SH7450 Group/SH7451 Group

Register Definition Header File

R01AN0190EJ0102
Rev.1.02
Sep 01, 2010

Introduction

This application note describes the contents of the register definition header file for the SH7450 Group and SH7451 Group and shows examples of its use.

Target Devices

SH74504 (R5F74504KBG)

SH74513 (R5F74513KBG)

Contents

1. Explanation

This document applies to the following microcomputers (MCUs) and coding tools:

- MCUs: SH74504 (R5F74504KBG) and SH74513 (R5F74513KBG)
- Compiler: SH SERIES C/C++ Compiler V.9.03.02 (SHC compiler)
- MISRA C Rule Checker: SQMlint V.1.03

1.1 Structure

The structure of the special function register definition header file (SFR header file) is as follows:

(1) All registers in the MCU hardware manual are defined.
(2) Registers and bit symbols follow those found in the MCU hardware manual (Rev. 1.00).
(3) The access size of each register (including bit-field access) is defined based on the maximum bit width of the register.
(4) Register addresses are defined with the P4 area address (accessible in privileged mode only).
(5) 8-/16-/32-bit access sizes are written as BYTE/WORD/LONG, respectively.

Note that to improve the use of the SFR header file, some register and bit-field names are only defined in the SFR header file. Refer to section "2.2 Special Register Definitions" in this document for details.

1.2 MISRA C Compliance

The SFR header file is created in compliance with MISRA C:1998 rules. Information on features and compliance is as follows:

(1) Features

To comply with MISRA C:1998 Rule 13, unsigned 8-/16-/32-bit data types are defined using typedef as _U1/_U2/_U4, respectively.

- MISRA C:1998 Rule 13 (Advisory)
  The basic types of char, int, short, long, float and double should not be used, but specific-length equivalents should be typedef'd for the specific compiler, and these type names used in the code.

(2) MISRA C Compliance

The SFR header file does not conform to rules 110, 111, and 45. An outline of each rule and the reason for noncompliance is as follows:

- MISRA C:1998 Rule 110 (Advisory)
  Unions shall not be used to access the sub-parts of larger data types.

  Reason for noncompliance
  The registers are defined in structures by peripheral functions, and the SFR header file is created to access registers in byte, word, or longword sizes (including bit-field access).

- MISRA C:1998 Rule 111 (Advisory)
  Bit fields shall only be defined to be of type unsigned int or signed int.

  Reason for noncompliance
  The Renesas Electronics SHC compiler generates code that accesses registers based on the data type of the bit-field definition.

- MISRA C:1998 Rule 45 (Advisory)
  Type casting from any type to or from pointers shall not be used.

  Reason for noncompliance
  Registers are defined by structures for each peripheral function, and the start address of the structures are defined with numerical values by a macro definition (#define).

Even though the SFR header file does not comply with these MISRA C rules, the Renesas Electronics SHC compiler generates code as expected, so there should be no problems if this compiler is used.

2. Using the SFR Header File

This chapter describes the basic usage and the special register definitions of the SFR header file.

2.1 Basic Usage

The basic usage of the SFR header file is described below.

2.1.1 Basic Formats

The basic formats for a blanket access and a bit-field access using the SFR header file are as follows:

(1) Basic format for a blanket access
Module name*1 .register symbol.access size*2
(2) Basic format for a bit-field access
Module name*1.register symbol.BIT.bit symbol

*1 Module name: The module name is defined by a define macro. For details, refer to the define declarations at the end of the SFR header file.
*2 Access size: Access size is defined based on the register size and written in BYTE, WORD, and LONG. For details, refer to the MCU hardware manual or the header file contents.

2.1.2 Basic Usage Examples

As a basic usage example of the SFR header file, definitions for the TRAPA exception register (TRA) are described below. The TRA register is a 32-bit register assigned to address H'FF00 0020.

TRAPA Exception Register (TRA) <P4 address: location H'FF00 0020>

Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
After Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	TRACODE								—	—
After Reset:	0	0	0	0	0	0	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	0	0

<After Reset: Undefined>

Bit	Abbreviation	After Reset	R	W	Description
31 to 10	—	All 0	0	0	Reserved Bits These bits are always read as “0”. The write value should always be “0”.
9 to 2	TRACODE	Undefined	R	W	TRAPA Code 8-bit immediate data of TRAPA instruction is set
1, 0	—	All 0	0	0	Reserved Bits These bits are always read as “0”. The write value should always be “0”.

(1) Definition of Structure Contents

The TRA register is defined by the struct st_exp structure in the EXP register.

```
/*****************************************************************************
 * Exception Handling (EXP)                                                  *
 *****************************************************************************/
struct st_exp                                   /* struct exp        */
{                                               /*                   */
    union                                       /*                   */
    {                                           /* TRA               */
        _U4 LONG;                               /*  Long Access      */
        struct                                  /*                   */
        {                                       /*  Bit Access       */
            _U4 reserved1:22;                   /*   Reserved Bits   */
            _U4 TRACODE:8;                      /*   TRACODE[9:2]    */
            _U4 reserved2:2;                    /*   Reserved Bits   */
        }BIT;                                   /*                   */
    }TRA;                                       /*                   */

    union                                       /*                   */
    {                                           /* EXPEVT            */
        _U4 LONG;                               /*  Long Access      */
        struct                                  /*                   */
        {                                       /*  Bit Access       */
            _U4 reserved1:20;                   /*   Reserved Bits   */
            _U4 EXPCODE:12;                     /*   EXPCODE[11:0]   */
        }BIT;                                   /*                   */
    }EXPEVT;                                    /*                   */

    union                                       /*                   */
    {                                           /* INTEVT            */
        _U4 LONG;                               /*  Long Access      */
        struct                                  /*                   */
        {                                       /*  Bit Access       */
            _U4 reserved1:18;                   /*   Reserved Bits   */
            _U4 INTCODE:14;                     /*   INTCODE[13:0]   */
        }BIT;                                   /*                   */
    }INTEVT;                                    /*                   */

    _U1 reserved1[0x2EFFD8];

    union                                       /*                   */
    {                                           /* EXPMASK           */
        _U4 LONG;                               /*  Long Access      */
        struct                                  /*                   */
        {                                       /*  Bit Access       */
            _U4 reserved1:27;                   /*   Reserved Bits   */
            _U4 MMCAW:1;                        /*   MMCAW[4:4]      */
            _U4 reserved2:3;                    /*   Reserved Bits   */
            _U4 RTEDS:1;                        /*   RTEDS[0:0]      */
        }BIT;                                   /*                   */
    }EXPMASK;                                   /*                   */

};
                                        :
                                        :
                                        :

/*****************************************************************************
 *  Defines                                                                  *
 *****************************************************************************/
#define EXP        (*(volatile struct st_exp *)0xFF000020ul)        /* Exception Handling (EXP)  */
```

(2) Blanket Access

EXP.TRA.LONG is used as a blanket access (32-bit) of the TRA register.

Example: An example of a blanket access of the TRA register.

```
EXP.TRA.LONG = 0x000003FCul;

unsigned long temp;
temp = EXP.TRA.LONG;
```

(3) Bit-Field Access

EXP.TRA.BIT.TRACODE is used for bit-field accessing the TRACODE bit in the TRA register.

The Renesas Electronics SHC compiler generates code that accesses a register based on the data type of the bit-field definition. Note that compilers other than the SHC complier may generate code that performs an 8-bit register access regardless of the data type of the bit-field definition.

Example: An example of accessing the TRACODE bit in the TRA register with a bit-field access.

```
EXP.TRA.BIT.TRACODE = 0xFFul;

unsigned long temp;
temp = EXP.TRA.BIT.TRACODE;
```

2.1.3 Notes on Bit-Field Access

When a bit-field access is used to write to a register, the values of the fields other than the target field are read, and then the same values are written back. If the register value changes between when the values of these fields are read and when the values are written back, unintended changes could happen. A bit-field access should therefore not be used in some cases.

For example, when a bit-field access is used to clear a particular flag in the INTREQ register, the read values of flags except for the flag to be cleared are written back without being modified. Therefore, if an interrupt request occurs, and the value of flags except the flag to be cleared change between reading and writing them back to the INTREQ register (if a flag value is 0 when read but 1 when written back, for example), the interrupt request is cleared to 0. To prevent this, perform a blanket access as shown in the example below.

Example: An example of clearing the IRQ7 bit in the INTREQ register to 0 when using edge detection.

```
temp = INTC.INTREQ.LONG;
if (temp & 0x01000000) {
    INTC.INTREQ.LONG = 0xFE000000ul;
}
```

Carefully read the MCU hardware manual to confirm compliance with the hardware limitations.

2.2 Special Register Definitions

This section describes bit and register names unique to the SFR header file. Make sure there is no symbol overlap in the user program.

(1) Registers with "_" Preceding the Bit Symbol

The following registers are defined using a format in which the bit symbols are preceded by an underscore "_" in order to avoid overlap between the module name (define macro name) definition and the bit symbol.

- Interrupt priority setting registers 0 to 12 (INT2PRI0 to INT2PRI12)
- Interrupt source register 00 (INT2A00)
- Interrupt source register 01 (INT2A01)
- Interrupt source register 10 (INT2A10)
- Interrupt source register 11 (INT2A11)
- Interrupt mask register 0 (INT2MSKR)
- Interrupt mask register 1 (INT2MSKR1)
- Interrupt mask clear register 0 (INT2MSKCR)
- Interrupt mask clear register 1 (INT2MSKCR1)
- A/Di conversion value addition count select register (AD0ADC and AD1ADC) (i = 0, 1)
- Some symbols (DRO, DRI0, DRI1, DRI2, and PDAC) in the module stop register 0 (MSTPCR0)

Example: Declaration of the interrupt priority setting register 10 (INT2PRI10).

```
union                               /*                          */
{                                   /* INT2PRI10                */
    _U4 LONG;                       /*  Long Access             */
    struct                          /*                          */
    {                               /*  Bit Access              */
        _U4 reserved1:3;            /*   Reserved Bits          */
        _U4 _CAN0:5;                /*   _CAN0[28:24]           */
        _U4 reserved2:3;            /*   Reserved Bits          */
        _U4 _CAN1:5;                /*   _CAN1[20:16]           */
        _U4 reserved3:3;            /*   Reserved Bits          */
        _U4 _CAN2:5;                /*   _CAN2[12:8]            */
        _U4 reserved4:3;            /*   Reserved Bits          */
        _U4 _CAN3:5;                /*   _CAN3[4:0]             */
    }BIT;                           /*                          */
}INT2PRI10;                         /*                          */
```

(2) Registers with No Bit Symbol Defined

The following registers are defined using a format in which no bit-field structure members are used because the number of bits changes according to the operating mode of timer TOU.

- TOUnm counter (TOnmCNT) (n = 0 to 4; m = 0 to 7)
- TOUnm reload register (TOnmRLD)

Example: Declaration of the TOU00 counter (TO00CNT) and TOU00 reload register (TO00RLD).

```
union                               /*                          */
{                                   /* TO00CNT                  */
    _U4 LONG;                       /*  Long Access             */
}TO00CNT;                           /*                          */

union                               /*                          */
{                                   /* TO00RLD                  */
    _U4 LONG;                       /*  Long Access             */
}TO00RLD;                           /*                          */
```

The following registers are defined with no bit-field structure members because the bit format changes depending on the right-shift and left-shift formats, and the number of bits changes according to the presence or absence of the addition mode.

- A/D0 data registers 0 to 15 (AD0DR0 to AD0DR15)
- A/D1 data registers 0 to 7 (AD1DR0 to AD1DR7)
- A/D0 data register DIAG0 (AD0DRD)
- A/D1 data register DIAG1 (AD1DRD)

(3) SPiDR Register

Since the read and write access size for the SPiDR register differs depending on the setting of the RSPI longword access/word access setting bit (SPLW) in the RSPIi data control register (SPiDCR), a structure definition for word access is added to accommodate both word and longword register access (i = 0 to 2).

- RSPIi data register (SPiDR)

Example: Declaration of the RSPI0 data register (SP0DR).

```
    union                               /*                          */
    {                                   /* SP0DR                    */
        _U4 LONG;                       /*  Long Access             */
        struct                          /*                          */
        {                               /*  Bit Access              */
            _U4 SPD31:1;                /*   SPD31[31:31]           */
            _U4 SPD30:1;                /*   SPD30[30:30]           */
            _U4 SPD29:1;                /*   SPD29[29:29]           */
            _U4 SPD28:1;                /*   SPD28[28:28]           */
            _U4 SPD27:1;                /*   SPD27[27:27]           */
            _U4 SPD26:1;                /*   SPD26[26:26]           */
            _U4 SPD25:1;                /*   SPD25[25:25]           */
            _U4 SPD24:1;                /*   SPD24[24:24]           */
            _U4 SPD23:1;                /*   SPD23[23:23]           */
            _U4 SPD22:1;                /*   SPD22[22:22]           */
            _U4 SPD21:1;                /*   SPD21[21:21]           */
            _U4 SPD20:1;                /*   SPD20[20:20]           */
            _U4 SPD19:1;                /*   SPD19[19:19]           */
            _U4 SPD18:1;                /*   SPD18[18:18]           */
            _U4 SPD17:1;                /*   SPD17[17:17]           */
            _U4 SPD16:1;                /*   SPD16[16:16]           */
            _U4 SPD15:1;                /*   SPD15[15:15]           */
            _U4 SPD14:1;                /*   SPD14[14:14]           */
            _U4 SPD13:1;                /*   SPD13[13:13]           */
            _U4 SPD12:1;                /*   SPD12[12:12]           */
            _U4 SPD11:1;                /*   SPD11[11:11]           */
            _U4 SPD10:1;                /*   SPD10[10:10]           */
            _U4 SPD9:1;                 /*   SPD9[9:9]              */
            _U4 SPD8:1;                 /*   SPD8[8:8]              */
            _U4 SPD7:1;                 /*   SPD7[7:7]              */
            _U4 SPD6:1;                 /*   SPD6[6:6]              */
            _U4 SPD5:1;                 /*   SPD5[5:5]              */
            _U4 SPD4:1;                 /*   SPD4[4:4]              */
            _U4 SPD3:1;                 /*   SPD3[3:3]              */
            _U4 SPD2:1;                 /*   SPD2[2:2]              */
            _U4 SPD1:1;                 /*   SPD1[1:1]              */
            _U4 SPD0:1;                 /*   SPD0[0:0]              */
        }BIT;                           /*                          */
        {                               /*  Bit Access              */
            _U2 SPD;                    /*   SPD[31:16]             */
            _U2 reserved1;              /*   Reserved Bits          */
        }WORD;                          /*                          */
    }SP0DR;                             /*                          */
```

(4) CiMCTLj Register

Since the roles and names of the bits in this register differ in transmit and receive mode, the bit-fields are defined according to user-defined structure names to accommodate both transmit and receive operations.

- CANi message control register j (CiMCTLj) (i = 0 to 4; j = 0 to 63)

Example: Declaration of the CAN0 message control register 0 (C0MCTL0).

```
    union                                   /*                      */
    {                                       /* C0MCTL0              */
        _U1 BYTE;                           /*  Byte Access         */
        struct                              /*                      */
        {                                   /*  Bit Access          */
            _U1 reserved1:1;                /*   Reserved Bits      */
            _U1 RECREQ:1;                   /*   RECREQ[6:6]        */
            _U1 reserved2:3;                /*   Reserved Bits      */
            _U1 MSGLOST:1;                  /*   MSGLOST[2:2]       */
            _U1 INVALDATA:1;                /*   INVALDATA[1:1]     */
            _U1 NEWDATA:1;                  /*   NEWDATA[0:0]       */
        }BIT;                               /*                      */
        struct                              /*                      */
        {                                   /*  Bit Access          */
            _U1 reserved1:1;                /*   Reserved Bits      */
            _U1 RECREQ:1;                   /*   RECREQ[6:6]        */
            _U1 reserved2:3;                /*   Reserved Bits      */
            _U1 MSGLOST:1;                  /*   MSGLOST[2:2]       */
            _U1 INVALDATA:1;                /*   INVALDATA[1:1]     */
            _U1 NEWDATA:1;                  /*   NEWDATA[0:0]       */
        }BIT_RECEIVE;                       /*                      */
    }C0MCTL0;                               /*                      */
```

Example: Declaration of the CAN0 message control register 32 (C0MCTL32).

```
    union                                   /*                      */
    {                                       /* C0MCTL32             */
        _U1 BYTE;                           /*  Byte Access         */
        struct                              /*                      */
        {                                   /*  Bit Access          */
            _U1 TRMREQ:1;                   /*   TRMREQ[7:7]        */
            _U1 RECREQ:1;                   /*   RECREQ[6:6]        */
            _U1 reserved1:1;                /*   Reserved Bits      */
            _U1 ONESHOT:1;                  /*   ONESHOT[4:4]       */
            _U1 reserved2:1;                /*   Reserved Bits      */
            _U1 TRMABT:1;                   /*   TRMABT[2:2]        */
            _U1 TRMACTIVE:1;                /*   TRMACTIVE[1:1]     */
            _U1 SENTDATA:1;                 /*   SENTDATA[0:0]      */
        }BIT_TRANSMIT;                      /*                      */
        struct                              /*                      */
        {                                   /*  Bit Access          */
            _U1 TRMREQ:1;                   /*   TRMREQ[7:7]        */
            _U1 RECREQ:1;                   /*   RECREQ[6:6]        */
            _U1 reserved1:1;                /*   Reserved Bits      */
            _U1 ONESHOT:1;                  /*   ONESHOT[4:4]       */
            _U1 reserved2:1;                /*   Reserved Bits      */
            _U1 MSGLOST:1;                  /*   MSGLOST[2:2]       */
            _U1 INVALDATA:1;                /*   INVALDATA[1:1]     */
            _U1 NEWDATA:1;                  /*   NEWDATA[0:0]       */
        }BIT_RECEIVE;                       /*                      */
    }C0MCTL32;                              /*                      */
```

(5) CiCLKR and CiBCR Registers

To accommodate both byte access to the CiCLKR register and longword access to the concatenated CiBCR and CiCLKR registers, the CiCLKR register is defined as 1 byte within the CiBCR register (i = 0 to 4).

- CANi clock select register (CiCLKR)
- CANi bit configuration register (CiBCR)

Example: Declaration of the CAN0 bit configuration register (C0BCR).

```
    union                               /*                          */
    {                                   /* C0BCR                    */
        _U4 LONG;                       /*  Long Access             */
        struct                          /*                          */
        {                               /*  Bit Access              */
            _U4 TSEG1:4;                /*   TSEG1[31:28]           */
            _U4 reserved1:2;            /*   Reserved Bits          */
            _U4 BRP:10;                 /*   BRP[25:16]             */
            _U4 reserved2:2;            /*   Reserved Bits          */
            _U4 SJW:2;                  /*   SJW[13:12]             */
            _U4 reserved3:1;            /*   Reserved Bits          */
            _U4 TSEG2:3;                /*   TSEG2[10:8]            */
            _U4 reserved4:7;            /*   Reserved Bits          */
            _U4 CCLKS:1;                /*   CCLKS[0:0]             */
        }BIT;                           /*                          */
        struct                          /*                          */
        {                               /*  Bit Access              */
            _U1 reserved1;              /*   Reserved Bits          */
            _U1 reserved2;              /*   Reserved Bits          */
            _U1 reserved3;              /*   Reserved Bits          */
            _U1 C0CLKR;                 /*   C0LKR[7:0]             */
        }BYTE;                          /*                          */
    }C0BCR;                             /*                          */
```

(6) CiMBj Register

As the CiMBj register is defined as a structure consisting of multiple concatenated registers, each register symbol within each mailbox register is defined with an individual name (i = 0 to 4; j = 0 to 63). In addition, 64 mailboxes are defined as an array for each CAN channel. Use the element number corresponding to j in the CiMBj register when referencing the array. To access the DLC bit in C0MB0 register for instance, use CAN0.C0MB[0].BIT.DLC.

- CANi mailbox register j (CiMBj)

Example: Declaration of the CAN0 mailbox registers 0 to 63 (C0MB0 to C0MB63).

```
    union                           /*                      */
    {                               /* C0MB                 */
        _U1 cmbx[16];               /*                      */
        struct                      /*                      */
        {                           /*                      */
            _U4 CMBID;              /*   CMBID              */
            _U4 CMB_1;              /*   CMB_1              */
            _U4 CMB_2;              /*   CMB_2              */
            _U4 CMB_3;              /*   CMB_3              */
        }LONG;                      /*                      */
        struct                      /*                      */
        {                           /*                      */
            _U2 CMBID_0;            /*   CMBID_0            */
            _U2 CMBID_1;            /*   CMBID_1            */
            _U2 CMBDLC;             /*   CMBDLC             */
            _U2 DATA0;              /*   DATA0              */
            _U2 DATA2;              /*   DATA2              */
            _U2 DATA4;              /*   DATA4              */
            _U2 DATA6;              /*   DATA6              */
            _U2 TSP;                /*   TSP                */
        }WORD;                      /*                      */
        struct                      /*                      */
        {                           /*                      */
            _U1 MID0;               /*   MID0               */
            _U1 MID1;               /*   MID1               */
            _U1 MID2;               /*   MID2               */
            _U1 MID3;               /*   MID3               */
            _U1 RESERVED1;          /*   RESERVED1          */
            _U1 DLC;                /*   DLC                */
            _U1 DATA0;              /*   DATA0              */
            _U1 DATA1;              /*   DATA1              */
            _U1 DATA2;              /*   DATA2              */
            _U1 DATA3;              /*   DATA3              */
            _U1 DATA4;              /*   DATA4              */
            _U1 DATA5;              /*   DATA5              */
            _U1 DATA6;              /*   DATA6              */
            _U1 DATA7;              /*   DATA7              */
            _U1 TSH;                /*   TSH                */
            _U1 TSL;                /*   TSL                */
        }BYTE;                      /*                      */
        struct                      /*                      */
        {                           /*                      */
            _U4 IDE:1;              /*   IDE[31:31]         */
            _U4 RTR:1;              /*   RTR[30:30]         */
            _U4 reserved1:1;        /*   Reserved Bits      */
            _U4 SID:11;             /*   SID[28:18]         */
            _U4 EID:18;             /*   EID[17:0]          */
            _U4 reserved2:12;       /*   Reserved Bits      */
            _U4 DLC:4;              /*   DLC[19:16]         */
            _U4 DATA0:8;            /*   DATA0[15:8]        */
            _U4 DATA1:8;            /*   DATA1[7:0]         */
            _U4 DATA2:8;            /*   DATA2[31:24]       */
            _U4 DATA3:8;            /*   DATA3[23:16]       */
            _U4 DATA4:8;            /*   DATA4[15:8]        */
            _U4 DATA5:8;            /*   DATA5[7:0]         */
            _U4 DATA6:8;            /*   DATA6[31:24]       */
            _U4 DATA7:8;            /*   DATA7[23:16]       */
            _U4 TSH:8;              /*   TSH[15:8]          */
            _U4 TSL:8;              /*   TSL[7:0]           */
        }BIT;                       /*                      */
    }C0MB[64];                      /*                      */
```

(7) Registers Defined as Arrays

The following registers are defined as unions consisting of arrays of consecutive registers. Note that the array element numbers do not match the target register numbers. When referencing the array, use the register number – 1 as the element number. To access the PDIRTA1 register for instance, use PDA.PDIRTA[0].BYTE.

- PDAC modulation A rise output time registers 1 to 120 (PDIRTA1 to PDIRTA120)
- PDAC modulation A fall output time registers 1 to 120 (PDIFTA1 to PDIFTA120)
- PDAC modulation B rise output time registers 1 to 200 (PDIRTB1 to PDIRTB200)
- PDAC modulation B fall output time registers 1 to 200 (PDIFTB1 to PDIFTB200)
- PDAC modulation C rise output time registers 1 to 600 (PDIRTC1 to PDIRTC600)
- PDAC modulation C fall output time registers 1 to 600 (PDIFTC1 to PDIFTC600)

Example: Declaration of the PDAC modulation A rise output time registers 1 to 120 (PDIRTA1 to DIRTA120).

```
    union                               /*                          */
    {                                    /* PDIRTA                  */
        _U1 BYTE;                        /*  Byte Access            */
        struct                           /*                         */
        {                                /*  Bit Access             */
            _U1 RTA:8;                   /*   RTA[7:0]              */
        }BIT;                            /*                         */
    }PDIRTA[120];                        /*                         */
```

- FlexRay write data section registers 1 to 64 (FRWRDS1 to FRWRDS64)
- FlexRay read data section registers 1 to 64 (FRRDDS1 to FRRDDS64)

Note that the array element numbers do not match the target register numbers. When referencing the array, use the register number – 1 as the element number. To access the FRWRDS1 register for instance, use FR.FRWRDS[0].LONG.

(8) Dummy Access Areas

To accommodate byte, word, and longword accesses to the dummy access areas, variables are defined with the names BYTE, WORD, and LONG.

- Dummy access area (DUMMYHPB1): H'FFA0 0000 to H'FFA0 0003
- Dummy access area (DUMMYHPB0): H'FFFF 5020 to H'FFFF 5023

Example: Declaration of the dummy access areas (DUMMYHPB0 and DUMMYHPB1).

```
union                               /*                          */
    {                                /* DUMMYHPB1               */
        _U4 LONG;                    /*  Long Access            */
        _U2 WORD;                    /*  Word Access            */
        _U1 BYTE;                    /*  Byte Access            */
    }DUMMYHPB1;                      /*                         */

AND

    union                           /*                          */
    {                               /* DUMMYHPB0               */
        _U4 LONG;                   /*  Long Access            */
        _U2 WORD;                   /*  Word Access            */
        _U1 BYTE;                   /*  Byte Access            */
    }DUMMYHPB0;                     /*                         */
```

For details of the special definitions for each register, refer to the SFR header file.

3. Reference Documents

- Hardware manual
  SH7450 Group, SH7451 Group Hardware Manual Rev.1.00 (REJ09B0454-0100)
  The latest version can be downloaded from the Renesas Electronics website.

- Software manual
  SH-4A Extended Function Software Manual (RJJ09B0235-0100)
  The latest version can be downloaded from the Renesas Electronics website.

Website and Support

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

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

REVISION HISTORY	SH7450 Group/SH7451 Group Register Definition Header File

Rev.	Date	Description	
		Page	Summary
1.00	Oct 26, 2009	-	First edition issued
1.01	Jul 09, 2010	-	Modified SFR header file (added MDCR register)
			Integrated Japanese application note style
1.02	Sep 01, 2010	-	Modified document style and page numbers
			Changed document number from "REJ06B0941-0101" to "R01AN0190EJ0102"

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

— The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

— The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

— The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

— When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

— The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

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

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

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

RENESAS

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

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

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

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

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

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

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

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

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

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

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

Colophon 1.0