Crystal Image through
Imaging Innovation

PIXELPLUS

SURROUND VIEW MONITORING SYSTEM

PI5008K Serial Flash User Guide

Rev 0.2

Last Update : 2018.12.05

6th Floor, 105, Gwanggyo-ro, Yeongtong-gu,
Suwong-si, Gyeonggi-do, 16229, Korea
Tel : +82-31-888-5300, FAX : +82-31-888-5399

Contents

Figure

1. Serial Flash

1.1. Introduction

This guide describes serial flash API of PI5008 SVM SDK.

1.2. Layer description

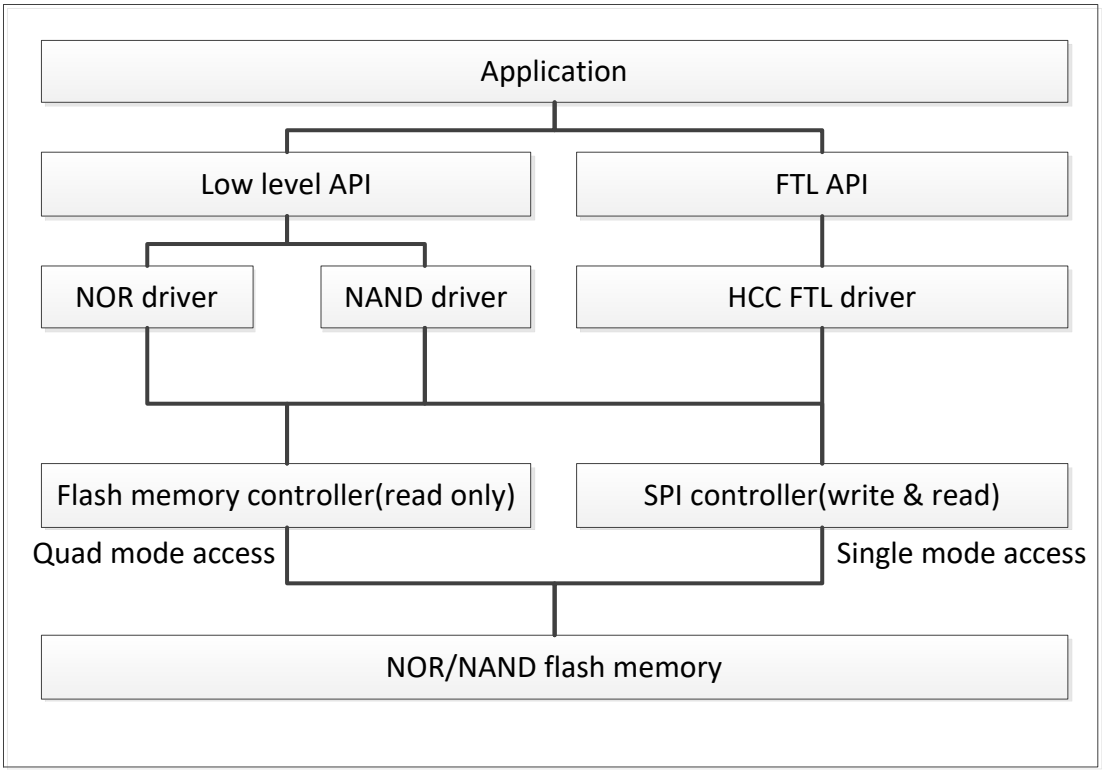

Figure 1 Serial Flash Layer

1.3. Serial Flash API

The flash API supports both NOR and NAND flash memory. NAND flash memory is composed of non-FTL area and FTL area. NOR flash memory and NAND flash memory non-FTL area can be accessed using low level APIs. NAND flash memory FTL area can be accessed using FTL APIs.

1.3.1. Flash header

Flash header information exists in the base address of each area(NOR, NAND FTL, NAND nonFTL area) in flash memory. Flash header includes flash memory information and address, size of each section data and so on.

1.3.1.1. Data type

1) PP_FLASH_SECT_E

A. Syntax

```
typedef enum ppFLASH_SECT_E{
        eFLASH_SECT_BL1 = 0,            //0
        eFLASH_SECT_BL2,                //1
        eFLASH_SECT_IFW,                //2
        eFLASH_SECT_IFW1,               //3
        eFLASH_SECT_MFW,                //4
        eFLASH_SECT_MFW1,               //5
        eFLASH_SECT_SVM_LUT,            //6
        eFLASH_SECT_CAR_IMG,            //7
        eFLASH_SECT_MENU,               //8
        eFLASH_SECT_UI_IMG,             //9
        eFLASH_SECT_PGL_IMG,            //10
        eFLASH_SECT_ISP_DATA,           //11
        eFLASH_SECT_SENSOR_DATA,  //12
        eFLASH_SECT_CAM_CALIB_MAIN_TYPE,        //13
        eFLASH_SECT_CAM_CALIB_SUB_TYPE,         //14
        eFLASH_SECT_CAM_CAPTURE,  //15
        eFLASH_SECT_AUDIO,              //16
        eFLASH_SECT_STANDBY_M,          //17
        eFLASH_SECT_STANDBY_I,          //18
        eFLASH_SECT_MAX,                //19
}PP_FLASH_SECT_E;
```

B. Members

eFLASH_SECT_BL1	Bootloader section
eFLASH_SECT_BL2	Secondary bootloader section

eFLASH_SECT_IFW	ISP firmware section
eFLASH_SECT_IFW1	ISP backup firmware section
eFLASH_SECT_MFW	Main firmware section
eFLASH_SECT_MFW1	Main backup firmware section
eFLASH_SECT_SVM_LUT	SVM LUT section
eFLASH_SECT_CAR_IMG	Car image section
eFLASH_SECT_MENU	Menu setting value section
eFLASH_SECT_UI_IMG	UI image section
eFLASH_SECT_PGL_IMG	PGL image section
eFLASH_SECT_ISP_DATA	ISP setting value section
eFLASH_SECT_SENSOR_DATA	Sensor setting value section
eFLASH_SECT_CAM_CALIB_MAIN_TYPE	Calibration data section for core0
eFLASH_SECT_CAM_CALIB_SUB_TYPE	Calibration data section for core1

2) PP_FLASH_SECT_S

A. Syntax

```
typedef struct ppFLASH_SECT_S{
        PP_U32 u32Ver;
        PP_U32 u32FlashAddr;    // flash address
        PP_U32 u32LoadAddr;     // load address
        PP_U32 u32Size;         // size
}PP_FLASH_SECT_S;
```

B. Members

u32Ver	Version of a section
u32FlashAddr	The address of a section in flash memory
u32LoadAddr	The address where a section will be loaded
u32Size	The size of a section

3) PP_FLASH_HDR_S

A. Syntax

```
typedef struct ppFLASH_HDR_S
{
        PP_U32 u32Sign;
        PP_U32 u32FlashType;
        PP_U32 u32PageSize;
```

```
        PP_U32 u32PagesBlock;
        PP_U32 u32ClockDiv;
        PP_FLASH_SECT_S stSect[eFLASH_SECT_MAX];
        PP_U32 u32Rsv[0x3AA];
        PP_U32 u32Max;
        PP_U32 u32CRC16;
        PP_U32 u32Rsv1[3];
}PP_FLASH_HDR_S;
```

B. Members

u32Sign	Signature 0xABCD1234
u32FlashType	Flash type (0: NOR, 1: NAND nonFTL, 2: NAND FTL)
u32PageSize	Flash memory page size in byte
u32PagesBlock	Flash memory pages per block
u32ClockDiv	SPI clock divider used by mask rom
stSect	Section information structure
u32Rsv	Reserved area
u32Max	The number of sections
u32CRC16	CRC16 value of a header from 0x0 to 0xFE0
u32Rsv1	Reserved area

1.3.2. NOR/NAND flash low level APIs

1.3.2.1. Data Type

4) PP_FLASH_TYPE_E

A. Syntax

```
typedef enum ppFLASH_TYPE_E
{
        eFLASH_TYPE_NOR = 0,
        eFLASH_TYPE_NAND,
}PP_FLASH_TYPE_E;
```

B. Members

eFLASH_TYPE_NOR	NOR flash memory
eFLASH_TYPE_NAND	NAND flash memory

1.3.2.2. PPAPI_FLASH_Initialize

Prototype	PP_RESULT_E PPAPI_FLASH_Initialize(PP_U32 IN u32PageSize, PP_U32 IN u32EraseBlockSize, PP_FLASH_TYPE_E IN enFlashType)
Description	Initialize flash API
Argument	u32PageSize : flash memory page size u32EraseBlockSize : flash memory erase block size enFlashType : flash memory type
Return value	eSUCCESS: success else : fail
Example	

1.3.2.3. PPAPI_FLASH_Erase

Prototype	PP_VOID PPAPI_FLASH_Erase(PP_U32 IN u32Addr, PP_U32 IN u32Size)
Description	Erase flash memory
Argument	u32Addr : flash memory address
Return value	none
Example	

1.3.2.4. PPAPI_FLASH_Write

Prototype	PP_VOID PPAPI_FLASH_Write(PP_U8 IN *pu8Buf, PP_U32 IN u32Addr, PP_S32 IN s32Size)
Description	Write data to flash memory. This function erases flash memory first and write data to flash memory
Argument	pu8Buf : Pointer to data buffer to be written

	u32Addr : Flash memory address s32Size : Size of data to be written.
Return value	none
Example	

1.3.2.5. PPAPI_FLASH_WriteErasedBlock

Prototype	PP_VOID PPAPI_FLASH_WriteErasedBlock(PP_U8 IN *pu8Buf, PP_U32 IN u32Addr, PP_S32 IN s32Size)
Description	Write data to flash memory. This function does not erase flash memory. So user must erase flash memory before using this function.
Argument	pu8Buf : Pointer to data buffer to be written u32Addr : Flash memory address s32Size : Size of data to be written.
Return value	none
Example	

1.3.2.6. PPAPI_FLASH_Read

Prototype	PP_VOID PPAPI_FLASH_Read(PP_U8 OUT *pu8Buf, PP_U32 IN u32Addr, PP_S32 IN s32Size)
Description	Read data from flash memory.
Argument	pu8Buf : Pointer to data buffer to be read u32Addr : Flash memory address s32Size : Size of data to be read.
Return value	none
Example	

1.3.2.7. PPAPI_FLASH_ReadQDMA

Prototype	PP_VOID PPAPI_FLASH_ReadQDMA(PP_U8 OUT *pu8Buf, PP_U32 IN u32Addr, PP_S32 IN s32Size)

Description	Read data from flash memory using internal quad dma block for fast reading. This function does not return until reading is done. All arguments should be aligned to 16 bytes
Argument	pu8Buf : Pointer to data buffer to be read u32Addr : Flash memory address s32Size : Size of data to be read.
Return value	none
Example	

1.3.2.8. PPAPI_FLASH_ReadHeader

Prototype	PP_RESULT_E PPAPI_FLASH_ReadHeader(PP_VOID)
Description	Read flash header data from flash memory and save it to gstFlashHeader structure.
Argument	none
Return value	eSUCCESS: success else : fail
Example	

1.3.3. NAND flash FTL APIs

1.3.3.1. Data Type

1) PP_FTL_STATS_S

A. Syntax

```
typedef struct ppFTL_STATS_S
{
   PP_U32   u32BlocksTotal;
   PP_U32   u32BlocksReserved;
   PP_U32   u32BlocksBad;
   PP_U32   u32PagesPerBlock;
   PP_U32   u32BytesPerPages;;
}PP_FTL_STATS_S;
```

B. Members

u32BlocksTotal	The number of total blocks
u32BlocksReserved	The number of reserved blocks These blocks are used as non-ftl area and can be accessed flash low level APIs
u32BlocksBad	The number of bad blocks
u32PagesPerBlock	Pages per block
u32BytesPerPages	Bytes per page

1.3.3.2. PPAPI_FTL_Initialize

Prototype	PP_RESULT_E PPAPI_FTL_Initialize(PP_U8 IN u8DeviceType,PP_FTL_STATS_S* OUT pstStats)
Description	Initialize FTL
Argument	U8DeviceType: NAND_DEVICE_GD1G2G (GigaDevice flash memory) NAND_DEVICE_MT29FXG01 (Micron flash memory) NAND_DEVICE_W25NXG (Winbond flash memory) ● Device type can be obtained from gstFlashNandID.u8FTLDeviceID. gstFlashNandID structure is available after calling PPAPI_FLASH_Initialize function. pstStats : Statistics about an FTL volume.
Return value	none
Example	

1.3.3.3. PPAPI_FTL_Release

Prototype	PP_VOID PPAPI_FTL_Release(PP_VOID)
Description	Release resources
Argument	none.
Return value	none
Example	

1.3.3.4. PPAPI_FTL_Write

Prototype	PP_RESULT_E PPAPI_FTL_Write(PP_VOID* IN pvData, PP_U32 IN u32Addr, PP_S32 IN s32Size)
Description	Write data to NAND flash memory FTL area
Argument	pvData : Pointer to data buffer to be written. u32Addr : Flash memory FTL address. 0 means the beginning of the FTL area s32Size : Size of data to be written
Return value	none
Example	

1.3.3.5. PPAPI_FTL_Read

Prototype	PP_RESULT_E PPAPI_FTL_Read(PP_VOID* OUT pvData, PP_U32 IN u32Addr, PP_S32 IN s32Size)
Description	Read data from nand flash memory FTL area
Argument	pvData : Pointer to data buffer to be written. u32Addr : Flash memory FTL address. 0 means the beginning of the FTL area s32Size : Size of data to be written
Return value	none
Example	

1.3.3.6. PPAPI_FLASH_ReadFTLHeader

Prototype	PP_RESULT_E PPAPI_FLASH_ReadFTLHeader (PP_VOID)
Description	Read flash header data from NAND flash memory FTL area and save it to gstFlashFTLHeader structure.
Argument	none
Return value	eSUCCESS: success else : fail
Example	

2. Revision History

Version	Date	Description
V0.1	20180608	