsigned short PA11MD:2;           /*    PA11MD    */
        unsigned short PA10MD:2;           /*    PA10MD    */
        unsigned short PA9MD:2;            /*    PA9MD     */
        unsigned short PA8MD:2;            /*    PA8MD     */
    } BIT;                             /*              */
}

union {
    unsigned short WORD;                      /*  Word Access */
    struct {
        unsigned short PA7MD:2;           /*    PA7MD     */
        unsigned short PA6MD:2;           /*    PA6MD     */
    } BIT;                             /*              */
}
unsigned short PA5MD:2;            /*    PA5MD     */
unsigned short PA4MD:2;            /*    PA4MD     */
unsigned short PA3MD:2;            /*    PA3MD     */
unsigned short PA2MD:2;            /*    PA2MD     */
unsigned short PA1MD:2;            /*    PA1MD     */
unsigned short PA0MD:2;            /*    PA0MD     */
} BIT;                             /*              */
} PACRL2;                                 /*              */
}

};                                                     /*              */
}

}

};                                                     /*              */

}

/* struct PORTB */
union {
unsigned short WORD;                      /*  Word Access */
struct {
unsigned short :10;                /*              */
unsigned short PB5DR:1;            /*    PB5DR     */
unsigned short PB4DR:1;            /*    PB4DR     */
unsigned short PB3DR:1;            /*    PB3DR     */
unsigned short PB2DR:1;            /*    PB2DR     */
unsigned short PB1DR:1;            /*    PB1DR     */
unsigned short PB0DR:1;            /*    PB0DR     */
} BIT;                             /*              */
*/
*/
} PBDR;
*/
*/
unsigned char wk0[2];                           /*              */
union {
/* PBIOR */
unsigned short WORD;                      /*  Word Access */
struct {
unsigned short :10;                /*              */
unsigned short PB5IOR:1;           /*    PB5IOR    */
unsigned short PB4IOR:1;           /*    PB4IOR    */
unsigned short PB3IOR:1;           /*    PB3IOR    */
unsigned short PB2IOR:1;           /*    PB2IOR    */
unsigned short PB1IOR:1;           /*    PB1IOR    */
unsigned short PB0IOR:1;           /*    PB0IOR    */
} BIT;                             /*              */
*/
*/
} PBIOR;
*/
*/
unsigned char wk1[2];                           /*              */
union {
/* PBCR1 */
unsigned short WORD;                      /*  Word Access */
struct {
unsigned short :2;                 /*              */
unsigned short PB5MD2:1;           /*    PB5MD2    */
unsigned short PB4MD2:1;           /*    PB4MD2    */
unsigned short PB3MD2:1;           /*    PB3MD2    */
unsigned short PB2MD2:1;           /*    PB2MD2    */
unsigned short PB1MD2:1;           /*    PB1MD2    */
unsigned short :9;                  /*              */
} BIT;                             /*              */
*/
*/
} PBCR1;
*/
*/
union {
/* PBCR2 */
unsigned short WORD;                      /*  Word Access */
struct {
*/
unsigned short :4;                 /*              */
unsigned short PB5MD:2;            /*    PB5MD     */
unsigned short PB4MD:2;             /*    PB4MD     */
unsigned short PB3MD:2;             /*    PB3MD     */
unsigned short PB2MD:2;             /*    PB2MD     */
unsigned short PB1MD:2;             /*    PB1MD     */
unsigned short PB0MD:2;             /*    PB0MD     */
} BIT;                             /*              */
} PBCR2;

struct st_portd {
    unsigned short WORD;
    union {
        struct {
            unsigned short :7;              /*              */
            unsigned short PD8DR:1;       /*    PD8DR     */
            unsigned short PD7DR:1;       /*    PD7DR     */
            unsigned short PD6DR:1;       /*    PD6DR     */
            unsigned short PD5DR:1;       /*    PD5DR     */
            unsigned short PD4DR:1;       /*    PD4DR     */
            unsigned short PD3DR:1;       /*    PD3DR     */
            unsigned short PD2DR:1;       /*    PD2DR     */
            unsigned short PD1DR:1;       /*    PD1DR     */
            unsigned short PD0DR:1;       /*    PD0DR     */
        } BIT;                             /*              */
    } PDDRL;                                  /*              */
    unsigned char wk0[2];                           /*              */
    union {
        unsigned short WORD;      /*    PD8IOR     */
        union {
            struct {
                unsigned short :7;              /*              */
                unsigned short PD8IOR:1;       /*    PD8IOR     */
                unsigned short PD7IOR:1;       /*    PD7IOR     */
                unsigned short PD6IOR:1;       /*    PD6IOR     */
                unsigned short PD5IOR:1;       /*    PD5IOR     */
                unsigned short PD4IOR:1;       /*    PD4IOR     */
                unsigned short PD3IOR:1;       /*    PD3IOR     */
                unsigned short PD2IOR:1;       /*    PD2IOR     */
                unsigned short PD1IOR:1;       /*    PD1IOR     */
                unsigned short PD0IOR:1;       /*    PD0IOR     */
            } BIT;                             /*              */
        } PDIORL;                                 /*              */
    } PDIORL;
    unsigned char wk1[4];                           /*              */
    union {
        unsigned short WORD;      /*    PD8MD0     */
        union {
            struct {
                unsigned short :7;              /*              */
                unsigned short PD8MD0:1;       /*    PD8MD0     */
                unsigned short PD7MD0:1;       /*    PD7MD0     */
                unsigned short PD6MD0:1;       /*    PD6MD0     */
                unsigned short PD5MD0:1;       /*    PD5MD0     */
            } BIT;                             /*              */
        } PDCRL1;                                 /*              */
    } PDCRL1;
}
unsigned short PD4MD0:1; /* PD4MD0 */
unsigned short PD3MD0:1; /* PD3MD0 */
unsigned short PD2MD0:1; /* PD2MD0 */
unsigned short PD1MD0:1; /* PD1MD0 */
unsigned short PD0MD0:1; /* PD0MD0 */
} BIT; /* */

} PDCRL1;
/* */
union {
  unsigned short WORD; /* Word Access */
  struct {
    unsigned short :7; /* */
    unsigned short PD8MD1:1; /* PD8MD1 */
    unsigned short PD7MD1:1; /* PD7MD1 */
    unsigned short PD6MD1:1; /* PD6MD1 */
    unsigned short PD5MD1:1; /* PD5MD1 */
    unsigned short PD4MD1:1; /* PD4MD1 */
    unsigned short PD3MD1:1; /* PD3MD1 */
    unsigned short PD2MD1:1; /* PD2MD1 */
    unsigned short PD1MD1:1; /* PD1MD1 */
    unsigned short PD0MD1:1; /* PD0MD1 */
  } BIT; /* */
} PDCRL2;
/* */
struct st_porte {
  /* struct PORTE */
  union {
    unsigned short WORD; /* Word Access */
    struct {
      unsigned short PE15DR:1; /* PE15DR */
      unsigned short PE14DR:1; /* PE14DR */
      unsigned short PE13DR:1; /* PE13DR */
      unsigned short PE12DR:1; /* PE12DR */
      unsigned short PE11DR:1; /* PE11DR */
      unsigned short PE10DR:1; /* PE10DR */
      unsigned short PE9DR:1; /* PE9DR */
      unsigned short PE8DR:1; /* PE8DR */
      unsigned short PE7DR:1; /* PE7DR */
      unsigned short PE6DR:1; /* PE6DR */
      unsigned short PE5DR:1; /* PE5DR */
      unsigned short PE4DR:1; /* PE4DR */
      unsigned short PE3DR:1; /* PE3DR */
      unsigned short PE2DR:1; /* PE2DR */
      unsigned short PE1DR:1; /* PE1DR */
      unsigned short PE0DR:1; /* PE0DR */
    } BIT; /* */
  } PEDRL;
/* */
unsigned char wk0[2]; /* */
union {
  unsigned short WORD; /* Word Access */
  struct {
    unsigned short PE15IOR:1; /* PE15IOR */
    unsigned short PE14IOR:1; /* PE14IOR */
  } PEIORL; /* */
} PEDRL;
/* */
unsigned short PE13IOR:1;          /*    PE13IOR   */
unsigned short PE12IOR:1;          /*    PE12IOR   */
unsigned short PE11IOR:1;          /*    PE11IOR   */
unsigned short PE10IOR:1;          /*    PE10IOR   */
unsigned short PE9IOR:1;           /*    PE9IOR    */
unsigned short PE8IOR:1;           /*    PE8IOR    */
unsigned short PE7IOR:1;           /*    PE7IOR    */
unsigned short PE6IOR:1;           /*    PE6IOR    */
unsigned short PE5IOR:1;           /*    PE5IOR    */
unsigned short PE4IOR:1;           /*    PE4IOR    */
unsigned short PE3IOR:1;           /*    PE3IOR    */
unsigned short PE2IOR:1;           /*    PE2IOR    */
unsigned short PE1IOR:1;           /*    PE1IOR    */
unsigned short PE0IOR:1;           /*    PE0IOR    */
} BIT;                             /*              */
} PEIORL;
union {
    unsigned short WORD;                      /*  Word Access */
    struct {
        unsigned short :10;                /*              */
        unsigned short PE21IOR:1;          /*    PE21IOR   */
        unsigned short PE20IOR:1;          /*    PE20IOR   */
        unsigned short PE19IOR:1;          /*    PE19IOR   */
        unsigned short PE18IOR:1;          /*    PE18IOR   */
        unsigned short PE17IOR:1;          /*    PE17IOR   */
        unsigned short PE16IOR:1;          /*    PE16IOR   */
    } BIT;                             /*              */
} PEIORH;                                 /*              */
union {
    unsigned short WORD;                      /*  Word Access */
    struct {
        unsigned short PE15MD:2;           /*    PE15MD    */
        unsigned short PE14MD:2;           /*    PE14MD    */
        unsigned short PE13MD:2;           /*    PE13MD    */
        unsigned short PE12MD:2;           /*    PE12MD    */
        unsigned short PE11MD:2;           /*    PE11MD    */
        unsigned short PE10MD:2;           /*    PE10MD    */
        unsigned short PE9MD:2;            /*    PE9MD     */
        unsigned short PE8MD:2;            /*    PE8MD     */
    } BIT;                             /*              */
} PECRL1;                                 /*              */
union {
    unsigned short WORD;                      /*  Word Access */
    struct {
        unsigned short PE7MD:2;            /*    PE7MD     */
        unsigned short PE6MD:2;            /*    PE6MD     */
        unsigned short PE5MD:2;            /*    PE5MD     */
        unsigned short PE4MD:2;            /*    PE4MD     */
        unsigned short PE3MD:2;            /*    PE3MD     */
        unsigned short PE2MD:2;            /*    PE2MD     */
        unsigned short PE1MD:2;            /*    PE1MD     */
    } BIT;                             /*              */
} PECRL2;                                 /*              */
unsigned short PE0MD:2; /* PE0MD */
} BIT;
/* */
} PECRL2;
/* */
union {
    /* PECRH */
    unsigned short WORD;
    /* Word Access */
} BIT;
/* */
} PECRH;
/* */
union {
    /* PEDRH */
    unsigned short WORD;
    /* Word Access */
} BIT;
/* */
} PEDRH;
/* */
};
/* */
}
/* */
};
/* */
struct st_portf {
    /* struct PORTF */
    unsigned short WORD;
    /* Word Access */
} PFDR;
/* */
union {
    /* PFDR */
    unsigned short WORD;
    /* Word Access */
} BIT;
/* */
} PFDR;
/* */
struct st_portf {
    /* struct PORTF */
    unsigned short WORD;
    /* Word Access */
} OUT;
/* */
union {
    /* OUT */
    unsigned short WORD;
    /* Word Access */
} BIT;
/* */
} OUT;
/* */
};
/* */
struct st_mtu {
    union {
        unsigned short WORD;
        struct {
            unsigned short POE3F:1; /* POE3F */
            unsigned short POE2F:1; /* POE2F */
            unsigned short POE1F:1; /* POE1F */
            unsigned short POE0F:1; /* POE0F */
            unsigned short :3; /* */
            unsigned short PIE:1; /* PIE */
            unsigned short POE3M:2; /* POE3M */
            unsigned short POE2M:2; /* POE2M */
            unsigned short POE1M:2; /* POE1M */
            unsigned short POE0M:2; /* POE0M */
        } BIT;
    } ICSR1;
    union {
        unsigned short WORD;
        struct {
            unsigned short OSF:1; /* OSF */
            unsigned short :5; /* */
            unsigned short OCE:1; /* OCE */
            unsigned short OIE:1; /* OIE */
            unsigned short :8; /* */
        } BIT;
    } OCSR;
};

struct st_mmt {
    union {
        unsigned short WORD;
        struct {
            unsigned short :1; /* */
            unsigned short POE6F:1; /* POE6F */
            unsigned short POE5F:1; /* POE5F */
            unsigned short POE4F:1; /* POE4F */
            unsigned short :3; /* */
            unsigned short PIE:1; /* PIE */
            unsigned short :2; /* */
            unsigned short POE6M:2; /* POE6M */
            unsigned short POE5M:2; /* POE5M */
            unsigned short POE4M:2; /* POE4M */
        } BIT;
    } ICSR2;
    unsigned char wk0[1594];
    union {
        unsigned char BYTE;
        struct {
            unsigned char CKS:4; /* CKS */
            unsigned char OLSN:1; /* OLSN */
            unsigned char OLSP:1; /* OLSP */
            unsigned char MD:2; /* MD */
        } BIT;
    } MMT_TMDR;
}
unsigned char wk1[1];
union {
  unsigned char BYTE;
  struct {
    unsigned char TTGE:1; /* TTGE */
    unsigned char CST:1; /* CST */
    unsigned char RPRO:1; /* RPRO */
    unsigned char :3; /* */
    unsigned char TGIEN:1; /* TGIEN */
    unsigned char TGIEM:1; /* TGIEM */
  } BIT;
} MMT_TMDR;

unsigned char wk2[1];
union {
  unsigned char BYTE;
  struct {
    unsigned char TCFD:1; /* TCFD */
    unsigned char :5; /* */
    unsigned char TGFN:1; /* TGFN */
    unsigned char TGFM:1; /* TGFM */
  } BIT;
} TCNR;

unsigned char wk3[1];
unsigned short MMT_TCNT; /* MMT_TCNT */
unsigned short TPDR; /* TPDR */
unsigned short TPBR; /* TPBR */
unsigned short MMT_TDDR; /* MMT_TDDR */
unsigned char wk4[2]; /* */
unsigned short TBRU_B; /* TBRU_B */
unsigned short TGRUU; /* TGRUU */
unsigned short TGRU; /* TGRU */
unsigned short TGRUD; /* TGRUD */
unsigned short TDCNT0; /* TDCNT0 */
unsigned short TDCNT1; /* TDCNT1 */
unsigned short TBRU_F; /* TBRU_F */
unsigned char wk5[2]; /* */
unsigned short TBRV_B; /* TBRV_B */
unsigned short TGRVU; /* TGRVU */
unsigned short TGRV; /* TGRV */
unsigned short TGRVD; /* TGRVD */
unsigned short TDCNT2; /* TDCNT2 */
unsigned short TDCNT3; /* TDCNT3 */
unsigned short TBRV_F; /* TBRV_F */
unsigned char wk6[2]; /* */
unsigned short TBRW_B; /* TBRW_B */
unsigned short TGRWU; /* TGRWU */
unsigned short TGRW; /* TGRW */
unsigned short TGRWD; /* TGRWD */
unsigned short TDCNT4; /* TDCNT4 */
unsigned short TDCNT5;                          /* TDCNT5 */
unsigned short TBRW_F;                          /* TBRW_F */
);                                                 /* */
struct st_portg {
  union {
    unsigned char BYTE;
  }                                                /* */
  struct {
    unsigned char :4;                              /* */
    unsigned char PG3DR:1;                           /* */
    unsigned char PG2DR:1;                           /* */
    unsigned char PG1DR:1;                           /* */
    unsigned char PG0DR:1;                           /* */
  } BIT;                                             /* */
} PGDR;
};                                                             /* */
struct st_cmt {
  union {
    unsigned short WORD;
  }                                                  /* */
  struct {
    unsigned short :14;                            /* */
    unsigned short STR:2;                            /* */
  } BIT;                                              /* */
} CMSTR;
};                                                             /* */
union {
  unsigned short WORD;
}                                                      /* */
struct {
  unsigned short :8;                               /* */
  unsigned short CMF:1;                             /* */
  unsigned short CMIE:1;                            /* */
  unsigned short :4;                                /* */
  unsigned short CKS:2;                             /* */
} BIT;                                                /* */
} CMCSR_0;
};                                                             /* */
unsigned short CMCNT_0;                        /* CMCNT_0 */
unsigned short CMCOR_0;                        /* CMCOR_0 */
};                                                     /* */
union {
  unsigned short WORD;
}                                                      /* */
struct {
  unsigned short :8;                               /* */
  unsigned short CMF:1;                             /* */
  unsigned short CMIE:1;                            /* */
  unsigned short :4;                                /* */
  unsigned short CKS:2;                             /* */
} BIT;                                                /* */
} CMCSR_1;
};                                                             /* */
unsigned short CMCNT_1;                        /* CMCNT_1 */
unsigned short CMCOR_1;                        /* CMCOR_1 */
};                                                     /* */
struct st_ad {
  union {
    unsigned short WORD;
}                                                      /* */
  struct {
    unsigned short :8;                               /* */
    unsigned short CMF:1;                             /* */
    unsigned short CMIE:1;                            /* */
    unsigned short :4;                                /* */
    unsigned short CKS:2;                             /* */
} BIT;                                                /* */
} ADDR0;
};                                                             /* */
union {
  unsigned short WORD;
}                                                      /* */
struct AD;                                           /* */
struct { /* Byte Access */
  unsigned char ADH; /* AD H */
  unsigned char wk; /* */
} BYTE;

struct { /* Bit Access */
  unsigned short AD:10; /* AD */
  unsigned short :6; /* */
} BIT;

} ADDR0;

union {
  unsigned short WORD; /* Word Access */
  struct { /* Byte Access */
    unsigned char ADH; /* AD H */
    unsigned char wk; /* */
  } BYTE;
  struct { /* Bit Access */
    unsigned short AD:10; /* AD */
    unsigned short :6; /* */
  } BIT;
} ADDR1;

union {
  unsigned short WORD; /* Word Access */
  struct { /* Byte Access */
    unsigned char ADH; /* AD H */
    unsigned char wk; /* */
  } BYTE;
  struct { /* Bit Access */
    unsigned short AD:10; /* AD */
    unsigned short :6; /* */
  } BIT;
} ADDR2;

union {
  unsigned short WORD; /* Word Access */
  struct { /* Byte Access */
    unsigned char ADH; /* AD H */
    unsigned char wk; /* */
  } BYTE;
  struct { /* Bit Access */
    unsigned short AD:10; /* AD */
    unsigned short :6; /* */
  } BIT;
} ADDR3;

union {
  unsigned short WORD; /* Word Access */
  struct { /* Byte Access */
    unsigned char ADH; /* AD H */
    unsigned char wk; /* */
  } BYTE;
  struct { /* Bit Access */
    unsigned short AD:10; /* AD */
    unsigned short :6; /* */
  } BIT;
} ADDR4;
union {
    unsigned short WORD;
    struct {
        unsigned char ADH;
        unsigned char wk;
    } BYTE;
    struct {
        unsigned short AD:10;
        unsigned short :6;
    } BIT;
} ADDR4;

union {
    unsigned short WORD;
    struct {
        unsigned char ADH;
        unsigned char wk;
    } BYTE;
    struct {
        unsigned short AD:10;
        unsigned short :6;
    } BIT;
} ADDR5;

union {
    unsigned short WORD;
    struct {
        unsigned char ADH;
        unsigned char wk;
    } BYTE;
    struct {
        unsigned short AD:10;
        unsigned short :6;
    } BIT;
} ADDR6;

union {
    unsigned short WORD;
    struct {
        unsigned char ADH;
        unsigned char wk;
    } BYTE;
    struct {
        unsigned short AD:10;
        unsigned short :6;
    } BIT;
} ADDR7;

union {
    unsigned short WORD;
    struct {
        unsigned char ADH;
        unsigned char wk;
    } BYTE;
    struct {
        unsigned short AD:10;
        unsigned short :6;
    } BIT;
} ADDR8;

union {
    unsigned short WORD;
    struct {
        unsigned char ADH;
        unsigned char wk;
    } BYTE;
    struct {
        unsigned short AD:10;
        unsigned short :6;
    } BIT;
} ADDR9;
union {                       /* ADDR9 */
  unsigned short AD:10;    /* AD Access */
  unsigned short :6;      /* */
} BIT;                      /* */
} ADDR9;                     /* */

union {                       /* ADDR10 */
  unsigned short WORD;      /* Word Access */
  struct {                  /* Byte Access */
    unsigned char ADH;      /* AD H */
    unsigned char wk;       /* */
  } BYTE;                   /* */
  struct {                  /* Bit Access */
    unsigned short AD:10;  /* AD Access */
    unsigned short :6;     /* */
  } BIT;                    /* */
} ADDR10;                     /* */

union {                       /* ADDR11 */
  unsigned short WORD;      /* Word Access */
  struct {                  /* Byte Access */
    unsigned char ADH;      /* AD H */
    unsigned char wk;       /* */
  } BYTE;                   /* */
  struct {                  /* Bit Access */
    unsigned short AD:10;  /* AD Access */
    unsigned short :6;     /* */
  } BIT;                    /* */
} ADDR11;                     /* */

union {                       /* ADDR12 */
  unsigned short WORD;      /* Word Access */
  struct {                  /* Byte Access */
    unsigned char ADH;      /* AD H */
    unsigned char wk;       /* */
  } BYTE;                   /* */
  struct {                  /* Bit Access */
    unsigned short AD:10;  /* AD Access */
    unsigned short :6;     /* */
  } BIT;                    /* */
} ADDR12;                     /* */

union {                       /* ADDR13 */
  unsigned short WORD;      /* Word Access */
  struct {                  /* Byte Access */
    unsigned char ADH;      /* AD H */
    unsigned char wk;       /* */
  } BYTE;                   /* */
  struct {                  /* Bit Access */
    unsigned short AD:10;  /* AD Access */
    unsigned short :6;     /* */
  } BIT;                    /* */
} ADDR13;                     /* */

union {                       /* ADDR14 */
  unsigned short WORD;      /* Word Access */
  struct {                  /* Byte Access */
    unsigned char ADH;      /* AD H */
    unsigned char wk;       /* */
  } BYTE;                   /* */
  struct {                  /* Bit Access */
    unsigned short AD:10;  /* AD Access */
    unsigned short :6;     /* */
  } BIT;                    /* */
} ADDR14;                     /* */
unsigned short WORD;                      /*  Word Access */
struct {                                  /*  Byte Access */
    unsigned char ADH;                 /*    AD H      */
    unsigned char wk;                  /*              */
} BYTE;                            /*              */
struct {                                  /*  Bit Access  */
    unsigned short AD:10;              /*    AD        */
    unsigned short :6;                 /*              */
} BIT;                             /*              */
} ADDR14;
union {                               /* ADDR15       */
    unsigned short WORD;                      /*  Word Access */
    struct {                                  /*  Byte Access */
        unsigned char ADH;                 /*    AD H      */
        unsigned char wk;                  /*              */
    } BYTE;                            /*              */
    struct {                                  /*  Bit Access  */
        unsigned short AD:10;              /*    AD        */
        unsigned short :6;                 /*              */
    } BIT;                             /*              */
} ADDR15;
union {                               /* ADDR16       */
    unsigned short WORD;                      /*  Word Access */
    struct {                                  /*  Byte Access */
        unsigned char ADH;                 /*    AD H      */
        unsigned char wk;                  /*              */
    } BYTE;                            /*              */
    struct {                                  /*  Bit Access  */
        unsigned short AD:10;              /*    AD        */
        unsigned short :6;                 /*              */
    } BIT;                             /*              */
} ADDR16;
union {                               /* ADDR17       */
    unsigned short WORD;                      /*  Word Access */
    struct {                                  /*  Byte Access */
        unsigned char ADH;                 /*    AD H      */
        unsigned char wk;                  /*              */
    } BYTE;                            /*              */
    struct {                                  /*  Bit Access  */
        unsigned short AD:10;              /*    AD        */
        unsigned short :6;                 /*              */
    } BIT;                             /*              */
} ADDR17;
union {                               /* ADDR18       */
    unsigned short WORD;                      /*  Word Access */
    struct {                                  /*  Byte Access */
        unsigned char ADH;                 /*    AD H      */
        unsigned char wk;                  /*              */
    } BYTE;                            /*              */
    struct {                                  /*  Bit Access  */
        unsigned short AD:10;              /*    AD        */
        unsigned short :6;                 /*              */
    } BIT;                             /*              */
} ADDR18;
typedef unsigned short :6;                 /*              */
} BIT;                             /*              */
} ADDR18;                                 /*              */
union {
    unsigned short WORD;                      /*  Word Access */
    struct {
        unsigned char ADH;                 /*    AD H      */
        unsigned char wk;                  /*    ADIE      */
    } BYTE;                            /*              */
    struct {
        unsigned short AD:10;              /*    AD        */
        unsigned short :6;                 /*    ADM       */
        unsigned char CH:3;                /*    CH        */
    } BIT;                             /*              */
} ADDR19;                                 /*              */
unsigned char wk0[56];                          /*              */
union {
    unsigned char BYTE;                       /*  Byte Access */
    struct {
        unsigned char ADF:1;               /*    ADF       */
        unsigned char ADIE:1;              /*    ADIE      */
        unsigned char ADM:2;               /*    ADM       */
        unsigned char :1;                  /*    CH        */
    } BIT;                             /*              */
} ADCSR_0;                                /*              */
union {
    unsigned char BYTE;                       /*  Byte Access */
    struct {
        unsigned char ADF:1;               /*    ADF       */
        unsigned char ADIE:1;              /*    ADIE      */
        unsigned char ADM:2;               /*    ADM       */
        unsigned char :1;                  /*    CH        */
    } BIT;                             /*              */
} ADCSR_1;                                /*              */
union {
    unsigned char BYTE;                       /*  Byte Access */
    struct {
        unsigned char ADF:1;               /*    ADF       */
        unsigned char ADIE:1;              /*    ADIE      */
        unsigned char ADM:2;               /*    ADM       */
        unsigned char :1;                  /*    CH        */
    } BIT;                             /*              */
} ADCSR_2;                                /*              */
unsigned char wk1[5];                           /*              */
union {
    unsigned char BYTE;                       /*  Byte Access */
    struct {
        unsigned char TRGE:1;              /*    TRGE      */
        unsigned char CKS:2;               /*    CKS       */
    } BIT;                             /*              */
} ACSR_0;                                  /*              */
unsigned char ADST:1;              /*    ADST      */
unsigned char ADCS:1;              /*    ADCS      */
unsigned char :3;                  /*              */
} BIT;                             /*              */
} ADCR_0;                                 /*              */
union {
    unsigned char BYTE;                       /*  Byte Access */
    struct {
        unsigned char TRGE:1;              /*    TRGE      */
        unsigned char CKS:2;               /*    CKS       */
        unsigned char ADST:1;              /*    ADST      */
        unsigned char ADCS:1;              /*    ADCS      */
        unsigned char :3;                  /*              */
    } BIT;                             /*              */
} ADCR_1;                                 /*              */
union {
    unsigned char BYTE;                       /*  Byte Access */
    struct {
        unsigned char TRGE:1;              /*    TRGE      */
        unsigned char CKS:2;               /*    CKS       */
        unsigned char ADST:1;              /*    ADST      */
        unsigned char ADCS:1;              /*    ADCS      */
        unsigned char :3;                  /*              */
    } BIT;                             /*              */
} ADCR_2;                                 /*              */
unsigned char wk2[873];                         /*              */
union {
    unsigned char BYTE;                       /*  Byte Access */
    struct {
        unsigned char TRGE:1;              /*    TRGE      */
        unsigned char CKS:2;               /*    CKS       */
        unsigned char ADST:1;              /*    ADST      */
        unsigned char ADCS:1;              /*    ADCS      */
        unsigned char :3;                  /*              */
    } BIT;                             /*              */
} ADTSR;                                  /*              */
};                                                     /*              */
struct st_flash {                                      /* struct FLASH */
    union {
        unsigned char BYTE;                       /*  Byte Access */
        struct {
            unsigned char FWE:1;               /*    FWE       */
            unsigned char SWE:1;               /*    SWE       */
            unsigned char ESU:1;               /*    ESU       */
            unsigned char PSU:1;               /*    PSU       */
            unsigned char EV:1;                /*    EV        */
            unsigned char PV:1;                /*    PV        */
            unsigned char E:1;                 /*    E         */
            unsigned char P:1;                 /*    P         */
        } BIT;                             /*              */
    } FLMCR1;                                 /*              */
    union {                                    /* FLMCR2 */
        unsigned char FWE:1;               /*    FWE       */
        unsigned char SWE:1;               /*    SWE       */
        unsigned char ESU:1;               /*    ESU       */
        unsigned char PSU:1;               /*    PSU       */
        unsigned char EV:1;                /*    EV        */
        unsigned char PV:1;                /*    PV        */
        unsigned char E:1;                 /*    E         */
        unsigned char P:1;                 /*    P         */
    } FLMCR2;                             /*              */
};
```c
unsigned char BYTE;                        /* Byte Access */
struct {
    unsigned char FLER:1;      /* FLER */
    unsigned char :7;          /* */
} BIT;                                     /* */
} FLMCR2;
union {
    unsigned char BYTE;       /* Byte Access */
    struct {
        unsigned char EB:8;    /* EB */
    } BIT;                  /* */
} EBR1;
union {
    unsigned char BYTE;       /* Byte Access */
    struct {
        unsigned char :4;      /* */
        unsigned char EB11:1;  /* EB11 */
        unsigned char EB10:1;  /* EB10 */
        unsigned char EB9:1;   /* EB9 */
        unsigned char EB8:1;   /* EB8 */
    } BIT;                  /* */
} EBR2;
unsigned char wk0[164];                    /* */
union {
    unsigned short WORD;      /* Word Access */
    struct {
        unsigned short :12;    /* */
        unsigned short RAMS:1; /* RAMS */
        unsigned short RAM:3;  /* RAM */
    } BIT;                    /* */
} RAMER;
};                                                /* */
struct st_ubc {
    unsigned short UBARH;     /* UBARH */
    unsigned short UBARL;     /* UBARL */
    union {
        unsigned short WORD;  /* Word Access */
        struct {
            unsigned short UBM31:1; /* UBM31 */
            unsigned short UBM30:1; /* UBM30 */
            unsigned short UBM29:1; /* UBM29 */
            unsigned short UBM28:1; /* UBM28 */
            unsigned short UBM27:1; /* UBM27 */
            unsigned short UBM26:1; /* UBM26 */
            unsigned short UBM25:1; /* UBM25 */
            unsigned short UBM24:1; /* UBM24 */
            unsigned short UBM23:1; /* UBM23 */
            unsigned short UBM22:1; /* UBM22 */
            unsigned short UBM21:1; /* UBM21 */
            unsigned short UBM20:1; /* UBM20 */
            unsigned short UBM19:1; /* UBM19 */
        };
    }
);                                      /* */
```

unsigned short UBM18:1;        /*    UBM18     */
unsigned short UBM17:1;        /*    UBM17     */
unsigned short UBM16:1;        /*    UBM16     */
} BIT;                          /*              */
} UBAMRH;
union {
    unsigned short WORD;        /*    Word Access */
    struct {
        unsigned short UBM15:1;    /*    UBM15     */
        unsigned short UBM14:1;    /*    UBM14     */
        unsigned short UBM13:1;    /*    UBM13     */
        unsigned short UBM12:1;    /*    UBM12     */
        unsigned short UBM11:1;    /*    UBM11     */
        unsigned short UBM10:1;    /*    UBM10     */
        unsigned short UBM9:1;     /*    UBM9      */
        unsigned short UBM8:1;     /*    UBM8      */
        unsigned short UBM7:1;     /*    UBM7      */
        unsigned short UBM6:1;     /*    UBM6      */
        unsigned short UBM5:1;     /*    UBM5      */
        unsigned short UBM4:1;     /*    UBM4      */
        unsigned short UBM3:1;     /*    UBM3      */
        unsigned short UBM2:1;     /*    UBM2      */
        unsigned short UBM1:1;     /*    UBM1      */
        unsigned short UBM0:1;     /*    UBM0      */
    } BIT;                          /*              */
} UBAMRL;
union {
    unsigned short WORD;        /*    Word Access */
    struct {
        unsigned short :8;             /*              */
        unsigned short CP:2;          /*    CP        */
        unsigned short ID:2;          /*    ID        */
        unsigned short RW:2;          /*    RW        */
        unsigned short SZ:2;          /*    SZ        */
    } BIT;                          /*              */
} UBBR;
union {
    unsigned short WORD;        /*    Word Access */
    struct {
        unsigned short :13;          /*              */
        unsigned short CKS:2;        /*    CKS       */
        unsigned short UBID:1;       /*    UBID      */
    } BIT;                          /*              */
} UBCR;
};
struct st_wdt {
    union {
        unsigned char BYTE;          /*    Byte Access */
        struct {
            unsigned char OVF:1;       /*    OVF       */
            unsigned char WIIT:1;      /*    WT/IT     */
        } TCSR;                      /*              */
    } TCSR;                        /*              */
} TCSR;
unsigned char TME:1;               /* TME        */
unsigned char :2;                  /*          */
unsigned char CKS:3;               /* CKS        */
} BIT;                             /*          */
} TCSR;
unsigned char TCNT;               /* TCNT        */
union {
    unsigned char BYTE;           /* Byte Access */
    struct {
        unsigned char WOVF:1;       /* WOVF        */
        unsigned char RSTE:1;       /* RSTE        */
        unsigned char RSTS:1;       /* RSTS        */
        unsigned char :5;          /*          */
    } BIT;                        /*          */
} RSTCSR;                                 /*          */
};

struct st_stby {
    union {
        unsigned char BYTE;        /* Byte Access */
        struct {
            unsigned char SSBY:1;     /* SSBY        */
            unsigned char HIZ:1;      /* HIZ         */
            unsigned char :5;        /*          */
            unsigned char IRQEL:1;    /* IRQEL       */
        } BIT;                      /*          */
    } SBYCR;                                      /*          */
    unsigned char wk0[3];                           /*          */
};

union {
    unsigned char BYTE;        /* Byte Access */
    struct {
        unsigned char :6;          /*          */
        unsigned char AUDSRST:1;    /* AUDSRST     */
        unsigned char RAME:1;       /* RAME        */
    } BIT;                      /*          */
} SYSCR;                                  /*          */
unsigned char wk1[3];                           /*          */
union {
    unsigned short WORD;         /* Word Access */
    struct {
        unsigned short :4;        /*          */
        unsigned short MSTP27:1;   /* MSTP27     */
        unsigned short MSTP26:1;   /* MSTP26     */
        unsigned short MSTP25:1;   /* MSTP25     */
        unsigned short MSTP24:1;   /* MSTP24     */
        unsigned short :3;         /*          */
        unsigned short MSTP20:1;   /* MSTP20     */
        unsigned short MSTP19:1;   /* MSTP19     */
        unsigned short MSTP18:1;   /* MSTP18     */
        unsigned short :2;         /*          */
    } BIT;                        /*          */
} MSTCR1;                               /*          */
union {
    unsigned short WORD;
    struct {
        unsigned short :1;
        unsigned short MSTP14:1; /* MSTP14 */
        unsigned short MSTP13:1; /* MSTP13 */
        unsigned short MSTP12:1; /* MSTP12 */
        unsigned short :2;
        unsigned short MSTP9:1; /* MSTP9 */
        unsigned short :2;
        unsigned short MSTP6:1; /* MSTP6 */
        unsigned short MSTP5:1; /* MSTP5 */
        unsigned short MSTP4:1; /* MSTP4 */
        unsigned short MSTP3:1; /* MSTP3 */
        unsigned short MSTP2:1; /* MSTP2 */
        unsigned short :1;
        unsigned short MSTP0:1; /* MSTP0 */
    } BIT;
    /* */
    /* */
} MSTCR2;

};

struct st_bsc {
    /* struct BSC */
    union {
        unsigned short WORD;
        struct {
            unsigned short :1;
            unsigned short MMTRWE:1; /* MMTRWE */
            unsigned short MTURWE:1; /* MTURWE */
            unsigned short :12;
            unsigned short A0SZ:1; /* A0SZ */
        } BIT;
        /* */
        /* */
    } BCR1;

    union {
        unsigned short WORD;
        struct {
            unsigned short :6;
            unsigned short IW:2; /* IW */
            unsigned short :3;
            unsigned short CW0:1; /* CW0 */
            unsigned short :3;
            unsigned short SW0:1; /* SW0 */
        } BIT;
        /* */
        /* */
    } BCR2;

    union {
        unsigned short WORD;
        struct {
            unsigned short :12;
            unsigned short W:4; /* W */
        } BIT;
        /* */
        /* */
    } WCR1;

};

struct st_dtc {
    /* struct DTC */

} st_dtc;
union {
    unsigned char BYTE; /* DTEA */
    struct {
        unsigned char TGI4A:1; /* Bit Access */
        unsigned char TGI4B:1; /* */
        unsigned char TGI4C:1; /* */
        unsigned char TGI4D:1; /* */
        unsigned char TGI4V:1; /* */
        unsigned char TGI3A:1; /* */
        unsigned char TGI3B:1; /* */
        unsigned char TGI3C:1; /* */
    } BIT;
} DTEA;

union {
    unsigned char BYTE; /* DTEB */
    struct {
        unsigned char TGI3D:1; /* Bit Access */
        unsigned char TGI2A:1; /* */
        unsigned char TGI2B:1; /* */
        unsigned char TGI1A:1; /* */
        unsigned char TGI1B:1; /* */
        unsigned char TGI0A:1; /* */
        unsigned char TGI0B:1; /* */
        unsigned char TGI0C:1; /* */
    } BIT;
} DTEB;

union {
    unsigned char BYTE; /* DTEC */
    struct {
        unsigned char TGI0D:1; /* Bit Access */
        unsigned char ADI0:1; /* */
        unsigned char IRQ0:1; /* */
        unsigned char IRQ1:1; /* */
        unsigned char IRQ2:1; /* */
        unsigned char IRQ3:1; /* */
        unsigned char b1:1; /* */
        unsigned char b0:1; /* */
    } BIT;
} DTEC;

union {
    unsigned char BYTE; /* DTED */
    struct {
        unsigned char b7:1; /* Bit Access */
        unsigned char b6:1; /* */
        unsigned char CMIO:1; /* */
        unsigned char CMII:1; /* */
        unsigned char b3:1; /* */
        unsigned char b2:1; /* */
        unsigned char b1:1; /* */
        unsigned char b0:1; /* */
    } BIT;
} DTED;
} DTED;

unsigned char wk0[2];
union {
    unsigned short WORD;
    struct {
        unsigned short :5;
        unsigned short NMIF:1; /* NMIF */
        unsigned short AE:1; /* AE */
        unsigned short SWDTE:1; /* SWDTE */
        unsigned char DTVEC7:1; /* DTVEC7 */
        unsigned char DTVEC6:1; /* DTVEC6 */
        unsigned char DTVEC5:1; /* DTVEC5 */
        unsigned char DTVEC4:1; /* DTVEC4 */
        unsigned char DTVEC3:1; /* DTVEC3 */
        unsigned char DTVEC2:1; /* DTVEC2 */
        unsigned char DTVEC1:1; /* DTVEC1 */
        unsigned char DTVEC0:1; /* DTVEC0 */
    } BIT;
} DCSR;

unsigned short DTBR;
unsigned char wk1[6];
union {
    unsigned char BYTE;
    struct {
        unsigned char b7:1; /* */
        unsigned char b6:1; /* */
        unsigned char ADI1:1; /* */
        unsigned char ADI2:1; /* */
        unsigned char RXI_2:1; /* */
        unsigned char TXI_2:1; /* */
        unsigned char RXI_3:1; /* */
        unsigned char TXI_3:1; /* */
    } BIT;
} DTEE;

union {
    unsigned char BYTE;
    struct {
        unsigned char RXI_4:1; /* */
        unsigned char TXI_4:1; /* */
        unsigned char TGN:1; /* */
        unsigned char TGM:1; /* */
        unsigned char b3:1; /* */
        unsigned char RM1:1; /* */
        unsigned char b1:1; /* */
        unsigned char b0:1; /* */
    } BIT;
} DTEF;
}

};

struct st_hudi {
    union {
        unsigned short WORD;
    }
} st_hudi; /* struct HUDI */

/* SDIR */
/* Word Access */
struct {
    unsigned short TS:4;  /* Bit Access */
    unsigned short :12;  /*   */
    } BIT;  /*   */
} SDIR;
union {
    unsigned short WORD;  /* Word Access */
    struct {
        unsigned short :15;  /*   */
        unsigned short SDTRF:1;  /*   */
    } BIT;  /*   */
    } SDSR;
unsigned short SDDRH;  /* SDDRH */
unsigned short SDDRL;  /* SDDRL */
} st_hcan2 {
union {
    unsigned short WORD;  /* struct HCAN2 */
    struct {
        unsigned short :8;  /*   */
        unsigned short MCR7:1;  /*   */
        unsigned short :1;  /*   */
        unsigned short MCR5:1;  /*   */
        unsigned short :2;  /*   */
        unsigned short MCR2:1;  /*   */
        unsigned short MCR1:1;  /*   */
        unsigned short MCR0:1;  /*   */
    } BIT;  /*   */
    } MCR;
union {
    unsigned short WORD;  /* GSR */
    struct {
        unsigned short :10;  /*   */
        unsigned short GSR5:1;  /*   */
        unsigned short :1;  /*   */
        unsigned short GSR4:1;  /*   */
        unsigned short :2;  /*   */
        unsigned short GSR3:1;  /*   */
        unsigned short GSR2:1;  /*   */
        unsigned short GSR1:1;  /*   */
        unsigned short GSR0:1;  /*   */
    } BIT;  /*   */
    } GSR;
union {
    unsigned short WORD;  /* HCAN2_BCR1 */
    struct {
        unsigned short TSG1:4;  /* TSG1 */
        unsigned short :1;  /*   */
        unsigned short TSG2:3;  /* TSG2 */
        unsigned short :2;  /*   */
        unsigned short SJW:2;  /* SJW */
        unsigned short :3;  /*   */
        unsigned short BSP:1;  /* BSP */
    } BIT;  /*   */
    } st_hcan2;
union {
    unsigned short WORD;
    struct {
        unsigned short :8;
        unsigned short BRP:8;
    } BIT;
} HCAN2_BCR0;
union {
    unsigned short WORD;
    struct {
        unsigned short IRR15:1;
        unsigned short IRR14:1;
        unsigned short IRR13:1;
        unsigned short IRR12:1;
        unsigned short :2;
        unsigned short IRR9:1;
        unsigned short IRR8:1;
        unsigned short IRR7:1;
        unsigned short IRR6:1;
        unsigned short IRR5:1;
        unsigned short IRR4:1;
        unsigned short IRR3:1;
        unsigned short IRR2:1;
        unsigned short IRR1:1;
        unsigned short IRR0:1;
    } BIT;
} IRR;
union {
    unsigned short WORD;
    struct {
        unsigned short IMR15:1;
        unsigned short IMR14:1;
        unsigned short IMR13:1;
        unsigned short IMR12:1;
        unsigned short :2;
        unsigned short IMR9:1;
        unsigned short IMR8:1;
        unsigned short IMR7:1;
        unsigned short IMR6:1;
        unsigned short IMR5:1;
        unsigned short IMR4:1;
        unsigned short IMR3:1;
        unsigned short IMR2:1;
        unsigned short IMR1:1;
        unsigned short :1;
    } BIT;
} IMR;
unsigned char TEC;
unsigned char REC;
unsigned char wk0[18]; /* */
union { /* TXPR1 */
    unsigned short WORD; /* Word Access */
    struct { /* Bit Access */
        unsigned short TXPR31:1; /* TXPR31 */
        unsigned short TXPR30:1; /* TXPR30 */
        unsigned short TXPR29:1; /* TXPR29 */
        unsigned short TXPR28:1; /* TXPR28 */
        unsigned short TXPR27:1; /* TXPR27 */
        unsigned short TXPR26:1; /* TXPR26 */
        unsigned short TXPR25:1; /* TXPR25 */
        unsigned short TXPR24:1; /* TXPR24 */
        unsigned short TXPR23:1; /* TXPR23 */
        unsigned short TXPR22:1; /* TXPR22 */
        unsigned short TXPR21:1; /* TXPR21 */
        unsigned short TXPR20:1; /* TXPR20 */
        unsigned short TXPR19:1; /* TXPR19 */
        unsigned short TXPR18:1; /* TXPR18 */
        unsigned short TXPR17:1; /* TXPR17 */
        unsigned short TXPR16:1; /* TXPR16 */
    } BIT;
    /* */
} TXPR1;
/* */
union { /* TXPR0 */
    unsigned short WORD; /* Word Access */
    struct { /* Bit Access */
        unsigned short TXPR15:1; /* TXPR15 */
        unsigned short TXPR14:1; /* TXPR14 */
        unsigned short TXPR13:1; /* TXPR13 */
        unsigned short TXPR12:1; /* TXPR12 */
        unsigned short TXPR11:1; /* TXPR11 */
        unsigned short TXPR10:1; /* TXPR10 */
        unsigned short TXPR9:1; /* TXPR9 */
        unsigned short TXPR8:1; /* TXPR8 */
        unsigned short TXPR7:1; /* TXPR7 */
        unsigned short TXPR6:1; /* TXPR6 */
        unsigned short TXPR5:1; /* TXPR5 */
        unsigned short TXPR4:1; /* TXPR4 */
        unsigned short TXPR3:1; /* TXPR3 */
        unsigned short TXPR2:1; /* TXPR2 */
        unsigned short TXPR1:1; /* TXPR1 */
        unsigned short :1; /* */
    } BIT;
    /* */
} TXPR0;
/* */
unsigned char wk1[4]; /* */
union { /* TXCR1 */
    unsigned short WORD; /* Word Access */
    struct { /* Bit Access */
        unsigned short TXCR31:1; /* TXCR31 */
        unsigned short TXCR30:1; /* TXCR30 */
        unsigned short TCR29:1; /* TCR29 */
        unsigned short TXCR28:1; /* TXCR28 */
    } BIT;
    /* */
} TXCR1;
unsigned short TXCR27:1;           /*    TXCR27    */
unsigned short TSCR26:1;           /*    TSCR26    */
unsigned short TXCR25:1;           /*    TXCR25    */
unsigned short TXCR24:1;           /*    TXCR24    */
unsigned short TXCR23:1;           /*    TXCR23    */
unsigned short TXCR22:1;           /*    TXCR22    */
unsigned short TXCR21:1;           /*    TXCR21    */
unsigned short TXCR20:1;           /*    TXCR20    */
unsigned short TXCR19:1;           /*    TXCR19    */
unsigned short TXCR18:1;           /*    TXCR18    */
unsigned short TXCR17:1;           /*    TXCR17    */
unsigned short TXCR16:1;           /*    TXCR16    */
} BIT;                             /*              */
} TXCR1;                                  /*              */
union {
  unsigned short WORD;                      /*  Word Access */
  struct {
    unsigned short TXCR15:1;           /*    TXCR15    */
    unsigned short TXCR14:1;           /*    TXCR14    */
    unsigned short TCR13:1;            /*    TCR13     */
    unsigned short TXCR12:1;           /*    TXCR12    */
    unsigned short TXCR11:1;           /*    TXCR11    */
    unsigned short TSCR10:1;           /*    TSCR10    */
    unsigned short TXCR9:1;            /*    TXCR9     */
    unsigned short TXCR8:1;            /*    TXCR8     */
    unsigned short TXCR7:1;            /*    TXCR7     */
    unsigned short TXCR6:1;            /*    TXCR6     */
    unsigned short TXCR5:1;            /*    TXCR5     */
    unsigned short TXCR4:1;            /*    TXCR4     */
    unsigned short TXCR3:1;            /*    TXCR3     */
    unsigned short TXCR2:1;            /*    TXCR2     */
    unsigned short TXCR1:1;            /*    TXCR1     */
    unsigned short :1;                 /*              */
  } BIT;                             /*              */
  } TXCR0;                                  /*              */
unsigned char wk2[4];                           /*              */
union {
  unsigned short WORD;                      /*  Word Access */
  struct {
    unsigned short TXACK31:1;          /*    TXACK31   */
    unsigned short TXACK30:1;          /*    TXACK30   */
    unsigned short TXACK29:1;          /*    TXACK29   */
    unsigned short TXACK28:1;          /*    TXACK28   */
    unsigned short TXACK27:1;          /*    TXACK27   */
    unsigned short TXACK26:1;          /*    TXACK26   */
    unsigned short TXACK25:1;          /*    TXACK25   */
    unsigned short TXACK24:1;          /*    TXACK24   */
    unsigned short TXACK23:1;          /*    TXACK23   */
    unsigned short TXACK22:1;          /*    TXACK22   */
    unsigned short TXACK21:1;          /*    TXACK21   */
    unsigned short TXACK20:1;          /*    TXACK20   */
  }
unsigned short TXACK19:1; /* TXACK19 */
unsigned short TXACK18:1; /* TXACK18 */
unsigned short TXACK17:1; /* TXACK17 */
unsigned short TXACK16:1; /* TXACK16 */
} BIT;
/* /*
} TXACK1;
union {
unsigned short WORD; /* Word Access */
struct {
unsigned short TXACK15:1; /* TXACK15 */
unsigned short TXACK14:1; /* TXACK14 */
unsigned short TXACK13:1; /* TXACK13 */
unsigned short TXACK12:1; /* TXACK12 */
unsigned short TXACK11:1; /* TXACK11 */
unsigned short TXACK10:1; /* TXACK10 */
unsigned short TXACK9:1; /* TXACK9 */
unsigned short TXACK8:1; /* TXACK8 */
unsigned short TXACK7:1; /* TXACK7 */
unsigned short TXACK6:1; /* TXACK6 */
unsigned short TXACK5:1; /* TXACK5 */
unsigned short TXACK4:1; /* TXACK4 */
unsigned short TXACK3:1; /* TXACK3 */
unsigned short TXACK2:1; /* TXACK2 */
unsigned short TXACK1:1; /* TXACK1 */
unsigned short :1; /* */
} BIT;
/* /*
} TXACK0;
/* /*
unsigned char wk3[4]; /* /*
union {
unsigned short WORD; /* Word Access */
struct {
unsigned short ABACK31:1; /* ABACK31 */
unsigned short ABACK30:1; /* ABACK30 */
unsigned short ABACK29:1; /* ABACK29 */
unsigned short ABACK28:1; /* ABACK28 */
unsigned short ABACK27:1; /* ABACK27 */
unsigned short ABACK26:1; /* ABACK26 */
unsigned short ABACK25:1; /* ABACK25 */
unsigned short ABACK24:1; /* ABACK24 */
unsigned short ABACK23:1; /* ABACK23 */
unsigned short ABACK22:1; /* ABACK22 */
unsigned short ABACK21:1; /* ABACK21 */
unsigned short ABACK20:1; /* ABACK20 */
unsigned short ABACK19:1; /* ABACK19 */
unsigned short ABACK18:1; /* ABACK18 */
unsigned short ABACK17:1; /* ABACK17 */
unsigned short ABACK16:1; /* ABACK16 */
} BIT;
/* /*
} ABACK1;
union {
unsigned short WORD; /* Word Access */
} ABACK0; /* /*
union {
unsigned short WORD; /* Word Access */
} ABACK1; /* /*
union {
unsigned short WORD; /* Word Access */
} ABACK0; /* /*

struct { /* Bit Access */
    unsigned short ABACK15:1; /* ABACK15 */
    unsigned short ABACK14:1; /* ABACK14 */
    unsigned short ABACK13:1; /* ABACK13 */
    unsigned short ABACK12:1; /* ABACK12 */
    unsigned short ABACK11:1; /* ABACK11 */
    unsigned short ABACK10:1; /* ABACK10 */
    unsigned short ABACK9:1; /* ABACK9 */
    unsigned short ABACK8:1; /* ABACK8 */
    unsigned short ABACK7:1; /* ABACK7 */
    unsigned short ABACK6:1; /* ABACK6 */
    unsigned short ABACK5:1; /* ABACK5 */
    unsigned short ABACK4:1; /* ABACK4 */
    unsigned short ABACK3:1; /* ABACK3 */
    unsigned short ABACK2:1; /* ABACK2 */
    unsigned short ABACK1:1; /* ABACK1 */
    unsigned short :1; /* */
} BIT;
} ABACK0;
unsigned char wk4[4];
union {
    /* RXPR1 */
    unsigned short WORD; /* Word Access */
    /* Bit Access */
    struct {
        unsigned short RXPR31:1; /* RXPR31 */
        unsigned short RXPR30:1; /* RXPR30 */
        unsigned short RXPR29:1; /* RXPR29 */
        unsigned short RXPR28:1; /* RXPR28 */
        unsigned short RXPR27:1; /* RXPR27 */
        unsigned short RXPR26:1; /* RXPR26 */
        unsigned short RXPR25:1; /* RXPR25 */
        unsigned short RXPR24:1; /* RXPR24 */
        unsigned short RXPR23:1; /* RXPR23 */
        unsigned short RXPR22:1; /* RXPR22 */
        unsigned short RXPR21:1; /* RXPR21 */
        unsigned short RXPR20:1; /* RXPR20 */
        unsigned short RXPR19:1; /* RXPR19 */
        unsigned short RXPR18:1; /* RXPR18 */
        unsigned short RXPR17:1; /* RXPR17 */
        unsigned short RXPR16:1; /* RXPR16 */
    } BIT;
} RXPR1;
union {
    /* RXPR0 */
    unsigned short WORD; /* Word Access */
    /* Bit Access */
    struct {
        unsigned short RXPR15:1; /* RXPR15 */
        unsigned short RXPR14:1; /* RXPR14 */
        unsigned short RXPR13:1; /* RXPR13 */
        unsigned short RXPR12:1; /* RXPR12 */
        unsigned short RXPR11:1; /* RXPR11 */
        unsigned short RXPR10:1; /* RXPR10 */
        unsigned short RXPR9:1; /* RXPR9 */
    } BIT;
} RXPR0;
unsigned short RXPR8:1; /* RXPR8 */
unsigned short RXPR7:1; /* RXPR7 */
unsigned short RXPR6:1; /* RXPR6 */
unsigned short RXPR5:1; /* RXPR5 */
unsigned short RXPR4:1; /* RXPR4 */
unsigned short RXPR3:1; /* RXPR3 */
unsigned short RXPR2:1; /* RXPR2 */
unsigned short RXPR1:1; /* RXPR1 */
unsigned short RXPR0:1; /* RXPR0 */
} BIT;
/* */
} RXPR0;
/* */
unsigned char wk5[4]; /* */
union {
  /* RFPR1 */
  unsigned short WORD; /* Word Access */
  struct {
    unsigned short RFPR31:1; /* RFPR31 */
    unsigned short RFPR30:1; /* RFPR30 */
    unsigned short RFPR29:1; /* RFPR29 */
    unsigned short RFPR28:1; /* RFPR28 */
    unsigned short RFPR27:1; /* RFPR27 */
    unsigned short RFPR26:1; /* RFPR26 */
    unsigned short RFPR25:1; /* RFPR25 */
    unsigned short RFPR24:1; /* RFPR24 */
    unsigned short RFPR23:1; /* RFPR23 */
    unsigned short RFPR22:1; /* RFPR22 */
    unsigned short RFPR21:1; /* RFPR21 */
    unsigned short RFPR20:1; /* RFPR20 */
    unsigned short RFPR19:1; /* RFPR19 */
    unsigned short RFPR18:1; /* RFPR18 */
    unsigned short RFPR17:1; /* RFPR17 */
    unsigned short RFPR16:1; /* RFPR16 */
  } BIT;
  /* */
} RFPR1;
/* */
union {
  /* RFPR0 */
  unsigned short WORD; /* Word Access */
  struct {
    unsigned short RFPR15:1; /* RFPR15 */
    unsigned short RFPR14:1; /* RFPR14 */
    unsigned short RFPR13:1; /* RFPR13 */
    unsigned short RFPR12:1; /* RFPR12 */
    unsigned short RFPR11:1; /* RFPR11 */
    unsigned short RFPR10:1; /* RFPR10 */
    unsigned short RFPR9:1; /* RFPR9 */
    unsigned short RFPR8:1; /* RFPR8 */
    unsigned short RFPR7:1; /* RFPR7 */
    unsigned short RFPR6:1; /* RFPR6 */
    unsigned short RFPR5:1; /* RFPR5 */
    unsigned short RFPR4:1; /* RFPR4 */
    unsigned short RFPR3:1; /* RFPR3 */
    unsigned short RFPR2:1; /* RFPR2 */
    unsigned short RFPR1:1; /* RFPR1 */
unsigned short RFPR0:1;            /*    RFPR0     */
} BIT;
} RFPR0;
unsigned char wk6[4];                           /*              */
union {
  unsigned short WORD;
  struct {
    unsigned short MBIMR31:1;          /*    MBIMR31   */
    unsigned short MBIMR30:1;          /*    MBIMR30   */
    unsigned short MBIMR29:1;          /*    MBIMR29   */
    unsigned short MBIMR28:1;          /*    MBIMR28   */
    unsigned short MBIMR27:1;          /*    MBIMR27   */
    unsigned short MBIMR26:1;          /*    MBIMR26   */
    unsigned short MBIMR25:1;          /*    MBIMR25   */
    unsigned short MBIMR24:1;          /*    MBIMR24   */
    unsigned short MBIMR23:1;          /*    MBIMR23   */
    unsigned short MBIMR22:1;          /*    MBIMR22   */
    unsigned short MBIMR21:1;          /*    MBIMR21   */
    unsigned short MBIMR20:1;          /*    MBIMR20   */
    unsigned short MBIMR19:1;          /*    MBIMR19   */
    unsigned short MBIMR18:1;          /*    MBIMR18   */
    unsigned short MBIMR17:1;          /*    MBIMR17   */
    unsigned short MBIMR16:1;          /*    MBIMR16   */
  } BIT;
} MBIMR1;
union {
  unsigned short WORD;
  struct {
    unsigned short MBIMR15:1;          /*    MBIMR15   */
    unsigned short MBIMR14:1;          /*    MBIMR14   */
    unsigned short MBIMR13:1;          /*    MBIMR13   */
    unsigned short MBIMR12:1;          /*    MBIMR12   */
    unsigned short MBIMR11:1;          /*    MBIMR11   */
    unsigned short MBIMR10:1;          /*    MBIMR10   */
    unsigned short MBIMR9:1;           /*    MBIMR9    */
    unsigned short MBIMR8:1;           /*    MBIMR8    */
    unsigned short MBIMR7:1;           /*    MBIMR7    */
    unsigned short MBIMR6:1;           /*    MBIMR6    */
    unsigned short MBIMR5:1;           /*    MBIMR5    */
    unsigned short MBIMR4:1;           /*    MBIMR4    */
    unsigned short MBIMR3:1;           /*    MBIMR3    */
    unsigned short MBIMR2:1;           /*    MBIMR2    */
    unsigned short MBIMR1:1;           /*    MBIMR1    */
    unsigned short MBIMR0:1;           /*    MBIMR0    */
  } BIT;
} MBIMR0;
unsigned char wk7[4];                           /*              */
union {
  unsigned short WORD;
  struct {
    unsigned short UMSR31:1;
  } UMSR1;
} UMSR0;
union {
  unsigned short WORD;
  struct {
    unsigned short UMSR31:1;
  } UMSR1;
} UMSR0;
unsigned short UMSR30:1; /* UMSR30 */
unsigned short UMSR29:1; /* UMSR29 */
unsigned short UMSR28:1; /* UMSR28 */
unsigned short UMSR27:1; /* UMSR27 */
unsigned short UMSR26:1; /* UMSR26 */
unsigned short UMSR25:1; /* UMSR25 */
unsigned short UMSR24:1; /* UMSR24 */
unsigned short UMSR23:1; /* UMSR23 */
unsigned short UMSR22:1; /* UMSR22 */
unsigned short UMSR21:1; /* UMSR21 */
unsigned short UMSR20:1; /* UMSR20 */
unsigned short UMSR19:1; /* UMSR19 */
unsigned short UMSR18:1; /* UMSR18 */
unsigned short UMSR17:1; /* UMSR17 */
unsigned short UMSR16:1; /* UMSR16 */
} BIT;
/* */
} UMSR1;
/* */
union {
    /* UMSR0 */
    unsigned short WORD;
    /* Word Access */
    struct {
        unsigned short UMSR15:1; /* UMSR15 */
        unsigned short UMSR14:1; /* UMSR14 */
        unsigned short UMSR13:1; /* UMSR13 */
        unsigned short UMSR12:1; /* UMSR12 */
        unsigned short UMSR11:1; /* UMSR11 */
        unsigned short UMSR10:1; /* UMSR10 */
        unsigned short UMSR9:1; /* UMSR9 */
    } BIT;
    /* */
    } UMSR0;
    /* */
unsigned char wk8[36]; /* */
unsigned short TCNTR; /* TCNTR */
union {
    /* TCR */
    unsigned short WORD;
    /* Word Access */
    struct {
        unsigned short TCR15:1; /* TCR15 */
        unsigned short TCR14:1; /* TCR14 */
        unsigned short TCR13:1; /* TCR13 */
        unsigned short TCR12:1; /* TCR12 */
        unsigned short TCR11:1; /* TCR11 */
        unsigned short TCR10:1; /* TCR10 */
        unsigned short TCR9:1; /* TCR9 */
        unsigned short TCR8:1; /* TCR8 */
        unsigned short TCR7:1; /* TCR7 */
        unsigned short :1; /* */
        unsigned short TPSC:6; /* TPSC */
    } BIT;
    /* */
    } TCR;
    /* */
union { /* TSR */
    unsigned short WORD;
    /* Word Access */
    struct {
        unsigned short :13;
        /* */
    }
unsigned short TSR2:1; /* TSR2 */
unsigned short TSR1:1; /* TSR1 */
unsigned short TSR0:1; /* TSR0 */
} BIT;
} TSR;
unsigned short TDCR; /* TDCR */
unsigned short LOSR; /* LOSR */
unsigned char wk9[2]; /* */
unsigned short HCAN2_ICR0; /* HCAN2_ICR0 */
unsigned short HCAN2_ICR1; /* HCAN2_ICR1 */
unsigned short TCMR0; /* TCMR0 */
unsigned short TCMR1; /* TCMR1 */
unsigned char wk10[108]; /* */

struct st_mb {
    union {
        unsigned char BYTE; /* Byte Access */
        struct {
            unsigned char :1;
            unsigned char STDID10:1; /* STDID10 */
            unsigned char STDID9:1; /* STDID9 */
            unsigned char STDID8:1; /* STDID8 */
            unsigned char STDID7:1; /* STDID7 */
            unsigned char STDID6:1; /* STDID6 */
            unsigned char STDID5:1; /* STDID5 */
            unsigned char STDID4:1; /* STDID4 */
        } BIT;
    } MB0;
    union {
        unsigned char BYTE; /* Byte Access */
        struct {
            unsigned char STDID:4; /* STDID */
            unsigned char RTR:1; /* RTR */
            unsigned char IDE:1; /* IDE */
            unsigned char EXTID17:1; /* EXTID17 */
            unsigned char EXTID16:1; /* EXTID16 */
        } BIT;
    } MB1;
    union {
        unsigned char BYTE; /* Byte Access */
        struct {
            unsigned char EXTID15:1; /* EXTID15 */
            unsigned char EXTID14:1; /* EXTID14 */
            unsigned char EXTID13:1; /* EXTID13 */
            unsigned char EXTID12:1; /* EXTID12 */
            unsigned char EXTID11:1; /* EXTID11 */
            unsigned char EXTID10:1; /* EXTID10 */
            unsigned char EXTID9:1; /* EXTID9 */
            unsigned char EXTID8:1; /* EXTID8 */
        } BIT;
    } MB2;
    union {
        unsigned char BYTE; /* Byte Access */
    } MB3;
};
unsigned char BYTE;                      /*  Byte Access */
structure                                /*  Bit Access  */
    unsigned char EXTID7:1;            /*    EXTID7    */
    unsigned char EXTID6:1;            /*    EXTID6    */
    unsigned char EXTID5:1;            /*    EXTID5    */
    unsigned char EXTID4:1;            /*    EXTID4    */
    unsigned char EXTID3:1;            /*    EXTID3    */
    unsigned char EXTID2:1;            /*    EXTID2    */
    unsigned char EXTID1:1;            /*    EXTID1    */
    unsigned char EXTID0:1;            /*    EXTID0    */
} BIT;
} MB3;
*/ */
union {                                     /* MB4          */
    unsigned char BYTE;                      /*  Byte Access */
structure                                /*  Bit Access  */
    unsigned char CCM:1;               /*    CCM       */
    unsigned char TTE:1;               /*    TTE       */
    unsigned char NMC:1;               /*    NMC       */
    unsigned char ATX:1;               /*    ATX       */
    unsigned char DART:1;              /*    DART      */
    unsigned char MBC:3;               /*    MBC       */
} BIT;
} MB4;                                    /*              */
*/ */
union {                                     /* MB5          */
    unsigned char BYTE;                      /*  Byte Access */
structure                                /*  Bit Access  */
    unsigned char PTE:1;               /*    PTE       */
    unsigned char TCT:1;               /*    TCT       */
    unsigned char CBE:1;               /*    CBE       */
    unsigned char :1;                  /*              */
    unsigned char DLC:4;               /*    DLC       */
} BIT;
} MB5;                                    /*              */
*/ */
unsigned char TIME_STAMP;                       /* TIME_STAMP   */
unsigned char wk11[1];                          /*              */
unsigned char MSG_DATA[8];                      /* MSG_DATA     */
unsigned char LAFM0[2];                         /* LAFM0        */
unsigned char LAFM1[2];                         /* LAFM1        */
unsigned char wk12[12];                         /*              */
}mb[32];
*/ */
#define P_SCI2 (*(volatile struct st_sci *)0xFFFF81C0)   /* SCI2 Address */
#define P_SCI3 (*(volatile struct st_sci *)0xFFFF81D0)   /* SCI3 Address */
#define P_SCI4 (*(volatile struct st_sci *)0xFFFF81E0)   /* SCI4 Address */
#define P_MTU34 (*(volatile struct st_mtu34 *)0xFFFF8200)/* MTU34 Address */
#define P_MTU0 (*(volatile struct st_mtu0 *)0xFFFF8260)  /* MTU0 Address */
#define P_MTU1 (*(volatile struct st_mtu1 *)0xFFFF8280)  /* MTU1 Address */
#define P_MTU2 (*(volatile struct st_mtu2 *)0xFFFF82A0)  /* MTU2 Address */
#define P_INTC (*(volatile struct st_intc *)0xFFFF8348)  /* INTC Address */
#define P_PORTA (*(volatile struct st_porta *)0xFFFF8382)/* PORTA Address */
Note: The header file shown here is the file generated automatically by Renesas Technology Integrated Development Environment (High-performance Embedded Workshop) C/C++ Compiler Ver. 6. Please refer to development environment upgrade information for details of modifications and amendments.
SH7046 Group On-Chip I/O Volume
Application Note