R20TS0026EJ0100
Rev.1.00
May 16, 2016

[Notes]

RX64M Group

Camera Sensor Solution

RX Driver Package Application

RX64M Group

Demonstration of Camera Function and Audio Replay with the HMI Expansion Board

RX Driver Package Application

Outline

When using a camera sensor solution or a program for demonstrating camera functionality and audio playback functionality with the HMI expansion board provided as RX Driver Package Application for the RX64M group, take note of the problem described in this note regarding the following point.

1. Handling of flags by functions pdc_frame_end() and pdc_error_interrupt() in the application program

1. Handling of Flags by Functions pdc_frame_end() and pdc_error_interrupt() in the Application Program

1.1 Applicable Products

- RX64M group camera sensor solution RX Driver Package Application

  Applicable revision: Rev1.00

  The following document is relevant.

  • RX64M Group Application Note
  Sensor solution which utilizes a camera
  RX Driver Package Application

  Document no.: R01AN2462EJ0100

- RX64M group demonstration of camera function and audio replay with the HMI expansion board
  RX Driver Package Application

  Applicable revision: Rev1.00

  The following document is relevant.

  • RX64M Group Application Note
  Camera function and sound play function demonstration using the HMI expansion board
  RX Driver Package Application

  Document no.: R01AN2609EJ0100

1.2 Applicable MCUs

RX64M group

1.3 Condition

The execution of R_pdc_init() registers pdc_frame_end() and pdc_error_interrupt() as callback functions for response to generation of the corresponding interrupts.

1.4 Details

Since there is an error in the code for handling flags in pdc_frame_end() and pdc_error_interrupt() in the source code file r_pdc_apl.c, the following faults occur and processing by the application program is not executed properly.

- Faults
  - If an overrun error, underrun error, vertical number of lines setting error, or horizontal number of bytes setting error occurs at the time of frame end interrupt generation, any of the other error flags if set is cleared when the frame end flag is cleared. Thus, the required error handling does not proceed.
  - If more than one error among an overrun error, underrun error, vertical number of lines setting error, or horizontal number of bytes setting error occurs, any other error flag that is set is cleared with the first error flag. Again, the required error handling will not proceed.

1.5 Workaround

Correct the handling of flags by functions pdc_frame_end() and pdc_error_interrupt() in source code file r_pdc_apl.c.

The details of the modification are as follows. Modify the handling in blue to the handling in red for each function.

Remark: As a workaround, newly declare the argument specified for handling which executes the R_PDC_Control function to issue the status clearing command. Also, set the flags which are cleared by calling the R_PDC_Control function as “true”, and the flags which are not cleared as “false”.

- Function pdc_frame_end() of application program file r_pdc_apl.c

Before modification: Flag handling (2 parts) extracted

```
static void pdc_frame_end(void)
{
    volatile pdc_return_t ret_pdc;
    pdc_data_cfg_t p_dummy;
    pdc_stat_t p_stat_pdc;
//Omit
        if (true == p_stat_pdc.pcsr_stat.underrun)
        {
            p_stat_pdc.pcsr_stat.fifo_empty = true;
            ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &p_stat_pdc);
            if (PDC_SUCCESS != ret_pdc)
//Omit
    /* Clear FIFO empty flag. */
    p_stat_pdc.pcsr_stat.fifo_empty = true;
    ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &p_stat_pdc);
    if (PDC_SUCCESS != ret_pdc)
```

After modification:

```
static void pdc_frame_end(void)
{
    volatile pdc_return_t ret_pdc;
    pdc_data_cfg_t p_dummy;
    pdc_stat_t p_stat_pdc;
    pdc_stat_t clear_stat_pdc;
//Omit
        clear_stat_pdc.pcsr_stat.frame_busy = false;
        clear_stat_pdc.pcsr_stat.fifo_empty = false;
        clear_stat_pdc.pcsr_stat.frame_end = true;
        clear_stat_pdc.pcsr_stat.overrun = false;
        clear_stat_pdc.pcsr_stat.underrun = false;
        clear_stat_pdc.pcsr_stat.verf_error = false;
        clear_stat_pdc.pcsr_stat.herf_error = false;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy,
        &clear_stat_pdc);
        if (PDC_SUCCESS != ret_pdc)
//Omit
    /* Clear FIFO empty flag. */
    clear_stat_pdc.pcsr_stat.frame_busy = false;
    clear_stat_pdc.pcsr_stat.fifo_empty = false;
    clear_stat_pdc.pcsr_stat.frame_end = true;
    clear_stat_pdc.pcsr_stat.overrun = false;
    clear_stat_pdc.pcsr_stat.underrun = false;
    clear_stat_pdc.pcsr_stat.verf_error = false;
    clear_stat_pdc.pcsr_stat.herf_error = false;
    ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &clear_stat_pdc);
    if (PDC_SUCCESS != ret_pdc)
```

■ Function pdc_error_interrupt () of application program file r_pdc_apl.c

Before modification: Flag handling (4 parts) extracted

```
static void pdc_error_interrupt(void)
{
    volatile pdc_return_t ret_pdc;
    pdc_data_cfg_t p_dummy;
    pdc_stat_t p_stat_pdc;
//Omit
        /* Clear Overrun flag. */
        p_stat_pdc.pcsr_stat.overrun = true;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &p_stat_pdc);
        if (PDC_SUCCESS != ret_pdc)
//Omit
        /* Clear Underrun flag. */
        p_stat_pdc.pcsr_stat.underrun = true;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &p_stat_pdc);
        if (PDC_SUCCESS != ret_pdc)
//Omit
        /* Clear Vertical line setting error flag. */
        p_stat_pdc.pcsr_stat.verf_error = true;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &p_stat_pdc);
        if (PDC_SUCCESS != ret_pdc)
//Omit
        /* Clear Horizontal byte setting error flag. */
        p_stat_pdc.pcsr_stat.herf_error = true;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &p_stat_pdc);
        if (PDC_SUCCESS != ret_pdc)
```

After modification:

```
static void pdc_error_interrupt(void)
{
    volatile pdc_return_t ret_pdc;
    pdc_data_cfg_t p_dummy;
    pdc_stat_t p_stat_pdc;
    pdc_stat_t clear_stat_pdc;
//Omit
        /* Clear Overrun flag. */
        clear_stat_pdc.pcsr_stat.frame_busy = false;
        clear_stat_pdc.pcsr_stat.fifo_empty = false;
        clear_stat_pdc.pcsr_stat.frame_end = false;
        clear_stat_pdc.pcsr_stat.overrun = true;
        clear_stat_pdc.pcsr_stat.underrun = false;
        clear_stat_pdc.pcsr_stat.verf_error = false;
        clear_stat_pdc.pcsr_stat.herf_error = false;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &clear_stat_pdc);
        if (PDC_SUCCESS != ret_pdc)
//Omit
        /* Clear Underrun flag. */
        clear_stat_pdc.pcsr_stat.frame_busy = false;
        clear_stat_pdc.pcsr_stat.fifo_empty = false;
        clear_stat_pdc.pcsr_stat.frame_end = false;
        clear_stat_pdc.pcsr_stat.overrun = false;
        clear_stat_pdc.pcsr_stat.underrun = true;
        clear_stat_pdc.pcsr_stat.verf_error = false;
        clear_stat_pdc.pcsr_stat.herf_error = false;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &clear_stat_pdc);
        if (PDC_SUCCESS != ret_pdc)
//Omit
        /* Clear Vertical line setting error flag. */
        clear_stat_pdc.pcsr_stat.frame_busy = false;
        clear_stat_pdc.pcsr_stat.fifo_empty = false;
        clear_stat_pdc.pcsr_stat.frame_end = false;
        clear_stat_pdc.pcsr_stat.overrun = false;
        clear_stat_pdc.pcsr_stat.underrun = false;
        clear_stat_pdc.pcsr_stat.verf_error = true;
        clear_stat_pdc.pcsr_stat.herf_error = false;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &clear_stat_pdc);
```

```
        if (PDC_SUCCESS != ret_pdc)
//Omit
        /* Clear Horizontal byte setting error flag. */
        clear_stat_pdc.pcsr_stat.frame_busy = false;
        clear_stat_pdc.pcsr_stat.fifo_empty = false;
        clear_stat_pdc.pcsr_stat.frame_end = false;
        clear_stat_pdc.pcsr_stat.overrun = false;
        clear_stat_pdc.pcsr_stat.underrun = false;
        clear_stat_pdc.pcsr_stat.verf_error = false;
        clear_stat_pdc.pcsr_stat.herf_error = true;
        ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &p_dummy, &clear_stat_pdc);
        if (PDC_SUCCESS != ret_pdc)
```

1.6 Schedule for Fixing the Problem

- RX64M group camera sensor solution RX Driver Package Application

  This problem will be fixed in the next version.

- RX64M group demonstration of camera function and audio replay with the HMI extension board RX Driver Package Application

  This problem will be fixed in the next version.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May 16, 2016	-	First edition issued

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061 Japan
Renesas Electronics Corporation

■Inquiry
http://www.renesas.com/en-hq/support/contact.html