RZ/A1H Group

Pixel Format Converter "PFV" Driver Example

R01AN1880EJ0103
Rev.1.03
Aug.24, 2017

Introduction

This application note describes the example driver which converts an image from a pixel format to another by using the RZ/A1H's pixel format converter (PFV).

The PFV example driver offers the following features:

- Converts image data among pixel formats RGB888(input only), ARGB8888(output only), RGB565, and YCbCr422.
- Uses a color matrix which allows a brightness (offset) adjustment and a 9-axis gain adjustment on image data when converting it.
- Uses two direct access controllers (DMACs) for consecutively converting image data. One DMAC inputs image data from RAM into the PFV. Another DMAC transfers image data from the PFV to RAM. The DMAC channel is specified in the initialization function. Interrupts (IFEI and OFFI) generated from the PFV are reported to the DMACs.
- PFV and DMAC channel number can be changed to any number in PFV example application.

Target Device

RZ/A1H Group

RZ/A1M Group

When applying the example program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Table of Contents

1. Specifications

Table 1-1 shows the Peripheral Functions and Their Applications, and Figure 1-1 shows the Operation Overview.

Table 1-1 Peripheral Functions and Their Applications

Peripheral functions	Uses
Pixel Format Converter (PFV)	Converts image data.
Direct Access Controller (DMAC)	Transfers image data. There are one DMAC for input to the PFV and another DMAC for output from the PFV.
Interrupt Controller (INTC)	Controls the PFV and DMAC interrupts.
Serial Communications Interface (SCIF)(UART)	Debug output

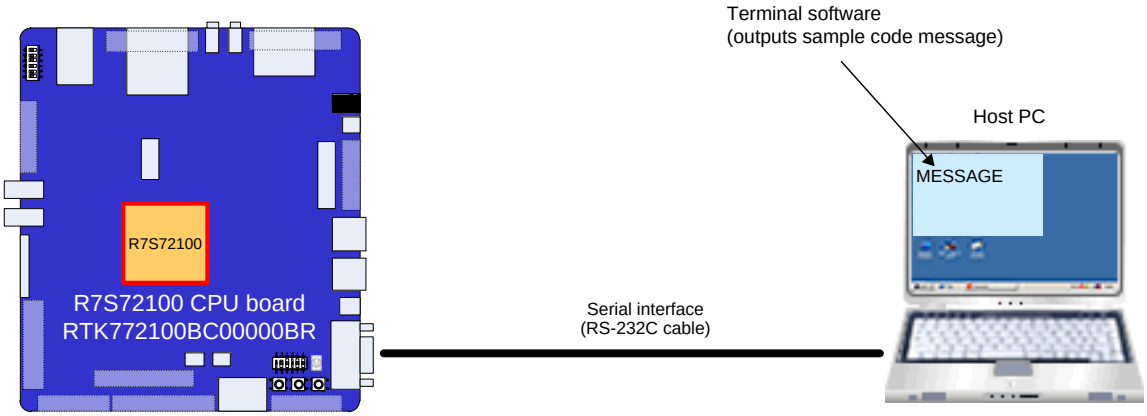


Figure 1-1 the Operation Overview

2. Operation Check Conditions

The example code contained in this application note has been checked under the conditions listed below.

Table 2-1 Operation Check Conditions

Item		Description
MCU used		RZ/A1H
Operating frequency		CPU clock (Iϕ): 400MHz Image processing clock (Gϕ): 266.67MHz Internal bus clock (Bϕ): 133.33MHz Peripheral clock 1 (P1ϕ): 66.67MHz Peripheral clock 0 (P0ϕ): 33.33MHz
Operating voltage		Power supply voltage (I/O): 3.3V Power supply voltage (Internal): 1.18V
ARM	Integrated development environment	ARM® integrated development environment ARM Development Studio 5 (DS-5™) Version 5.16
	C compiler	ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]
IAR	Integrated development environment	IAR Embedded Workbench for ARM 7.80.4.12495
	C compiler	
Renesas	Integrated development environment	e2 studio (Version: 5.3.0.023)
	C compiler	GNUARM-NONE-EABI v16.01
Operating mode		Boot mode 0 (CS0 space 16bit boot)
Communication setting of terminal software		• Communication speed: 115200bps • Data length: 8 bits • Parity: None • Stop bit length: 1 bit • Flow control: None
Board used		GENMAI board • RTK772100BC00000BR (R7S72100 CPU board) • RTK77210000B00000BR (R7S72100 Option board)
Device used		LCD. (Only the example should be used. The PFV driver should not be used.) Serial interface (D-sub 9-pin connector)

3. Reference Application Note(s)

For additional information associated with this document, refer to the following application note(s).

- RZ/A1H Group Example of Initialization (R01AN1646EJ)
- RZ/A1H Group I/O definition header file <iodefine.h> (R01AN1860EJ)
- RZ/A1H Group OS porting layer "OSPL" Example Program (R01AN1887EJ)
- RZ/A1H Group Direct Access Controller "DMAC_RM" Driver Example (Attached to PFV) (R01AN1888EJ)

4. Peripheral Functions

The basic functions of the PFV and DMAC are described in the RZ/A1H Group User's Manual: Hardware.

5. Description of Hardware

5.1 Hardware Configuration

Figure 5-1 shows examples of hardware devices connected. Figure 5-2 shows Block Diagram.

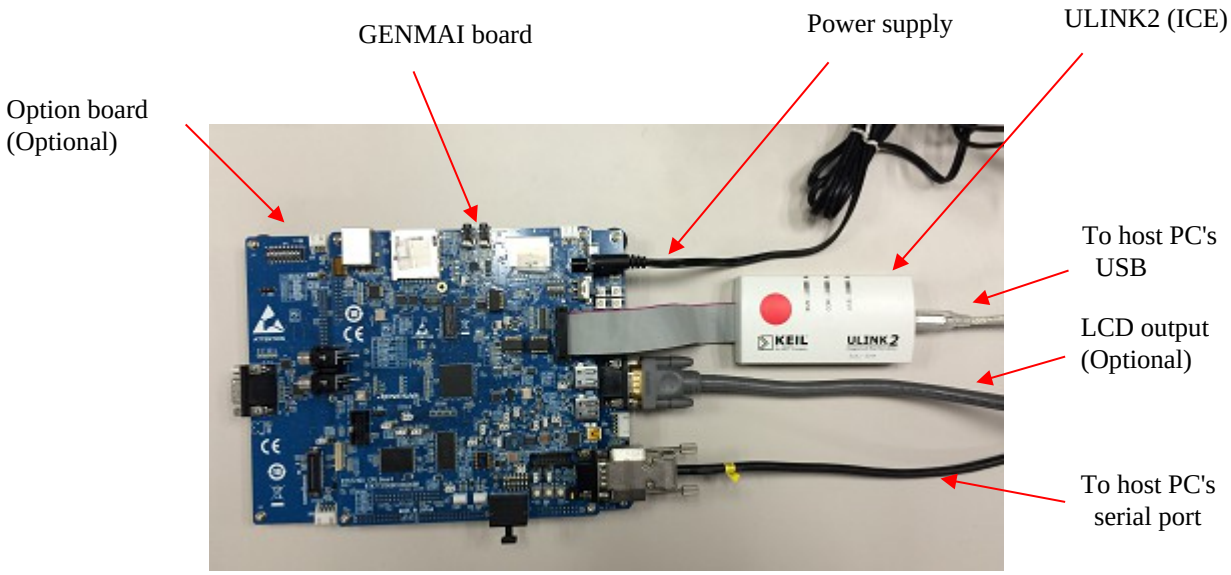
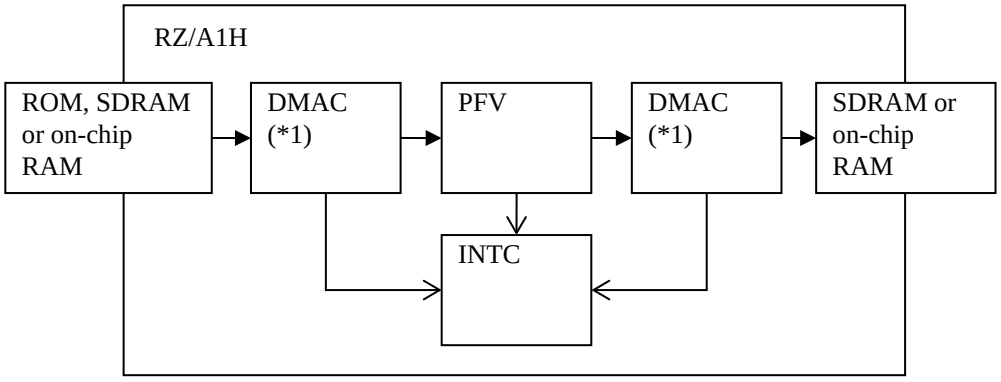


Figure 5-1 Examples of Hardware Devices Connected



(*1) CPU can access to PFV instead of DMAC.

Figure 5-2 Block Diagram

5.2 List of Pins to be Used

Table 5-1 lists the pins to be used and their functions.

Table 5-1 Pins to be Used and their Functions

Pin name	I/O	Description
None		

6. Description of Software

6.1 Operation Outline

Figure 6-1 shows the sequence of processes using the DMACs. Figure 6-2 shows the sequence of processes not using the DMACs.

Changing the OS requires changing the contents of the driver porting layer functions.

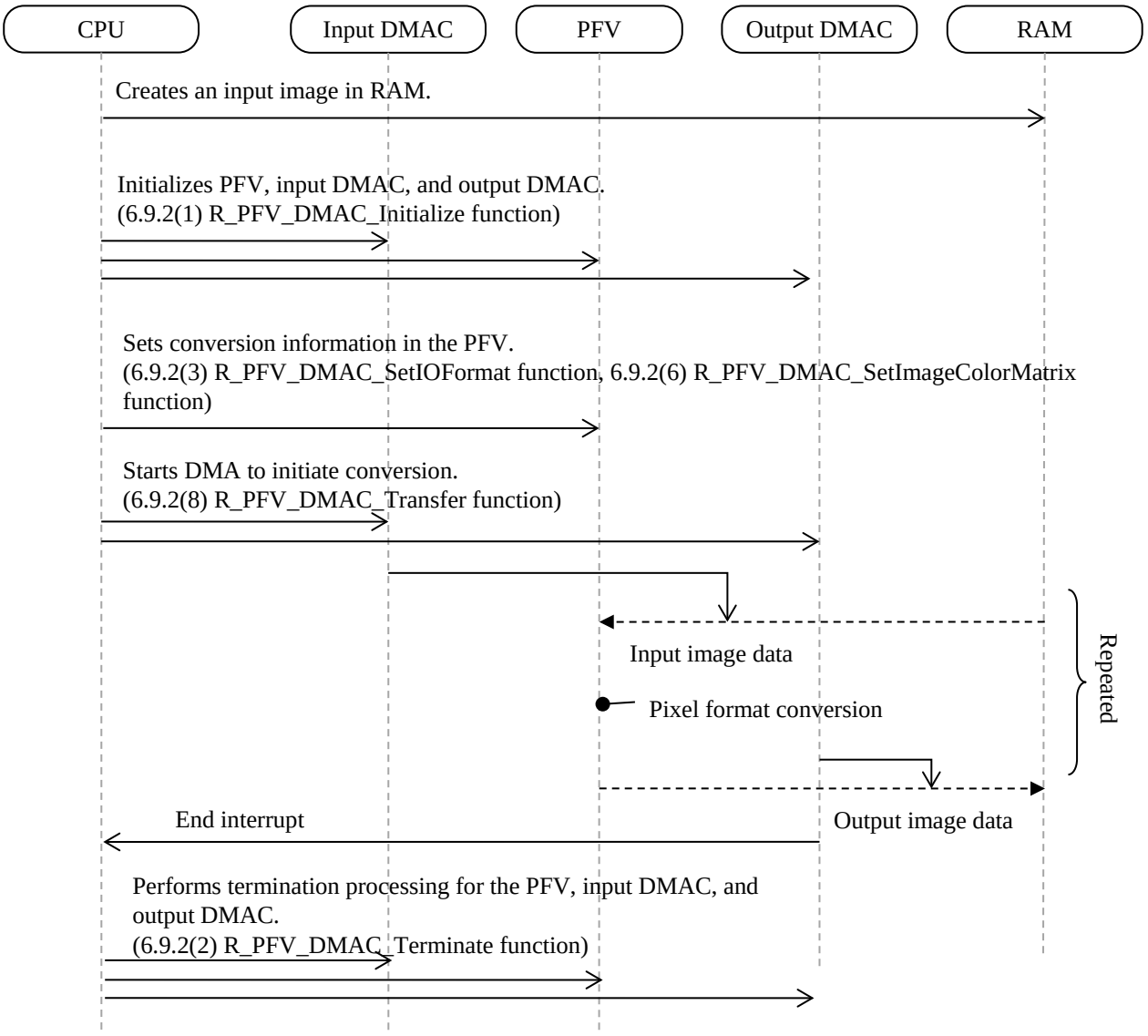


Figure 6-1 Sequence of Processes Using the DMACs

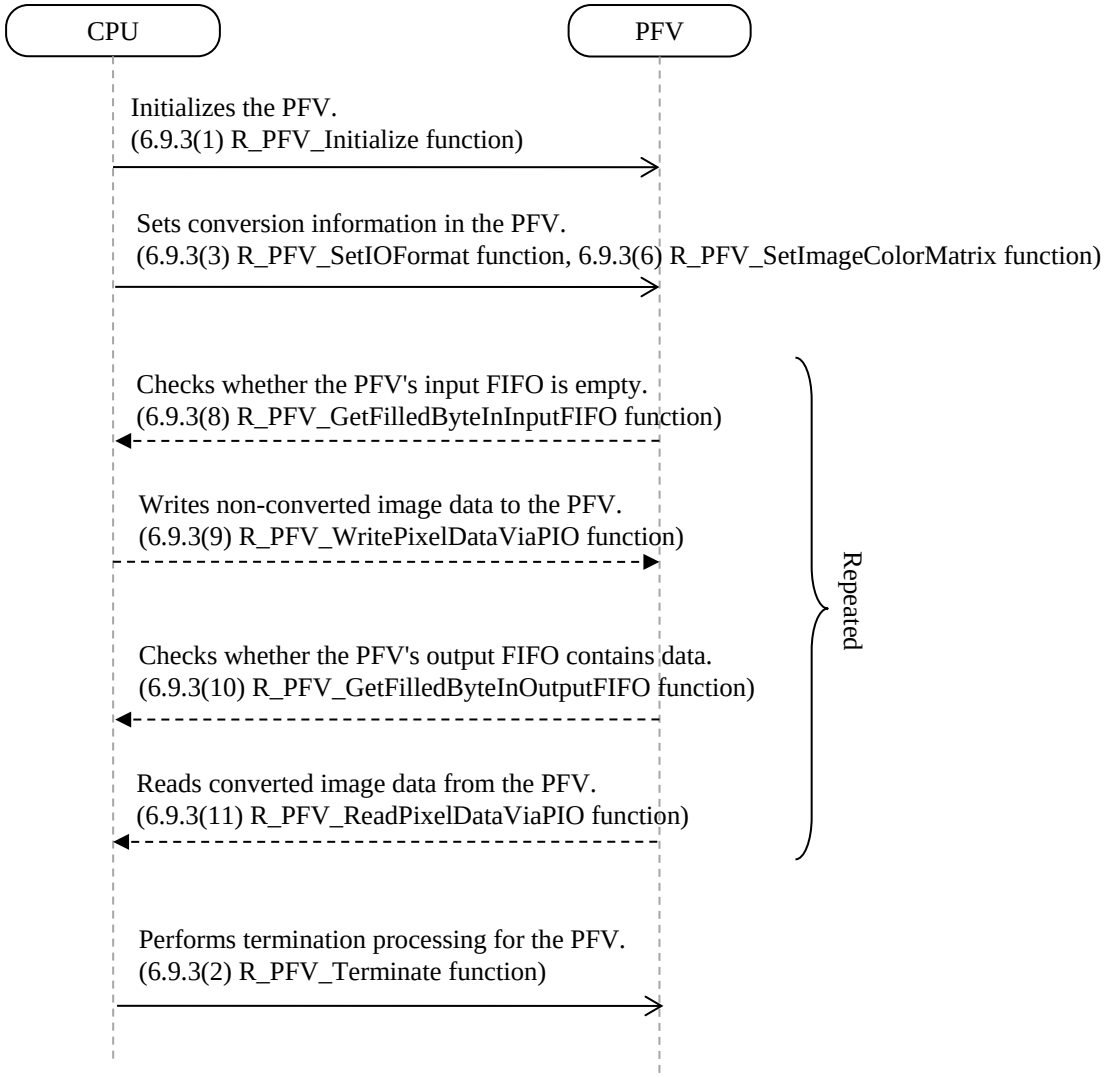
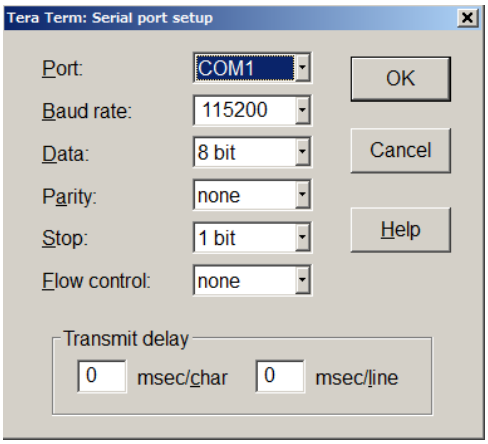


Figure 6-2 Sequence of Processes Not Using the DMACs

6.1.1 Preparations

The following preparations in Sample Code.

1. Terminal software is started in a host PC and it's established as follows. (In the case of Tera Term)



2. When a sample program is executed, a message is output at a terminal as follows.

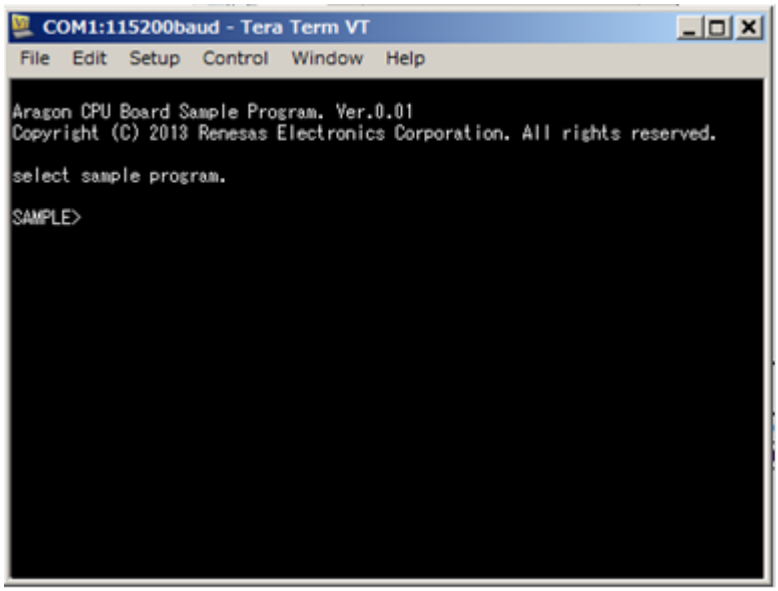


Figure 6-3 Message output at sample program execution

6.2 Memory Mapping

Figure 6.4 shows the Address Space of the RZ/A1H group and the Memory Mapping of the GENMAI board RTK772100BC00000BR.

In this sample code, the code and data used in the ROM area is located in the NOR flash memory connected to the CS0 space, and the code and data used in the RAM area is located in the large-capacity on-chip RAM.

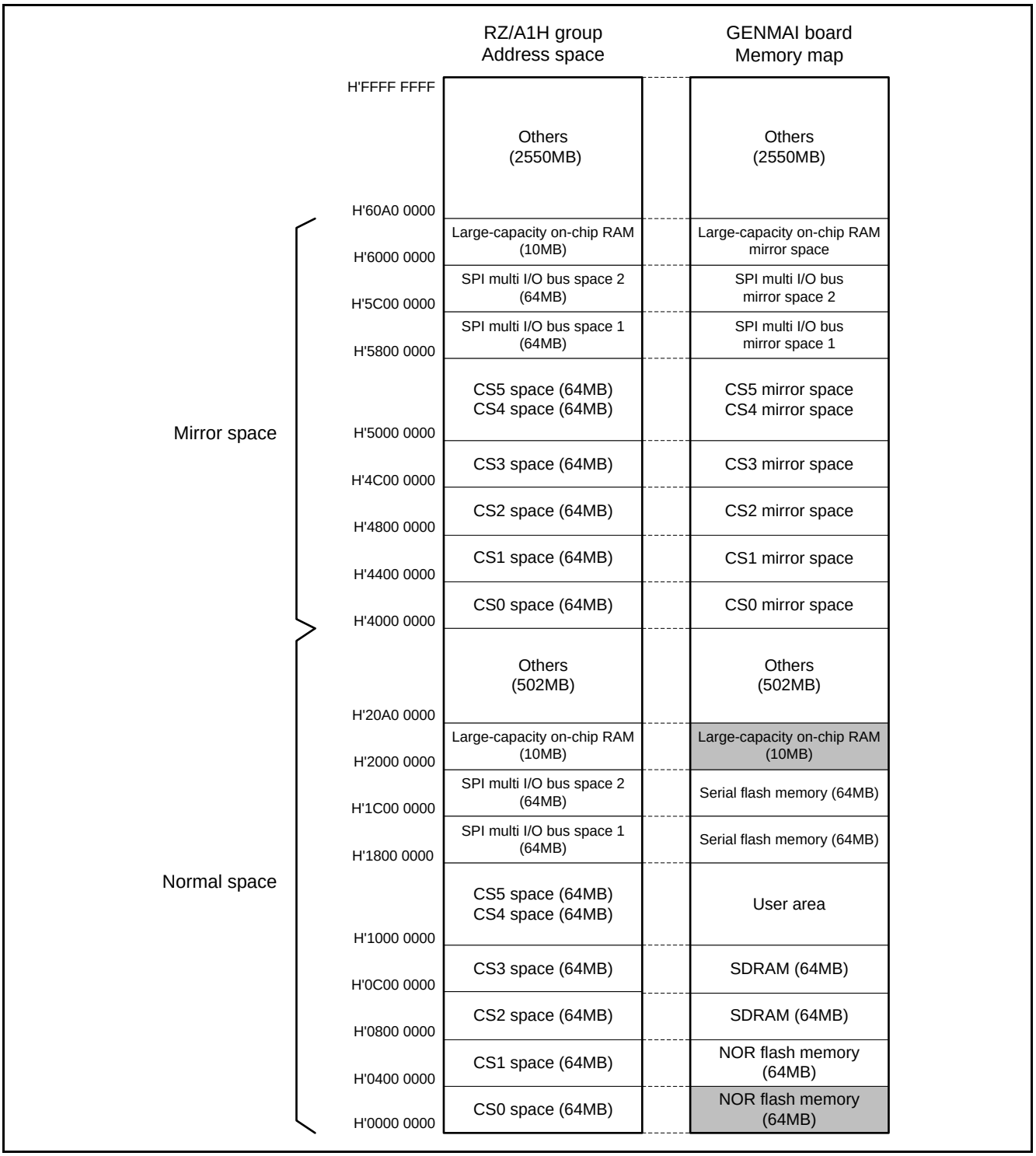


Figure 6.4 Memory Mapping

6.2.1 Section Assignment in Sample Code

In this sample code, the exception processing vector table and the IRQ interrupt handler are assigned to the large-capacity on-chip RAM, and they are executed in such RAM to speed up the interrupt processing. The transfer processing from the NOR flash memory area which is the program code of the exception processing vector table and the IRQ interrupt handler to the large-capacity on-chip RAM area, the clear to zero processing for the data selection without initial data, and the initialization for the data selection with initial data are executed by using the scatter-loading function. Refer to "Image structure and generation" in "ARM Compiler toolchain Using the Linker" provided by the ARM for more information about the scatter-loading function.

Table 6.1 and Table 6.2 list the Sections to be Used in this sample code. Figure 6.5 shows the Section Assignment for the initial condition of the sample code and the condition after using the scatter-loading function.

Table 6.1 Sections to be Used (1/2)

Area Name	Description	Type	Loading Area	Execution Area
VECTOR_TABLE	Exception processing vector table	Code	FLASH	FLASH
RESET_HANDLER	Program code area of reset handler processing This area consists of the following sections. • INITCA9CACHE (L1 cache setting) • INIT_TTB (MMU setting) • RESET_HANDLER (Reset handler)	Code	FLASH	FLASH
CODE_BASIC_SETUP	Program code area to optimize operating frequency and flash memory	Code	FLASH	FLASH
InRoot	This area consists of the sections located in the root area such as C standard library.	Code and RO Data	FLASH	FLASH
CODE_FPU_INIT	Program code area for NEON and VFP initializations This area consists of the following sections. • CODE_FPU_INIT • FPU_INIT	Code	FLASH	FLASH
CODE_RESET	Program code area for hardware initialization This area consists of the following sections. • CODE_RESET (Startup processing) • INIT_VBAR (Vector base setting)	Code	FLASH	FLASH
CODE_IO_REGRW	Program code area for read/write functions of I/O register	Code	FLASH	FLASH
CODE	Program code area for defaults All the Code type sections which do not define section names with C source are assigned in this area.	Code	FLASH	FLASH
CONST	Constant data area for defaults All the RO Data type sections which do not define section names with C source are assigned in this area.	RO Data	FLASH	FLASH

Table 6.2 Sections to be Used (2/2)

Area Name	Description	Type	Loading Area	Execution Area
VECTOR_MIRROR_TABLE	Exception processing vector table (Section to transfer data to large-capacity on-chip RAM)	Code	FLASH	LRAM
CODE_HANDLER_JMPTBL	Program code area for user-defined functions of IRQ interrupt handler	Code	FLASH	LRAM
CODE_HANDLER	Program code area of IRQ interrupt handler This area consists of the following sections. • CODE_HANDLER • IRQ_FIQ_HANDLER	Code	FLASH	LRAM
DATA_HANDLER_JMPTBL	Registration table data area for user-defined functions of IRQ interrupt handler	RW Data	FLASH	LRAM
ARM_LIB_STACK	Application stack area	ZI Data	-	LRAM
IRQ_STACK	IRQ mode stack area	ZI Data	-	LRAM
FIQ_STACK	FIQ mode stack area	ZI Data	-	LRAM
SVC_STACK	Supervisor (SVC) mode stack area	ZI Data	-	LRAM
ABT_STACK	Abort (ABT) mode stack area	ZI Data	-	LRAM
TTB	MMU translation table area	ZI Data	-	LRAM
ARM_LIB_HEAP	Application heap area	ZI Data	-	LRAM
DATA	Data area with initial value for defaults All the RW Data type sections which do not define section names with C source are assigned in this area.	RW Data	FLASH	LRAM
BSS	Data area without initial value for defaults All the ZI Data type sections which do not define section names with C source area assigned in this area.	ZI Data	-	LRAM

Notes: 1. "FLASH" and "LRAM" shown in Loading Area and Execution Area indicate the NOR flash memory area and the large-capacity on-chip RAM area respectively.

2. Basically the section name is set to be the same as the region's, however it consists of some sections in the areas of RESET_HANDLER, InRoot, CODE_FPU_INIT, CODE_RESET, CODE, CONST, CODE_HANDLER, DATA, and BSS. Refer to the ARM compiler toolchain manual about the region and the section.

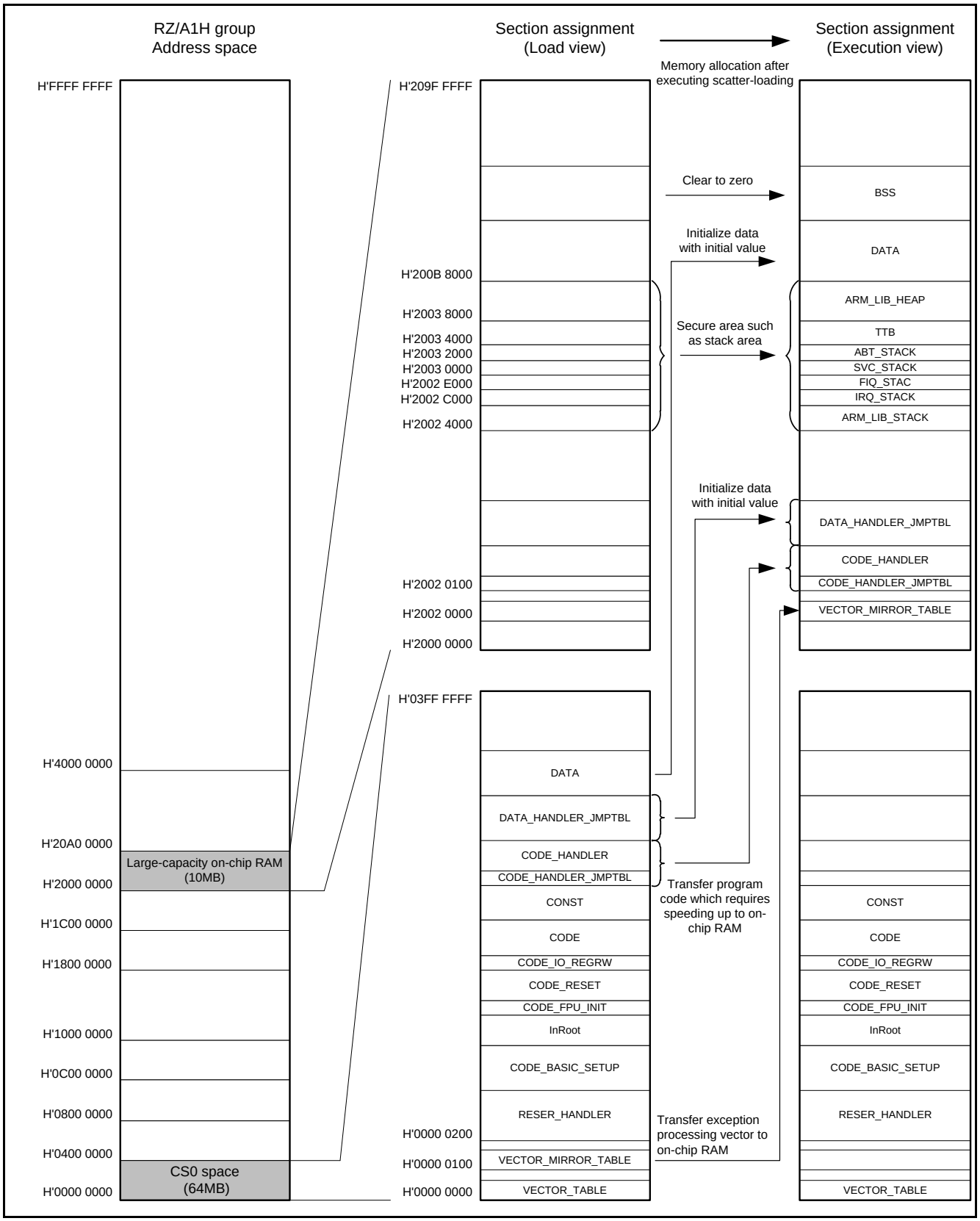


Figure 6.5 Section Assignment

6.2.2 Setting for MMU

The MMU is set to manage the 4 GB area in 1MB unit from the address H'0000 0000 in response to the memory map of the hardware resource used for the **GENMAI board**. (Set by the ttb_init.s file.) The minimum unit should be 1MB when customizing the MMU based on the system.

Table 6.3 lists the Setting for MMU.

Table 6.3 Setting for MMU

Definition Name	Contents	Address	Size	Memory Type
M_SIZE_NOR	CS0 and CS1 spaces (NOR flash memory)	H'0000 0000 to H'07FF FFFF	128MB	L1 cache enable, Normal memory
M_SIZE_SDRAM	CS2 and CS3 spaces (SDRAM)	H'0800 0000 to H'0FFF FFFF	128MB	L1 cache enable, Normal memory
M_SIZE_CS45	CS4 and CS5 spaces	H'1000 0000 to H'17FF FFFF	128MB	Strongly-ordered memory (L1 cache disable)
M_SIZE_SPI	SPI multi IO bus space 1 and 2 (serial flash memory)	H'1800 0000 to H'1FFF FFFF	128MB	L1 cache enable, Normal memory
M_SIZE_RAM	Large-capacity on-chip RAM space	H'2000 0000 to H'209F FFFF	10MB	L1 cache enable, Normal memory
M_SIZE_IO_1	On-chip peripheral module and reserved area	H'20A0 0000 to H'3FFF FFFF	502MB	Strongly-ordered memory (L1 cache disable)
M_SIZE_NOR_M	CS0 and CS1 mirror spaces	H'4000 0000 to H'47FF FFFF	128MB	L1 cache disable, Normal memory
M_SIZE_SDRAM_M	CS2 and CS3 mirror spaces	H'4800 0000 to H'4FFF FFFF	128MB	L1 cache disable, Normal memory
M_SIZE_CS45_M	CS4 and CS5 mirror spaces	H'5000 0000 to H'57FF FFFF	128MB	Strongly-ordered memory (L1 cache disable)
M_SIZE_SPI_M	SPI multi IO bus mirror space 1 and 2	H'5800 0000 to H'5FFF FFFF	128MB	L1 cache disable, Normal memory
M_SIZE_RAM_M	Large-capacity on-chip RAM mirror space	H'6000 0000 to H'609F FFFF	10MB	L1 cache disable, Normal memory
M_SIZE_IO_2	On-chip peripheral module and reserved area	H'60A0 0000 to H'FFFF FFFF	2550MB	Strongly-ordered memory (L1 cache disable)

6.2.3 Exception Processing Vector Table

The RZ/A1H has seven types of exception processing (reset, undefined instruction, software interrupt, prefetch abort, data abort, IRQ, and FIQ). In the case of boot mode 0, the exception processing vector table is assigned to the area from H'0000 0000 to the area of 32 bytes (from H'0000 0000 to H'0000 001F) after the reset cancellation.

Figure 6.6 shows the contents of the sample code exception processing vector table as a description example.

```
vector_table
  LDR pc, =reset_handler       ; 0x0000_0000 : Reset exception
  LDR pc, =undefined_handler   ; 0x0000_0004 : Undefined instructions
exception
  LDR pc, =svc_handler         ; 0x0000_0008 : Software interrupts exceptions
  LDR pc, =prefetch_handler    ; 0x0000_000c : Prefetch abort exception
  LDR pc, =abort_handler       ; 0x0000_0010 : Data abort exception
  LDR pc, =reserved_handler    ; 0x0000_0014 : Reserved
  LDR pc, =irq_handler         ; 0x0000_0018 : IRQ exception
  LDR pc, =fiq_handler         ; 0x0000_001c : FIQ exception
```

Figure 6.6 Description Example of Exception Processing Vector Table

6.3 List of commands

Table 6-4 shows commands of sample program. Command name is not case sensitive.

Table 6-4 List of commands

Command	Summary
Sample_PFV_PIO	This sample code directly outputs pixel data from the CPU to the PFV's FIFO. It transfers pixel data from memory to memory.
Sample_PFV_DMAC	This sample code outputs pixel data from the DMAC to the PFV's FIFO. It transfers pixel data from memory to memory.
Sample_PFV_DMAC_Image	This sample code outputs pixel data from the DMAC to the PFV's FIFO. It transfers pixel data from memory to memory. Output data is shown in LCD. Program changes gain of image.

6.4 Interrupt

Table 6-5 shows Interrupts using by example code.

Table 6-5 Interrupts using by example code

Interrupt (Source ID)	Priority	Summary
DMAINT0 - DMAINT#	INTERRUPT_LEVEL_OF_INPUT_DMAC(=20) INTERRUPT_LEVEL_OF_OUTPUT_DMAC(=19)	Receives the interrupt of DMAINT#
IFEI0 - IFEI#	INTERRUPT_LEVEL_OF_INPUT_EMPTY(=24)	Receives the interrupt of IFEI# or is the target of DMA Extended Resource Selector.
OFFI0 - OFFI#	INTERRUPT_LEVEL_OF_OUTPUT_FULL(=23)	Receives the interrupt of OFFI# or is the target of DMA Extended Resource Selector.
PFVEI0 - PFVEI#	INTERRUPT_LEVEL_OF_ERROR(=31)	Receives the interrupt of PFVEI#

6.5 Basic Types

Symbol	Description
char_t	8-bit character
bool_t	Logical data type. The value is true (1) or false (0).
int_t	The signed integer for this library is a 32-bit signed integer.
int8_t	8-bit signed integer (defined by standard library)
int16_t	16-bit signed integer (defined by standard library)
int32_t	32-bit signed integer (defined by standard library)
int64_t	64-bit signed integer (defined by standard library)
uint8_t	8-bit unsigned integer (defined by standard library)
uint16_t	16-bit unsigned integer (defined by standard library)
uint32_t	32-bit unsigned integer (defined by standard library)
uint64_t	64-bit unsigned integer (defined by standard library)
int_fast8_t	Fastest 8-bit minimum-width signed integer
int_fast16_t	Fastest 16-bit minimum-width signed integer
int_fast32_t	Fastest 32-bit minimum-width signed integer
uint_fast8_t	Fastest 8-bit minimum-width unsigned integer
uint_fast16_t	Fastest 16-bit minimum-width unsigned integer
uint_fast32_t	Fastest 32-bit minimum-width unsigned integer
uintptr_t	Same as pointer bit width unsigned integer as physical address
size_t	Same as pointer bit width unsigned integer as byte size
ptrdiff_t	Same as pointer bit width signed integer as difference between pointers
bit_flags_fast32_t	Same as uint_fast32_t bit flags (bit field)
bit_flags32_t	Same as uint32_t bit flags (bit field)
float32_t	32-bit float (Defined by standard library when "__ARM_NEON__" defined)
float64_t	64-bit float (Defined by standard library when "__ARM_NEON__" defined)
float128_t	128-bit float

6.6 Constants, Enumerations and Error code

Section	Type Symbol	Description
6.6.1	-	Version
6.6.2	errnum_t	Error Codes
6.6.3	pfv_format_t	Pixel format of image data
6.6.4	pfv_swap_t	Byte-order swapping for image data
6.6.5	pfv_color_matrix_mode_t	Pixel format of image data before and after conversion
6.6.6	pfv_dc_offset_index_t	Array number for an offset component in the color matrix
6.6.7	pfv_matrix_multiply_index_t	Array number for a gain component in the color matrix
6.6.8	pfv_idtrg_t	Number of bytes when the input FIFO is determined to be empty
6.6.9	pfv_odtrg_t	Number of bytes when the output FIFO is determined to be full
6.6.10	pfv_interrupt_line_t	A kind of interrupt line
6.6.11	pfv_interrupt_lines_t	Bit flags of "pfv_interrupt_line_t"
6.6.12	pfv_dmac_interrupt_unit_t	A kind of peripheral signaled interrupt

*1	r_ospl_interrupt_t	Struct of interrupt source
*1	r_ospl_caller_t	Information of driver internal interrupt operations
6.6.13	Unnamed enumerations and constants	Other constants

6.6.1 Version

Symbol	Value	Description
PFV_VERSION	101	Version number of PFV
PFV_VERSION_STRING	"1.01"	String of version number of PFV

6.6.2 Error Codes

Symbol	Value	Description
0	0	No error is detected.
E_OTHERS	1	Others error
E_FEW_ARRAY	2	Error of few fixed length array
E_FEW_MEMORY	3	Few heap memory area
E_FIFO_OVER	4	Failed to enqueue
E_NOT_FOUND_SYMBOL	5	Not defined the symbol
E_NO_NEXT	6	There is not next element of list
E_ACCESS_DENIED	7	Error of denied read or write
E_NOT_IMPLEMENT_YET	9	Not implemented yet
E_ERRNO	0x0E(=14)	Refer to "errno"
E_LIMITATION	0x0F(=15)	Temporary limitation
E_STATE	0x10(=16)	Cannot do at this state
E_NOT_THREAD	0x11(=17)	Not a thread, Cannot call from interrupt context.
E_PATH_NOT_FOUND	0x12(=18)	Not found file or folder
E_BAD_COMMAND_ID	0x16(=22)	Out of number of command ID
E_TIME_OUT	0x17(=23)	Time out
E_STACK_OVERFLOW	0x1C(=28)	Stack overflow
E_NO_DEBUG_TLS	0x1D(=29)	Not set debug work area.
E_EXIT_TEST	0x1E(=30)	Request of exit from the test
E_PFV_HW_ERROR	0x4701(=18177)	Error of detected by PFV hardware

6.6.3 pfv_format_t

Pixel formats of image data

Symbol	Value	Description
PFV_RGB888	0	32-bit color. The bits are in this order: 8 bits unused, 8 bits for red, 8 bits for green, and 8 bits for blue. They are arranged from left to right in descending order of significance or byte order used by the CPU. (This order is used by default for input.)
PFV_ARGB8888	0	32-bit color. The bits are in this order: 8 bits for alpha, 8 bits for red, 8 bits for green, and 8 bits for blue. They are arranged from left to right in descending order of significance or byte order used

*1 RZ/A1H Group OS porting layer "OSPL" Example Program (R01AN1887EJ)

		by the CPU. Output alpha value is fixed value setting at pfv_io_format_t::output_alpha variable. (This order is used by default for output.)
PFV_RGB565	1	16-bit color. The bits are in this order: 5 bits for red, 6 bits for green, and 5 bits for blue. They are arranged from left to right in descending order of significance or byte order used by the CPU.
PFV_YCbCr422	3	32-bit color x 2 pixels. The four bytes are in this order: 1 byte for Cb (color difference U), 1 byte for Y0 (brightness of the pixel on the left), 1 byte for Cr (color difference V), and 1 byte for Y1 (brightness of the pixel on the right). They are ordered from left to right in ascending order of addresses.

6.6.4 pfv_swap_t

Byte-order swapping for image data

Symbol	Value	Description
When the pixel format is PFV_RGB888 or PFV_ARGB8888:		
PFV_SWAP_ARGB8888	0	32-bit color. The bits are in this order: 8 bits for alpha (or 8 bits unused), 8 bits for red, 8 bits for green, and 8 bits for blue. They are arranged from left to right in descending order of significance or byte order used by the CPU. (Default)
PFV_SWAP_RABG8888	1	The bits are in this order: bits for red, bits for alpha (or bits unused), bits for blue, and bits for green. They are arranged from left to right in descending order of significance like the bits for PFV_SWAP_ARGB8888.
PFV_SWAP_GBAR8888	2	The bits are in this order: bits for green, bits for blue, bits for alpha (or bits unused), and bits for red. They are ordered from left to right in descending order of significance like the bits for PFV_SWAP_ARGB8888.
PFV_SWAP_BGRA8888	3	The bits are in this order: bits for blue, bits for green, bits for red, and bits for alpha (or bits unused). They are ordered from left to right in descending order of significance like the bits for PFV_SWAP_ARGB8888.
When the pixel format is PFV_RGB565:		
PFV_SWAP_RGB565_PIXEL10	2	16-bit color x 2 pixels. The high-order 16 bits are for the pixel on the right. The low-order 16 bits are for the pixel on the left. The bits are in this order: 5 bits for red, 6 bits for green, and 5 bits for blue. They are arranged from left to right in descending order of significance or byte order used by the CPU. (Default) Do not set this value when accessing 16 bits in the FIFO.
PFV_SWAP_RGB565_PIXEL01	0	Same as above except that the high-order 16 bits are for the pixel on the left and the low-order 16 bits are for the pixel on the right. Do not set this value when accessing 16 bits in the FIFO.
When the pixel format is PFV_YCbCr422:		
PFV_SWAP_YCbCr422_Cb_Y0_Cr_Y1	3	32-bit color x 2 pixels. The four bytes are in this order: 1 byte for Cb (color difference U), 1 byte for Y0 (brightness of the pixel on the left), 1 byte for Cr (color difference V), and 1 byte for Y1 (brightness of the pixel on the right). They are ordered from left to right in ascending order of addresses. (Default)
PFV_SWAP_YCbCr422_Y0_Cb_Y1_Cr	2	The four bytes are in this order: 1 byte for Y0, 1 byte for Cb (U), 1 byte for Y1, and 1 byte for Cr (V). They are ordered from left to right in ascending order of addresses like the bytes for PFV_SWAP_YCbCr422_Y1CrY0Cb.

PFV_SWAP_YCbCr422_Cr_Y1_Cb_Y0	1	The four bytes are in this order: 1 byte for Cr (V), 1 byte for Y1, 1 byte for Cb (U), and 1 byte for Y0. They are ordered from left to right in ascending order of addresses like the bytes for PFV_SWAP_YCbCr422_Y1CrY0Cb.
PFV_SWAP_YCbCr422_Y1_Cr_Y0_Cb	0	The four bytes are in this order: 1 byte for Y1, 1 byte for Cr (V), 1 byte for Y0, and 1 byte for Cb (U). They are ordered from left to right in ascending order of addresses like the bytes for PFV_SWAP_YCbCr422_Y1CrY0Cb.
PFV_SWAP_YCbCr422_Y1CrY0Cb	3	32-bit color x 2 pixels. The bits are in this order: 8 bits for Y1 (brightness of the pixel on the right), 8 bits for Cr (color difference U), and 8 bits for Cb (color difference V). They are ordered from left to right in descending order of significance or byte order used by the CPU.
PFV_SWAP_YCbCr422_CrY1CbY0	2	The bits are in this order: bits for Cr, bits for Y1, bits for Cb, and bits for Y0. They are ordered from left to right in descending order of significance like the bits for PFV_SWAP_YCbCr422_Y1CrY0Cb.
PFV_SWAP_YCbCr422_Y0CbY1Cr	1	The bits are in this order: bits for Y0, bits for Cb, bits for Y1, and bits for Cr. They are ordered from left to right in descending order of significance like the bits for PFV_SWAP_YCbCr422_Y1CrY0Cb.
PFV_SWAP_YCbCr422_CbY0CrY1	0	The bits are in this order: bits for Cb, bits for Y0, bits for Cr, and bits for Y1. They are ordered from left to right in descending order of significance like the bits for PFV_SWAP_YCbCr422_Y1CrY0Cb.

Refer to Section 47.3.2, Input/Output Data Format of the RZ/A1H Group User's Manual: Hardware.

6.6.5 pfv_color_matrix_mode_t

Type of image data conversion between pixel formats

Refer to Section 6.9.4(1), R_PFV_STATIC_GetColorMatrixMode Function

Symbol	Value	Description
PFV_GBR_TO_GBR	0	Converts image data from RGB to RGB.
PFV_GBR_TO_YCbCr	1	Converts image data from RGB to YCbCr.
PFV_YCbCr_TO_GBR	2	Converts image data from YCbCr to RGB.
PFV_YCbCr_TO_YCbCr	3	Converts image data from YCbCr to YCbCr.
PFV_MATRIX_MODE_COUNT	4	Number of values of the pfv_color_matrix_mode_t type

6.6.6 pfv_dc_offset_index_t

Array number for an offset component in the color matrix.

The components of input image data are incremented for brightness adjustment.

When image data is converted from YCbCr to another format, Cb (color difference U) and Cr (color difference V) are always decremented by 128.

For RGB565, only the high-order 5 or 6 bits of the 8-bit integer are effective. Small-scale offset adjustment by changing only the low-order bits has no effect.

Refer to member variable dc_offset_plus_128 of the pfv_color_matrix_t type described in Section 6.7.5.

Symbol	Value	Description

PFV_DC_OFFSET_Y_OR_GREEN	0	Array number assigned to the offset for Y (brightness) or green
PFV_DC_OFFSET_BLUE	1	Array number assigned to the offset for blue
PFV_DC_OFFSET_RED	2	Array number assigned to the offset for red

6.6.7 pfv_matrix_multiply_index_t

This is array number for a gain component in the color matrix.

The calculated offset components are added together for gain adjustment.

Refer to member variable matrix_multiply_256 of the pfv_color_matrix_t type described in Section 6.7.5.

Symbol	Value	Description
PFV_COLOR_MATRIX_GG	0	When image data is converted to RGB: G1 = GG*G0 + GB*B0 + GR*R0 B1 = BG*G0 + BB*B0 + BR*R0 R1 = RG*G0 + RB*B0 + RR*R0 When image data is converted to YCbCr: Y1 = YY*G0 + GB*B0 + GR*R0 Cb1 = BG*G0 + BB*B0 + BR*R0 + 128 Cr1 = RG*G0 + RB*B0 + RR*R0 + 128 R0, G0, and B0 are the calculated offset components for red/Cr (color difference V), green/Y (brightness), and blue/Cb (color difference U), respectively. R1, G1, B1, Y1, Cb1, and Cr1 are the components of output image data. GG, GB, GR, BG, BB, BR, RG, RB, RR, and YY are the suffixes for the symbols on the left.
PFV_COLOR_MATRIX_GB	1	
PFV_COLOR_MATRIX_GR	2	
PFV_COLOR_MATRIX_BG	3	
PFV_COLOR_MATRIX_BB	4	
PFV_COLOR_MATRIX_BR	5	
PFV_COLOR_MATRIX_RG	6	
PFV_COLOR_MATRIX_RB	7	
PFV_COLOR_MATRIX_RR	8	
PFV_COLOR_MATRIX_YY	0	

6.6.8 pfv_idtrg_t

This is number of bytes of data in the input FIFO when it is determined to be empty. The maximum capacity of this FIFO is 32 bytes.

Symbol	Value	Description
PFV_IDTRG_SPACED_2_BYTE	0	2 bytes
PFV_IDTRG_SPACED_4_BYTE	1	4 bytes
PFV_IDTRG_SPACED_16_BYTE	2	16 bytes
PFV_IDTRG_SPACED_32_BYTE	3	32 bytes

6.6.9 pfv_odtrg_t

This is number of bytes of data in the output FIFO when it is determined to be full. The maximum capacity of this FIFO is 32 bytes.

Symbol	Value	Description
PFV_ODTRG_FILLED_2_BYTE	0	2 bytes
PFV_ODTRG_FILLED_4_BYTE	1	4 bytes
PFV_ODTRG_FILLED_16_BYTE	2	16 bytes
PFV_ODTRG_FILLED_32_BYTE	3	32 bytes

6.6.10 pfv_interrupt_line_t

A kind of interrupt line

Symbol	Value	Description
PFV_INTERRUPT_LINE_PFVEIn	1	Interrupt of PFVEI. PFV error
PFV_INTERRUPT_LINE_IFEIn	2	Interrupt of IFEI. Input FIFO is empty
PFV_INTERRUPT_LINE_OFFIn	4	Interrupt of OFFIn. Output FIFO is full

6.6.11 pfv_interrupt_status_t

This is bit flags which contains two or more pfv_interrupt_line_t values (described in Section 6.6.10, pfv_interrupt_line_t.)

It is used by setting of enabling or disabling interrupt line.

Symbol	Value	Description
PFV_INTERRUPT_LINE_ALL	-	All kind of interrupt line

6.6.12 pfv_dmac_interrupt_unit_t

A kind of peripheral signaled interrupt

Symbol	Value	Description
PFV_INTERRUPT_UNIT_PFV	1	PFV
PFV_INTERRUPT_UNIT_DMAC	2	DMAC

6.6.13 Values within Unnamed Enumerations and Constants

Symbol	Value	Description
PFV_CHANNEL_MIN	0	Minimum channel number of the PFV
PFV_CHANNEL_MAX	1	Maximum channel number of the PFV
PFV_INPUT_FIFO_FULL_BYTE	32	Number of bytes in the PFV's input FIFO
PFV_OUTPUT_FIFO_FULL_BYTE	32	Number of bytes in the PFV's output FIFO

6.7 Structures and Unions

Table 6-6 Structures and Unions

Section	Symbol	Outline
6.7.1	pfv_dmac_config_t	Parameters for the R_PFV_DMAC_Initialize function
6.7.2	pfv_config_t	Parameters for the R_PFV_Initialize function
6.7.3	pfv_io_format_t	Parameters for the R_PFV_DMAC_SetIOFormat function
6.7.4	pfv_transfer_config_t	Parameters for the R_PFV_DMAC_Transfer function
6.7.5	pfv_color_matrix_t	Parameters for the R_PFV_DMAC_SetImageColorMatrix function
6.7.6	pfv_reset_color_matrix_t	Reset value for the color matrix
6.7.7	pfv_dma_bit_count_t	DMA transfer size and the full and empty levels for the PFV's FIFO
*2	r_ospl_async_t	Setting of Notification
*2	pfv_async_status_t	Status and Interrupt status. Same as "r_ospl_async_status_t"

6.7.1 pfv_dmac_config_t

Outline: Parameters for the R_PFV_DMAC_Initialize function
Header: devdrv_pfv.h
Description:
Member variable:

bit_flags_t flags	Flagged structure parameters. Refer to Section 6.10.1. F_PFV_DMAC_INPUT_DMAC_CHANNEL F_PFV_DMAC_OUTPUT_DMAC_CHANNEL F_PFV_DMAC_RESET_COLOR_MATRIXES
int_fast32_t input_DMAC_channel	Channel number of the DMAC which transfers input image data from memory to the PFV. If this variable is omitted, the DMAC is not used.
int_fast32_t output_DMAC_channel	Channel number of the DMAC which transfers output image data from the PFV to memory. If this variable is omitted, the DMAC is not used.
pfv_color_matrix_sub_t* reset_color_matrixes	Reset value for the color matrix. Refer to Section 6.7.6.
pfv_get_dma_bit_count_func_t get_dma_bit_count_func	Using function instead of "R_Userdef_PFV_GetDMA_BitCount" function

6.7.2 pfv_config_t

Outline: Parameters for the R_PFV_Initialize function
Header: devdrv_pfv.h
Description:
Member variable:

bit_flags_t flags	Flagged structure parameter. Refer to Section 6.10.1. F_PFV_RESET_COLOR_MATRIXES
pfv_color_matrix_sub_t* reset_color_matrixes	Reset value for the color matrix. Refer to Section 6.7.6.

6.7.3 pfv_io_format_t

Outline: Parameters for the R_PFV_DMAC_SetIOFormat function, etc.
Header: devdrv_pfv.h
Description:

*2 Refer to RZ/A1H Group OS porting layer "OSPL" (R01AN1887JJ)

Member variable

bit_flags_t flags	Flagged structure parameters. Refer to Section 6.10.1. F_PFV_INPUT_FORMAT F_PFV_INPUT_SWAP F_PFV_OUTPUT_FORMAT F_PFV_OUTPUT_SWAP F_PFV_OUTPUT_ALPHA F_PFV_IS_DITHER F_PFV_IMAGE_WIDTH F_PFV_IMAGE_HEIGHT
pfv_format_t input_format	Pixel format of the input image. If this variable is omitted, the setting is not changed. The initial value is PFV_RGB888.
pfv_swap_t input_swap	Byte-order swapping which applies to the pixel format of the input image. The default order is one of the following depending on the input_format member variable: PFV_SWAP_ARGB8888 PFV_SWAP_RGB565_PIXEL10 PFV_SWAP_YCbCr422_Cb_Y0_Cr_Y1
pfv_format_t output_format	Pixel format of the output image. If this variable is omitted, the setting is not changed. The initial value is PFV_ARGB8888.
pfv_swap_t output_swap	Byte-order swapping which applies to the pixel format of the output image. The default order is one of the following depending on the output_format member variable: PFV_SWAP_ARGB8888 PFV_SWAP_RGB565_PIXEL10 PFV_SWAP_YCbCr422_Cb_Y0_Cr_Y1
int_t output_alpha	Value of the alpha component of the output image. This variable is effective only when output_format is PFV_ARGB8888. This variable should be in the range of 0 to 255. If this variable is omitted, 255 is assumed.
bool_t is_dither	Specifies whether to perform dithering. This variable is effective only when output_format is PFV_RGB565. If the number of input data color bits is not greater than that of output data color bits, the effects of dithering are not obtained even if this variable is effective. If this variable is omitted, the setting is not changed. The initial value is true.
int_t image_width	Width (in pixels) of an image. This variable is required for dithering only. If this variable is omitted, the setting is not changed. The initial value is 0. This variable overwrites pfv_transfer_config_t::buffer_width specified for the R_PFV_DMAC_Transfer function.
int_t image_height	Height (in pixels) of an image. This variable is required for dithering only. If this variable is omitted, the setting is not changed. The initial value is 0. This variable overwrites pfv_transfer_config_t::buffer_height specified for the R_PFV_DMAC_Transfer function.

6.7.4 pfv_transfer_config_t

Outline: Parameters for the R_PFV_DMAC_Transfer function
Header: devdrv_pfv.h
Description:
Member variable:

bit_flags_t flags	Flagged structure parameters. Refer to Section 6.10.1. F_PFV_TRANSFER_INPUT_BUFFER_ADDRESS F_PFV_TRANSFER_INPUT_BUFFER_SIZE F_PFV_TRANSFER_OUTPUT_BUFFER_ADDRESS F_PFV_TRANSFER_OUTPUT_BUFFER_SIZE F_PFV_TRANSFER_BUFFER_WIDTH F_PFV_TRANSFER_BUFFER_HEIGHT
uintptr_t input_buffer_address	Physical start address of memory containing the input image. If this variable is omitted, the DMAC is not used. In this case, write the data directly to the PFV by using the R_PFV_WritePixelDataViaPIO function.
size_t input_buffer_size	Size (in bytes) of memory containing the input image. This variable is mandatory if input_buffer_address is specified. This variable is used to check whether the buffer size is large enough when the buffer_width and buffer_height member variables are specified. The size of data to be converted is calculated from the buffer_width and buffer_height member variables.
uintptr_t output_buffer_address	Physical start address of memory containing the output image. If this variable is omitted, the DMAC is not used. In this case, read the data directly from the PFV by using the R_PFV_ReadPixelDataViaPIO function.
size_t output_buffer_size	Size (in bytes) of memory containing the output image. This variable is mandatory if output_buffer_address is specified. If the buffer_width and buffer_height member variables are omitted and if the size specified with this variable is greater than the output image size, then this driver does not determine that the transfer has completed. This variable is used to check whether the buffer size is large enough when the buffer_width and buffer_height member variables are specified. The size of data to be converted is calculated from the buffer_width and buffer_height member variables. After execution of the R_PFV_DMAC_Transfer function, this variable becomes the size of the output image.
int_t buffer_width	Width (in pixels) of the image. If this variable is omitted, the size of data to be converted is equal to the size specified with input_buffer_size and not the size calculated from the width and height of the image.
int_t buffer_height	Height (in pixels) of the image. This variable is mandatory if buffer_width is specified.
int_fast32_t interval_count_of_DMA	Value of setting to "ITVL" bit in "CHITVL_n" register of DMAC. Interval count. Number of counting from finish of reading or writing to next reading or writing. Unit of count is Bϕ clock. Default is 0.

r_ospl_axi_cache_attribute_t source_AXI_cache_attribute	Value of setting to "SCA" bit in "CHEXT_n" register of DMAC. Cache attribute of AXI bus, when DMAC reads via L2 cache. Default is R_OSPL_AXI_CACHE_ZERO that is for internal bus of RZ/A1H.
r_ospl_axi_protection_t source_AXI_protection	Value of setting to "SPR" bit in "CHEXT_n" register of DMAC. Protection attribute of AXI bus, when DMAC reads via L2 cache. Default is R_OSPL_AXI_PROTECTION_ZERO that is for internal bus of RZ/A1H.
r_ospl_axi_cache_attribute_t destination_AXI_cache_attribute	Value of setting to "DCA" bit in "CHEXT_n" register of DMAC. Cache attribute of AXI bus, when DMAC writes via L2 cache. Default is R_OSPL_AXI_CACHE_ZERO that is for internal bus of RZ/A1H.
r_ospl_axi_protection_t destination_AXI_protection	Value of setting to "DPR" bit in "CHEXT_n" register of DMAC. Protection attribute of AXI bus, when DMAC writes via L2 cache. Default is R_OSPL_AXI_PROTECTION_ZERO that is for internal bus of RZ/A1H.

6.7.5 pfv_color_matrix_t

Outline: Parameters for the R_PFV_DMAC_SetImageColorMatrix function, etc.

Header: devdrv_pfv.h

Description: The variables of this data type are arranged in the same configuration as those of the vdc5_color_matrix_t type for the VDC5 driver. Variables of the vdc5_color_matrix_t type can be cast to the pfv_color_matrix_t type, and then the variables of this data type can be used.

Member variable:

int32_t dummy	Unused. Available to achieve compatibility with the vdc5_color_matrix_t type.
pfv_color_matrix_mode_t mode	Type of conversion of image data between pixel formats. Refer to Section 6.6.5.
uint16_t dc_offset_plus_128[ ]	Array of values each of which is the sum of an offset component in the color matrix and 128. The components are incremented for brightness adjustment. The array number is of the pfv_dc_offset_index_t type described in Section 6.6.6, pfv_dc_offset_index_t.
int16_t matrix_multiply_256[ ]	Array of values each of which is the product of a gain component in the color matrix and 256. The components are added together for gain adjustment. The array number is of the pfv_matrix_multiply_index_t type described in Section 6.6.7, pfv_matrix_multiply_index_t.

Example:

```
#define NUM_2048 2048 /* if ( matrix_multiply_256 < 0 ) matrix_multiply_256
= register_value - 2048 */
static const pfv_reset_color_matrix_t
gs_ResetColorMatrixes[ PFV_MATRIX_MODE_COUNT ] =
{
    {   /* GBR => GBR */
        { 128, 128, 128 },
        { 256, 0, 0, 0, 256, 0, 0, 0, 256 }
    },
    {   /* GBR => YCBCR, SMPTE 293M */
        { 128, 128, 128 },
```

```
        { 150, 29, 77, 1963-NUM_2048, 128, 2005-NUM_2048, 1941-NUM_2048,
          2027-NUM_2048, 128 }
    },
    {   /* YCBCR => GBR, SMPTE 293M */
        { 128, 128, 128 },
        { 256, 1960-NUM_2048, 1865-NUM_2048, 256, 454, 0, 256, 0, 359 }
    },
    {   /* YCBCR => YCBCR */
        { 128, 128, 128 },
        { 256, 0, 0, 0, 256, 0, 0, 0, 256 }
    }
};
#undef  NUM_2048
```

6.7.6 pfv_reset_color_matrix_t

Outline	Reset value for the color matrix	
Header	r_pfv.h	
Description	Array of the pfv_reset_color_matrix_t type, which consists of the number of elements specified with PFV_MATRIX_MODE_COUNT. The element numbers of this array are of the pfv_color_matrix_mode_t type.	
Member variable	uint16_t dc_offset_plus_128[ ]	Array of values each of which is the sum of an offset component in the color matrix and 128. The components are incremented for brightness adjustment. The array number is of the pfv_dc_offset_index_t type described in Section 6.6.6, pfv_dc_offset_index_t.
	int16_t matrix_multiply_256[ ]	Array of values each of which is the product of a gain component in the color matrix and 256. The components are added together for gain adjustment. The array number is of the pfv_matrix_multiply_index_t type described in Section 6.6.7, pfv_matrix_multiply_index_t.

Example:

```
#define  NUM_2048  2048  /* if ( matrix_multiply_256 < 0 )  matrix_multiply_256
= register_value - 2048 */
static const pfv_reset_color_matrix_t
gs_ResetColorMatrixes[ PFV_MATRIX_MODE_COUNT ] =
{
    {   /* GBR => GBR */
        { 128, 128, 128 },
        { 256, 0, 0, 0, 256, 0, 0, 0, 256 }
    },
    {   /* GBR => YCBCR, SMPTE 293M */
        { 128, 128, 128 },
        { 150, 29, 77, 1963-NUM_2048, 128, 2005-NUM_2048, 1941-NUM_2048,
          2027-NUM_2048, 128 }
    },
    {   /* YCBCR => GBR, SMPTE 293M */
        { 128, 128, 128 },
        { 256, 1960-NUM_2048, 1865-NUM_2048, 256, 454, 0, 256, 0, 359 }
    },
    {   /* YCBCR => YCBCR */
        { 128, 128, 128 },
        { 256, 0, 0, 0, 256, 0, 0, 0, 256 }
    }
};
#undef  NUM_2048
```

6.7.7 pfv_dma_bit_count_t

Outline	DMA transfer size and the full and empty levels for the PFV FIFO
Header	r_pfv.h
Description	Arguments for the R_Userdef_PFV_GetDMA_BitCount function (6.9.5(11))

Member variable

dmac_bit_count_t input_buffer_DMA_bit_count	Size of DMA transfer to the input buffer. Refer to RZ/A1H Group DMAC_RM Example program (attached PFV) (R01AN1888JJ).
dmac_bit_count_t input_PFV_DMA_bit_count	Size of DMA transfer to the PFV's input FIFO
pfv_idtrg_t input_trigger_byte_count	Number of bytes when the PFV's input FIFO becomes empty. Refer to Section 6.6.8.
dmac_bit_count_t output_PFV_DMA_bit_count	Size of DMA transfer to the PFV's output FIFO
dmac_bit_count_t output_buffer_DMA_bit_count	Size of DMA transfer to the output buffer
pfv_odtrg_t output_trigger_byte_count	Number of bytes when the PFV's output FIFO becomes full. Refer to Section 6.6.9.

6.8 List of Variables

Table 6-7 shows the static variables. Table 6-8 shows the const Variables.

Table 6-7 Static Variables

Type	Variable Name	Contents	Function Used
pfv_dmac_contexts_t	gs_pfv_dmac_contexts	PFV-DMAC	Some functions
pfv_contexts_t	gs_pfv_contexts	PFV	Some functions
r_dmac_temporary_channel_t	gs_dmac_temporary_channel	DMAC	Some functions

Table 6-8 const Variables

Type	Variable Name	Contents	Function Used
None			

6.9 Functions

6.9.1 List

Section	Abstract
(1)	Functions for cooperation between the PFV and DMACs
(2)	Functions for PFV operation only
(3)	Functions, available before initialization, for PFV operation only
(4)	List of driver porting layer functions

(1) Functions for cooperation between the PFV and DMACs

Section	Function Name	Outline
6.9.2(1)	R_PFV_DMAC_Initialize	Initializes the PFV, input DMAC, and output DMAC.
6.9.2(2)	R_PFV_DMAC_Terminate	Performs termination processing for the PFV, input DMAC, and output DMAC.
6.9.2(3)	R_PFV_DMAC_SetIOFormat	Configures the settings for conversion of image data between pixel formats.
6.9.2(4)	R_PFV_DMAC_GetIOFormat	Obtains the settings for conversion of image data between pixel formats.
6.9.2(5)	R_PFV_DMAC_ResetImageColorMatrix	Resets the color matrix.
6.9.2(6)	R_PFV_DMAC_SetImageColorMatrix	Sets the color matrix.
6.9.2(7)	R_PFV_DMAC_GetImageColorMatrix	Obtains the color matrix settings.
6.9.2(8)	R_PFV_DMAC_Transfer	Converts image data. (Synchronous)
6.9.2(9)	R_PFV_DMAC_TransferAsync	Converts image data. (Asynchronous)
6.9.2(10)	R_PFV_DMAC_OnInterrupting	Receives an interrupt
6.9.2(11)	R_PFV_DMAC_OnInterrupted	Responds an interrupt
6.9.2(12)	R_PFV_DMAC_StopTransfer	Aborts image data conversion.

(2) Functions for PFV operation only

Section	Function Name	Outline
6.9.3(1)	R_PFV_Initialize	Initializes the PFV.
6.9.3(2)	R_PFV_Terminate	Performs termination processing for the PFV.
6.9.3(3)	R_PFV_SetIOFormat	Configures the settings for image data conversion between pixel formats.
6.9.3(4)	R_PFV_GetIOFormat	Obtains the settings for image data conversion between pixel formats.
6.9.3(5)	R_PFV_ResetImageColorMatrix	Resets the color matrix.
6.9.3(6)	R_PFV_SetImageColorMatrix	Sets the color matrix.
6.9.3(7)	R_PFV_GetImageColorMatrix	Obtains the color matrix settings.
6.9.3(8)	R_PFV_GetFilledByteInInputFIFO	Obtains the number of bytes of data in the PFV's input FIFO.
6.9.3(9)	R_PFV_WritePixelDataViaPIO	Writes image data to the PFV.
6.9.3(10)	R_PFV_GetFilledByteInOutputFIFO	Obtains the number of bytes of data in the PFV's output FIFO.
6.9.3(11)	R_PFV_ReadPixelDataViaPIO	Reads image data from the PFV.
6.9.3(12)	R_PFV_OnInterrupting	Receives an interrupt
6.9.3(13)	R_PFV_OnInterrupted	Responds an interrupt
6.9.3(14)	R_PFV_GetAsyncStatus	Gets the pointer to the struct of interrupts and asynchronous operation

(3) Functions, available before initialization, for PFV operation only

Section	Function Name	Outline
6.9.4(1)	R_PFV_STATIC_GetColorMatrixMode	Obtains the appropriate value of the pfv_color_matrix_mode_t type.

(4) List of driver porting layer functions

Section	Function Name	Outline
6.9.5(1)	R_Userdef_PFV_SetDefaultAsync	Sets default value of r_ospl_async_t type structure
6.9.5(2)	R_Userdef_PFV_OnInitialize	Initializes the driver porting layer part.
6.9.5(3)	R_Userdef_PFV_OnFinalize	Performs termination processing for the driver porting layer part.
6.9.5(4)	R_Userdef_PFV_SetInterruptCallbackCaller	Sets interrupt callback function caller to driver porting layer part
6.9.5(5)	R_Userdef_PFV_OnEnableInterrupt	Enables interrupts
6.9.5(6)	R_Userdef_PFV_OnDisableInterrupt	Disables interrupts
6.9.5(7)	R_Userdef_PFV_OnInterruptDefault	Default interrupt callback function
6.9.5(8)	R_Userdef_PFV_DMAC_SetDefaultConfig	Sets default configuration of pfv_dmac_config_t type structure
6.9.5(9)	R_Userdef_PFV_SetDefaultConfig	Sets default configuration of pfv_config_t type structure
6.9.5(10)	R_Userdef_PFV_DMAC_GetInterruptUnit	Gets a kind of peripheral signaled interrupt
6.9.5(11)	R_Userdef_PFV_GetDMA_BitCount	Obtains the DMA transfer size. Also, obtains the full and empty levels for the PFV's FIFO.

6.9.2 Functions for cooperation between the PFV and DMACs

(1) R_PFV_DMAC_Initialize

Outline	Initializes the PFV, input DMAC, and output DMAC.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_DMAC_Initialize(int_fast32_t pfv_channel, pfv_dmac_config_t* in_out_config);	
Description	Initializes the internal variables. The input DMAC transfers input image data from memory to the PFV. The output DMAC transfers output image data from the PFV to memory. Interrupts (IFEI and OFFI) from the PFV are sent to the DMAC but not to the CPU. They set all the elements of the color matrix to 0.	
Arguments	int_fast32_t pfv_channel	PFV channel numbers. PFV_CHANNEL_MIN to PFV_CHANNEL_MAX.
	pfv_dmac_config_t* in_out_config	Other settings. Refer to Section 6.7.1. NULL is not permitted.
Return value	Error code. If there is no error, the return value is 0.	

(2) R_PFV_DMAC_Terminate

Outline	Performs termination processing for the PFV, input DMAC, and output DMAC.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_DMAC_Terminate(int_fast32_t pfv_channel);	
Description	This function aborts the processing if the DMACs are active.	
Arguments	int_fast32_t pfv_channel	PFV channel number
Return value	Error code. If there is no error, the return value is 0.	

(3) R_PFV_DMAC_SetIOFormat

Outline	Configures the settings for image data conversion between pixel formats.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_DMAC_SetIOFormat(int_fast32_t pfv_channel, pfv_io_format_t* in_out_io_format);	
Description	This function also resets the color matrix. Refer to (5) R_PFV_DMAC_ResetImageColorMatrix.	
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_io_format_t* in_out_io_format	Setting. Refer to Section 6.7.3. NULL is not permitted.
Return value	Error code. If there is no error, the return value is 0.	

(4) R_PFV_DMAC_GetIOFormat

Outline	Obtains the settings for image data conversion between pixel formats.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_DMAC_GetIOFormat(int_fast32_t pfv_channel, pfv_io_format_t* out_io_format);	
Description		
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_io_format_t* out_io_format	Output: Setting. Refer to Section 6.7.3.
Return value	Error code. If there is no error, the return value is 0.	

(5) R_PFV_DMAC_ResetImageColorMatrix

Outline	Resets the color matrix.
Header	r_pfv.h

Declaration	errnum_t R_PFV_DMAC_ResetImageColorMatrix(int_fast32_t pfv_channel, pfv_color_matrix_mode_t mode);
Description	The offset is 0. The gain factor is 1. For conversion between RGB and YCbCr formats, the conversion equation includes the color matrix values defined in SMPTE 293M.

Arguments		
	int_fast32_t pfv_channel	PFV channel number
	pfv_color_matrix_mode_t mode	Type of image data conversion between pixel formats. Refer to Section 6.6.5.
Return value	Error code. If there is no error, the return value is 0.	

(6) R_PFV_DMAC_SetImageColorMatrix

Outline	Sets the color matrix.
Header	r_pfv.h
Declaration	errnum_t R_PFV_DMAC_SetImageColorMatrix(int_fast32_t pfv_channel, pfv_color_matrix_t* offset_and_matrix);
Description	

Arguments		
	int_fast32_t pfv_channel	PFV channel number
	pfv_color_matrix_t* offset_and_matrix	Color matrix. Refer to Section 6.7.5.
Return value	Error code. If there is no error, the return value is 0.	

(7) R_PFV_DMAC_GetImageColorMatrix

Outline	Obtains the color matrix settings.
Header	r_pfv.h
Declaration	errnum_t R_PFV_DMAC_GetImageColorMatrix(int_fast32_t pfv_channel, pfv_color_matrix_t* out_offset_and_matrix);
Description	

Arguments		
	int_fast32_t pfv_channel	PFV channel number
	pfv_color_matrix_t* out_offset_and_matrix	Output: Color matrix. Refer to Section 6.7.5.
Return value	Error code. If there is no error, the return value is 0.	

(8) R_PFV_DMAC_Transfer

Outline	Converts image data. (Synchronous)
Header	r_pfv.h
Declaration	errnum_t R_PFV_DMAC_Transfer(int_fast32_t pfv_channel, pfv_transfer_config_t* in_out_config);
Description	This function converts via the DMAC and the PFV. This is synchronous function that does not return until to complete the convert.

Arguments		
	int_fast32_t pfv_channel	PFV channel number
	pfv_transfer_config_t* in_out_config	Setting. Refer to Section 6.7.4.
Return value	Error code. If there is no error, the return value is 0.	

(9) R_PFV_DMAC_TransferAsync

Outline	Converts image data. (Asynchronous)
Header	r_pfv.h
Declaration	errnum_t R_PFV_DMAC_TransferAsync(int_fast32_t pfv_channel, pfv_transfer_config_t* in_out_config, r_ospl_async_t* async);
Description	This function starts transfer via the DMAC for conversion which uses the PFV.

This is asynchronous function that returns soon after starting the convert.
The detail of async argument is explained at the section of R_DRIVER_TransferStart function in RZ/A1H Group OS porting layer "OSPL" (R01AN1887JJ)

Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_transfer_config_t* in_out_config	Setting. Refer to Section 6.7.4.
	r_ospl_async_t* async	Setting of Notification
Return value	Error code. If there is no error, the return value is 0.	

(10) R_PFV_DMAC_OnInterrupting

Outline	Receives an interrupt.
Header	r_pfv.h
Declaration	errnum_t R_PFV_DMAC_OnInterrupting(const r_ospl_interrupt_t* InterruptSource);
Description	Normally, this function is called automatically from default interrupt callback function. This function calls R_PFV_OnInterrupting function receiving an interrupt or R_DMAC_TEMPORARY_OnInterrupting function.

Arguments	r_ospl_interrupt_t* InterruptSource	Interrupt source
Return value	Error code. If there is no error, the return value is 0.	

(11) R_PFV_DMAC_OnInterrupted

Outline	Responds an interrupt.
Header	r_pfv.h
Declaration	errnum_t R_PFV_DMAC_OnInterrupted(int_fast32_t pfv_channel);
Description	Normally, this function is called automatically from default interrupt callback function. This function calls R_PFV_OnInterrupted function responding an interrupt or R_DMAC_TEMPORARY_OnInterrupted function.

Arguments	int_fast32_t pfv_channel	PFV channel number
Return value	Error code. If there is no error, the return value is 0.	

(12) R_PFV_DMAC_StopTransfer

Outline	Aborts image data transfer.
Header	r_pfv.h
Declaration	errnum_t R_PFV_DMAC_StopTransfer(int_fast32_t pfv_channel);
Description	

Arguments	int_fast32_t pfv_channel	PFV channel number
Return value	Error code. If there is no error, the return value is 0.	

6.9.3 Functions for PFV operation only

(1) R_PFV_Initialize

Outline	Initializes the PFV.
Header	r_pfv.h
Declaration	errnum_t R_PFV_Initialize(int_fast32_t pfv_channel, pfv_config_t* in_out_config);
Description	This function Initializes the internal variables and PFV. It internally calls driver porting layer function R_Userdef_PFV_OnInitialize. It sets all the elements of the color matrix to 0.

Arguments	int_fast32_t pfv_channel	PFV channel number PFV_CHANNEL_MIN to PFV_CHANNEL_MAX

	pfv_config_t* in_out_config	Setting. Refer to Section 6.7.2. NULL is permitted.
Return value	Error code. If there is no error, the return value is 0.	

(2) R_PFV_Terminate

Outline	Performs termination processing for the PFV.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_Terminate(int_fast32_t pfv_channel);	
Description	This function internally calls driver porting layer function R_Userdef_PFV_OnFinalize.	
Arguments	int_fast32_t pfv_channel	PFV channel number
Return value	Error code. If there is no error, the return value is 0.	

(3) R_PFV_SetIOFormat

Outline	Configures the settings for image data conversion between pixel formats.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_SetIOFormat(int_fast32_t pfv_channel, pfv_io_format_t* in_out_io_format);	
Description	This function also resets the color matrix. Refer to (5) R_PFV_ResetImageColorMatrix.	
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_io_format_t* in_out_io_format	Setting. Refer to Section 6.7.3. NULL is not permitted.
Return value	Error code. If there is no error, the return value is 0.	

(4) R_PFV_GetIOFormat

Outline	Obtains the settings for image data conversion between pixel formats.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_GetIOFormat(int_fast32_t pfv_channel, pfv_io_format_t* out_io_format);	
Description		
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_io_format_t* out_io_format	Output: Setting. Refer to Section 6.7.3.
Return value	Error code. If there is no error, the return value is 0.	

(5) R_PFV_ResetImageColorMatrix

Outline	Resets the color matrix.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_ResetImageColorMatrix(int_fast32_t pfv_channel, pfv_color_matrix_mode_t mode);	
Description	The offset is 0. The gain factor is 1. For conversion between RGB and YCbCr formats, the conversion equation includes the color matrix values defined in SMPTE 293M.	
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_color_matrix_mode_t mode	Type of image data conversion between pixel formats. Refer to Section 6.6.5.
Return value	Error code. If there is no error, the return value is 0.	

(6) R_PFV_SetImageColorMatrix

Outline	Sets the color matrix.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_SetImageColorMatrix(int_fast32_t pfv_channel, pfv_color_matrix_t* offset_and_matrix);	
Description		
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_color_matrix_t* offset_and_matrix	Color matrix. Refer to Section 6.7.5.
Return value	Error code. If there is no error, the return value is 0.	

(7) R_PFV_GetImageColorMatrix

Outline	Obtains the color matrix settings.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_GetImageColorMatrix(int_fast32_t pfv_channel, pfv_color_matrix_t* out_offset_and_matrix);	
Description		
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_color_matrix_t* out_offset_and_matrix	Output: Color matrix. Refer to Section 6.7.5.
Return value	Error code. If there is no error, the return value is 0.	

(8) R_PFV_GetFilledByteInInputFIFO

Outline	Obtains the number of bytes of data in the PFV's input FIFO.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_GetFilledByteInInputFIFO(int_fast32_t pfv_channel, int_t* out_filled_byte);	
Description	The maximum number of bytes in the PFV's input FIFO is equal to PFV_INPUT_FIFO_FULL_BYTE. If the PFV encounters an error, the return value from this function indicates the error code.	
Arguments	int_fast32_t pfv_channel	PFV channel number
	int_t* out_filled_byte	Output: Number of bytes in the PFV's input FIFO
Return value	Error code. If there is no error, the return value is 0.	

(9) R_PFV_WritePixelDataViaPIO

Outline	Writes non-converted image data to the PFV.	
Header	r_pfv.h	
Declaration	errnum_t R_PFV_WritePixelDataViaPIO(int_fast32_t pfv_channel, uint32_t pixel_data, int_t bit_count);	
Description	If the input FIFO is full when this function is executed, an error occurs. Using the R_PFV_GetFilledByteInInputFIFO function, check whether the FIFO is full.	
Arguments	int_fast32_t pfv_channel	PFV channel number
	uint32_t pixel_data	One pixel of image data. Two pixels of image data for PFV_YCbCr422.
	int_t bit_count	Number of bits of image data passed to pixel_data. It is 16 for PFV_RGB565 or 32 for others.
Return value	Error code. If there is no error, the return value is 0.	

(10) R_PFV_GetFilledByteInOutputFIFO

Outline	Obtains the number of bytes of data in the PFV's output FIFO.

Header	r_pfv.h
Declaration	errnum_t R_PFV_GetFilledByteInOutputFIFO(int_fast32_t pfv_channel, int_t* out_filled_byte);
Description	The maximum number of bytes in the PFV's output FIFO is equal to PFV_OUTPUT_FIFO_FULL_BYTE. If the PFV encounters an error, the return value from this function indicates the error code.

Arguments		
	int_fast32_t pfv_channel	PFV channel number
	int_t* out_filled_byte	Output: Number of bytes in the PFV's output FIFO
Return value	Error code. If there is no error, the return value is 0.	

(11) R_PFV_ReadPixelDataViaPIO

Outline	Reads converted image data from the PFV.
Header	r_pfv.h
Declaration	errnum_t R_PFV_ReadPixelDataViaPIO(int_fast32_t pfv_channel, uint32_t* out_pixel_data, int_t bit_count);
Description	If the output FIFO is empty when this function is executed, an error occurs. Using the R_PFV_GetFilledByteInOutputFIFO function, check whether the FIFO is empty.

Arguments		
	int_fast32_t pfv_channel	PFV channel number
	uint32_t* out_pixel_data	Output: One pixel of image data. Two pixels of image data for PFV_YCbCr422.
	int_t bit_count	Number of bits of image data passed to pixel_data. It is 16 for PFV_RGB565 or 32 for others.
Return value	Error code. If there is no error, the return value is 0.	

(12) R_PFV_OnInterrupting

Outline	Receives an interrupt.
Header	r_pfv.h
Declaration	errnum_t R_PFV_OnInterrupting(const r_ospl_interrupt_t* InterruptSource);
Description	Normally, this function is called automatically from default interrupt callback function. This function notifies an interrupt from interrupt status register to pfv_async_status_t::InterruptFlags variable and clears interrupt. Refer to the section of R_DRIVER_OnInterrupting function in RZ/A1H Group OS porting layer "OSPL" (R01AN1887JJ)

Arguments		
	r_ospl_interrupt_t* InterruptSource	Interrupt source
Return value	Error code. If there is no error, the return value is 0.	

(13) R_PFV_OnInterrupted

Outline	Responds an interrupt.
Header	r_pfv.h
Declaration	errnum_t R_PFV_OnInterrupted(int_fast32_t const pfv_channel);
Description	Normally, this function is called automatically from default interrupt callback function. This function clear to 0 in pfv_async_status_t::InterruptFlags variable set to 1 by R_PFV_OnInterrupting function and responds an interrupt. Refer to the section of R_DRIVER_OnInterrupted function in RZ/A1H Group OS porting layer "OSPL" (R01AN1887JJ)

Arguments		
	int_fast32_t pfv_channel	PFV channel number
Return value	Error code. If there is no error, the return value is 0.	

(14) R_PFV_GetAsyncStatus

Outline	Gets the pointer to the struct of interrupts and asynchronous operation

Header	r_pfv.h
Declaration	errnum_t R_PFV_GetAsyncStatus(int_fast32_t pfv_channel, const pfv_async_status_t** out_Status);
Description	The pointer variable passed to out_Status argument must be with const qualifier.

Arguments		
	int_fast32_t pfv_channel	PFV channel number
	pfv_async_status_t** out_Status	(Output) The pointer to the struct of interrupts and asynchronous operation
Return value	Error code. If there is no error, the return value is 0.	

6.9.4 Functions, available before initialization, for PFV operation only

(1) R_PFV_STATIC_GetColorMatrixMode

Outline	Obtains the appropriate value of the pfv_color_matrix_mode_t type.
Header	r_pfv.h
Declaration	errnum_t R_PFV_STATIC_GetColorMatrixMode(pfv_format_t input_format, pfv_format_t output_format, pfv_color_matrix_mode_t* out_color_matrix_mode);
Description	

Arguments		
	pfv_format_t input_format	Pixel format to be input to the PFV
	pfv_format_t output_format	Pixel format to be output from the PFV
	pfv_color_matrix_mode_t* out_color_matrix_mode	Refer to Section 6.6.5, pfv_color_matrix_mode_t.
Return value	Error code. If there is no error, the return value is 0.	

6.9.5 Driver porting layer functions

(1) R_Userdef_PFV_SetDefaultAsync

Outline	Sets default value of r_ospl_async_t type structure.
Header	r_pfv_pl.h
Declaration	void R_Userdef_PFV_SetDefaultAsync(r_ospl_async_t* Async);
Description	

Arguments		
	r_ospl_async_t* Async	Input/Output: Setting of Notification
Return value	Error code. If there is no error, the return value is 0.	

(2) R_Userdef_PFV_OnInitialize

Outline	Initializes the driver porting layer part.
Header	r_pfv_pl.h
Declaration	errnum_t R_Userdef_PFV_OnInitialize(int_fast32_t pfv_channel);
Description	If necessary, supply the clock for the PFV. This function is called from the R_PFV_DMAC_Initialize or R_PFV_Initialize function.

Arguments		
	int_fast32_t pfv_channel	PFV channel number
Return value	Error code. If there is no error, the return value is 0.	

(3) R_Userdef_PFV_OnFinalize

Outline	Performs termination processing for the driver porting layer part.
Header	r_pfv_pl.h
Declaration	errnum_t R_Userdef_PFV_OnFinalize(int_fast32_t pfv_channel, errnum_t e);
Description	If necessary, stop to supply the clock for the PFV. This function is called from the R_PFV_DMAC_Terminate or R_PFV_Terminate function.

Arguments		
	int_fast32_t pfv_channel	PFV channel number
	errnum_t e	Errors that have occurred. No error = 0
Return value	Error code or e 0 = successful and e = 0.	

(4) R_Userdef_PFV_SetInterruptCallbackCaller

Outline	Sets interrupt callback function caller to driver porting layer part	
Header	r_pfv_pl.h	
Declaration	errnum_t R_Userdef_PFV_SetInterruptCallbackCaller(int_fast32_t pfv_channel, const r_ospl_caller_t* caller);	
Description	This function is called by each starting to asynchronous operation with interrupts. Call R_OSPL_CallInterruptCallback function with "caller" argument of this function from interrupt handler. Refer to the section of R_OSPL_CallInterruptCallback function in RZ/A1H Group OS porting layer "OSPL" (R01AN1887JJ) It is not necessary to check channel number in this function.	
Arguments	int_fast32_t pfv_channel	PFV channel number
	r_ospl_caller_t* caller	The value passing to R_OSPL_CallInterruptCallback
Return value	Error code. If there is no error, the return value is 0	

(5) R_Userdef_PFV_OnEnableInterrupt

Outline	Enables interrupts	
Header	r_pfv_pl.h	
Declaration	void R_Userdef_PFV_OnEnableInterrupt(int_fast32_t pfv_channel, pfv_interrupt_lines_t enables);	
Description	It is not necessary to check channel number in this function.	
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_interrupt_lines_t enables	Bit flags set to 1 enabling interrupt line
Return value	Error code. If there is no error, the return value is 0	

(6) R_Userdef_PFV_OnDisableInterrupt

Outline	Disables interrupts	
Header	r_pfv_pl.h	
Declaration	void R_Userdef_PFV_OnDisableInterrupt(int_fast32_t pfv_channel, pfv_interrupt_lines_t disables);	
Description	It is not necessary to check channel number in this function.	
Arguments	int_fast32_t pfv_channel	PFV channel number
	pfv_interrupt_lines_t disables	Bit flags set to 1 disabling interrupt line
Return value	Error code. If there is no error, the return value is 0	

(7) R_Userdef_PFV_OnInterruptDefault

Outline	Default interrupt callback function	
Header	r_pfv_pl.h	
Declaration	errnum_t R_Userdef_PFV_OnInterruptDefault(const r_ospl_interrupt_t* interrupt_source, const r_ospl_caller_t* caller);	
Description	This function will be called when InterruptCallback member variable in async argument of R_PFV_DMAC_TransferAsync function was NULL. This function is r_ospl_callback_t type.	
Arguments	r_ospl_interrupt_t* interrupt_source	Interrupt source
	r_ospl_caller_t* caller	The value passed to R_OSPL_CallInterruptCallback
Return value	Error code. If there is no error, the return value is 0	

(8) R_Userdef_PFV_DMAC_SetDefaultConfig

Outline	Sets default configuration of pfv_dmac_config_t type structure
Header	r_pfv_pl.h
Declaration	errnum_t R_Userdef_PFV_DMAC_SetDefaultConfig(pfv_dmac_config_t* in_out_config);
Description	This function sets each member variables to default value related to bits set to 0 in Flags member variable in pfv_dmac_config_t type structure.

Arguments		
	pfv_dmac_config_t* in_out_config	Setting. NULL is not permitted.
Return value	Error code. If there is no error, the return value is 0	

(9) R_Userdef_PFV_SetDefaultConfig

Outline	Sets default configuration of pfv_config_t type structure
Header	r_pfv_pl.h
Declaration	errnum_t R_Userdef_PFV_SetDefaultConfig(pfv_config_t* in_out_config);
Description	This function sets each member variables to default value related to bits set to 0 in Flags member variable in pfv_config_t type structure.

Arguments		
	pfv_config_t* in_out_config	Setting. NULL is not permitted.
Return value	Error code. If there is no error, the return value is 0	

(10) R_Userdef_PFV_DMAC_GetInterruptUnit

Outline	Gets a kind of peripheral signaled interrupt
Header	r_pfv_pl.h
Declaration	errnum_t R_Userdef_PFV_DMAC_GetInterruptUnit(r_ospl_interrupt_t* InterruptSource, r_pfv_dmac_interrupt_unit_t* out_Unit);
Description	Call this function from interrupt callback function. This function outputs which managed peripheral's R_DRIVER_OnInterrupting function. In this function, it gets a kind of interrupt from which peripherals by comparing greater or less with IRQ_Num member variable in InterruptSource argument.

Arguments		
	r_ospl_interrupt_t* InterruptSource	Interrupt source
	r_pfv_dmac_interrupt_unit_t* out_Unit	A kind of peripheral
Return value	Error code. If there is no error, the return value is 0	

(11) R_Userdef_PFV_GetDMA_BitCount

Outline	Obtains the DMA transfer size. Also, obtains the full and empty levels for the FIFO for the PFV.
Header	r_pfv_pl.h
Declaration	errnum_t R_Userdef_PFV_GetDMA_BitCount(uintptr_t input_buffer_start_address, size_t input_buffer_size, uintptr_t output_buffer_start_address, size_t output_buffer_size, pfv_dma_bit_count_t* out);
Description	If input_buffer_size or output_buffer_size = 0, the DMAC is not used for data input to or output from the PFV. Information about the input DMAC or output DMAC is output to the out argument, but this information is ignored. This function is pfv_get_dma_bit_count_func_t type.

Arguments		
	uintptr_t input_buffer_start_address	Physical start address of the input buffer
	size_t input_buffer_size	Number of bytes transferred from the input buffer
	uintptr_t output_buffer_start_address	Physical start address of the output buffer
	size_t output_buffer_size	Number of bytes transferred to the output buffer
	pfv_dma_bit_count_t* out	Output: DMA transfer size and the full and empty levels for the PFV FIFO. Refer to Section 6.7.7.

Return value	Error code. If there is no error, the return value is 0.

6.10 Supplementary Information

6.10.1 Flagged structure parameters

Flags member variables in the structure are used as bit flags, and if a bit is 1, the corresponding member variable is enabled according to the coding pattern. If a bit is 0, the value of the member variable is assumed to be the default value. Even if the version is upgraded so that its structure contains additional members, the old and new versions can be binary compatible.

```
FuncA_ConfigClass  config;

config.Flags  = F_FuncA_Param1 | F_FuncA_Param2;
config.Param1 = 10;
config.Param2 = 2;
FuncA( &config );
```

Because there is not Flags |= F_FuncA_Param3, config.Param3 is the default value.

7. Example Codes

The example codes can be downloaded from the Renesas Electronics website.

8. Documents for Reference

User's Manual: Hardware

RZ/A1H Group User's Manual: Hardware

The latest version can be downloaded from the Renesas Electronics website.

R7S72100 RTK772100BC00000BR (GENMAI) User's Manual

The latest version can be downloaded from the Renesas Electronics website.

R7S72100 CPU (GENMAI) Optional Board RTK7721000B00000BR User's Manual

The latest version can be downloaded from the Renesas Electronics website.

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C

The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification Architecture version 1.0

The latest version can be downloaded from the ARM website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

ARM Software Development Tools (ARM Compiler toolchain, ARM DS-5 etc.) can be downloaded from the ARM website.

The latest version can be downloaded from the ARM website.

Website and Support

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

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

Revision History

Rev.	Date	Description
1.03	Aug. 24, 2017	Example application supports frame buffer on cached area. Updated to OSPL version 1.60.
1.02	Feb. 29, 2016	Added to support L2 cache. Updated initial settings to version 1.01. Updated internal OSPL to version 0.96. Updated internal DMAC driver to version 1.02
1.00	Jun. 20.2014	First edition issued

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 document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance 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 a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving 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, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "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 etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics 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 ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall 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. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, 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.

(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.

(Rev.3.0-1 November 2016)

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0