Crystal Image through
Imaging Innovation

PIXELPLUS

SURROUND VIEW MONITORING SYSTEM

PI5008K File System User Guide

Rev 0.1

Last Update : 2018.06.08

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

Contents

1. File system

1.1. Introduction

This guide describes file system of PI5008 SVM SDK.

1.2. Layer description

Application -> PP_File_Layer ->HCC FAT LIB -> Porting Layer -> SPI -> SD/MMC Card

1.3. File system API

1.3.1. PPAPI_FATFS_Initialize

Prototype	PP_RESULT_E PPAPI_FATFS_Initialize(PP_VOID);
Description	Use this function to initialize the file system. Call it once at start-up.
Argument	None
Return value	eSUCCESS : Successful execution Else : See Error codes
Example	

1.3.2. PPAPI_FATFS_DeInitialize

Used to release resources allocated during the initialization of the file system

Prototype	PP_RESULT_E PPAPI_FATFS_DeInitialize(PP_VOID);
Description	Used to release resources allocated during the initialization of the file system
Argument	None
Return value	eSUCCESS : Successful execution eERROR_FATFS_BUSY A volume has not been deleted and this prevented the successful completion of this function.

Example	

1.3.3. PPAPI_FATFS_EnterFS(PP_VOID)

Used to create resources for the calling task in the file system and allocate a current working directory for that task.

Prototype	PP_RESULT_E PPAPI_FATFS_EnterFS(PP_VOID);
Description	Used to create resources for the calling task in the file system and allocate a current working directory for that task.
Argument	None
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.4. PPAPI_FATFS_ReleaseFS

Prototype	PP_RESULT_E PPAPI_FATFS_ReleaseFS(PP_VOID);
Description	Use this function to release the file system from calling task
Argument	None
Return value	None
Example	

1.3.5. PPAPI_FATFS_GetLastError

Prototype	PP_RESULT_E PPAPI_FATFS_GetLastError(PP_VOID);
Description	Use this function to return the last error code
Argument	None
Return value	Error code : The last error code
Example	int myopen() { PP_VOID *file;

	file = PPAPI_FATFS_Open("nofile.tst", "rb");
	if (!file)
	{
	int rc = PPAPI_FATFS_GetLastError();
	printf("fopen failed, errorcode:%d\n", rc);
	return rc;
	}
	return 0;
	}

1.3.6. PPAPI_FATFS_InitVolume

Prototype	PP_RESULT_E PPAPI_FATFS_InitVolume(PP_VOID);
Description	Used to initialize a volume.
Argument	None
Return value	eSUCCESS : Successful execution F_ERR_CARDREMOVED The volume has been successfully created; PPAPI_FATFS_InitVolume () does not need to be called again. When a card is inserted, the volume will be fully functional. Else See Error codes
Example	

1.3.7. PPAPI_FATFS_DelVolume

Prototype	PP_RESULT_E PPAPI_FATFS_DelVolume(PP_VOID);
Description	Used to delete an existing volume.
Argument	None
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.8. PPAPI_FATFS_CheckVolume

Prototype	PP_RESULT_E PPAPI_FATFS_CheckVolume(PP_VOID);
Description	Used to check the status of an initializing drive
Argument	None
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.9. PPAPI_FATFS_GetFreeSpace

Prototype	PP_RESULT_E PPAPI_FATFS_GetFreeSpace(FN_SPACE_S* OUT pstSpace);
Description	Use this function to fill a structure with information about the drive space usage: total space, free space, used space, and bad (damaged) size.
Argument	pstSpace : A pointer to the F_SPACE structure.
Return value	eSUCCESS : Successful execution Else See Error codes

Example	void info(void)
	{
	FN_SPACE_S space;
	int ret;
	/* Get free space on current drive */
	int ret = PPAPI_FATFS_GetFreeSpace (&space);
	if (!ret)
	{
	printf("There are:\
	%d bytes total,\
	%d bytes free,\
	%d bytes used,\
	%d bytes bad.",\
	space.total, space.free, space.used, space.bad);
	}
	else

	{
	printf("\nError %d reading drive\n", ret);
	}
	}

1.3.10. PPAPI_FATFS_GetLabel

Prototype	PP_RESULT_E PPAPI_FATFS_GetLabel(PP_CHAR* OUT szLabel, PP_S32 IN s32Len);
Description	Use this function to return the label as a function value.
Argument	szLabel : Where to copy the label. This should be able to hold an 11 character string s32Len : The length of the storage area
Return value	eSUCCESS : Successful execution Else See Error codes

Example	void getlabel(void)
	{
	PP_CHAR label[12];
	PP_RESULT_E result;
	result = PPAPI_FATFS_GetLabel (label, 12);
	if (result)
	{
	printf("Error on drive!");
	}
	else
	{
	printf("Drive is %s", label);
	}
	}

1.3.11. PPAPI_FATFS_SetLabel

Prototype	PP_RESULT_E PPAPI_FATFS_SetLabel(const PP_CHAR* IN szLabel);

Description	Use this function to set a volume label. The volume label should be an ASCII string with a maximum length of 11 characters
Argument	szLabel : Pointer to the null-terminated string to use.
Return value	eSUCCESS : Successful execution Else See Error codes

Example	void setlabel(void)
	{
	PP_RESULT_E result = PPAPI_FATFS_SetLabel("DRIVE 1");
	if (result)
	printf("Error on Drive");
	}

1.3.12. PPAPI_FATFS_MakeDir

Prototype	PP_RESULT_E PPAPI_FATFS_MakeDir(const PP_CHAR* IN szDirName)
Description	Use this function to create a new directory
Argument	szDirName : New directory name to create
Return value	eSUCCESS : Successful execution Else See Error codes

Example	void myfunc(void)
	{
	.
	.
	PPAPI_FATFS_MakeDir(“subfolder”); /*creating directory */
	PPAPI_FATFS_MakeDir (“subfolder/sub1”);
	PPAPI_FATFS_MakeDir (“subfolder/sub2”);
	PPAPI_FATFS_MakeDir (“subfolder/sub3”);
	.
	.
	}

1.3.13. PPAPI_FATFS_ChangeDir

Prototype	PP_RESULT_E PPAPI_FATFS_ChangeDir(const PP_CHAR* IN szDirName)
Description	Use this function to change the current working directory
Argument	szDirName : Null-terminated string containing name of directory to change to
Return value	eSUCCESS : Successful execution Else See Error codes

Example	void myfunc(void)
	{
	.
	.
	PPAPI_FATFS_MakeDir(“subfolder”);
	PPAPI_FATFS_ChangeDir(“subfolder”); /* change directory */
	PPAPI_FATFS_MakeDir (“sub2”);
	PPAPI_FATFS_ChangeDir(“..”); /* go up one directory level */
	PPAPI_FATFS_ChangeDir(“subfolder/sub2”); /* goto into sub2 dir */
	.
	.
	}

1.3.14. PPAPI_FATFS_RemoveDir

Prototype	PP_RESULT_E PPAPI_FATFS_RemoveDir(const PP_CHAR* IN szDirName)
Description	Use this function to remove a directory
Argument	szDirName : Name of directory to remove
Return value	eSUCCESS : Successful execution Else See Error codes
Example	void myfunc(void) { . .

	PPAPI_FATFS_MakeDir(“subfolder”); /*creating directories */ PPAPI_FATFS_MakeDir(“subfolder/sub1”); . . /* doing some work */ . PPAPI_FATFS_RemoveDir(“subfolder/sub1”); PPAPI_FATFS_RemoveDir(“subfolder”); /*removes directory */ . . }

1.3.15. PPAPI_FATFS_GetCWD

Prototype	PP_RESULT_E PPAPI_FATFS_GetCWD(PP_CHAR* OUT szBuffer, PP_S32 IN s32MaxLen)
Description	Use this function to get the current working directory on the current drive
Argument	szBuffer : Where to store current working directory string s32MaxLen : Length of the buffer
Return value	eSUCCESS : Successful execution Else See Error codes

Example	#define BUFFLEN F_MAXPATH+F_MAXNAME
	void myfunc(void)
	{
	char buffer[BUFFLEN];
	if (!PPAPI_FATFS_GetCWD(buffer, BUFFLEN))
	{
	printf (“current directory is %s”,buffer);
	}
	else
	{
	printf (“Drive Error”)
	}
	}

1.3.16. PPAPI_FATFS_Open

<table>
<tr><td>Prototype</td><td>PP_VOID* PPAPI_FATFS_Open(const PP_CHAR* IN szFileName, const PP_CHAR* IN szMode)</td></tr>
<tr><td>Description</td><td>Use this function to open a file. The following opening modes are allowed

<table>
<tr><td>Mode</td><td>Description</td></tr>
<tr><td>"r"</td><td>Open existing file for reading. The stream is positioned at the beginning of the file.</td></tr>
<tr><td>"r+"</td><td>Open existing file for reading and writing. The stream is positioned at the beginning of the file.</td></tr>
<tr><td>"w"</td><td>Truncate file to zero length or create the file for writing. The stream is positioned at the beginning of the file.</td></tr>
<tr><td>"w+"</td><td>+" Open a file for reading and writing. The file is created if it does not exist; otherwise, it is truncated. The stream is positioned at the beginning of the file.</td></tr>
<tr><td>"a"</td><td>Open for appending (writing to the end of file). The file is created if it does not exist. The stream is positioned at the end of the file.</td></tr>
<tr><td>"a+"</td><td>Open for reading and appending (writing to the end of file). The file is created if it does not exist. The stream is positioned at the end of the file.</td></tr>
</table></td></tr>
<tr><td>Argument</td><td>szFileName : File to be opened.
szMode : The opening mode (see above).</td></tr>
<tr><td>Return value</td><td>Pointer to the associated opened file handle, or zero if it could not be opened</td></tr>
<tr><td>Example</td><td>void myfunc(void)
{
PP_VOID *file;
PP_CHAR c;
file= PPAPI_FATFS_Open(“myfile.bin”,”r”);</td></tr>
</table>

	if (!file)
	{
	printf (“File cannot be opened!”);
	return;
	}
	PPAPI_FATFS_Read(&c,1,1,file); /*read 1 byte */
	printf (“’%c’ is read from file”,c);
	PPAPI_FATFS_Close(file);
	}

1.3.17. PPAPI_FATFS_Close

Prototype	PP_RESULT_E PPAPI_FATFS_Close(PP_VOID* IN pvHandle)
Description	Use this function to close a previously opened file
Argument	pvHandle : Handle of target file
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.18. PPAPI_FATFS_Read

Prototype	PP_S32 PPAPI_FATFS_Read(PP_VOID* IN pvBuf, PP_S32 IN s32Size, PP_S32 IN s32NumItem, PP_VOID* IN pvHandle)
Description	Use this function to read bytes from the current position in the target file. The file must be opened with “r”, "r+", "w+" or "a+".
Argument	pvBuf : Buffer to store data in s32Size : Size of items to be read s32NumItem : Number of items to be read. pvHandle : Handle of target file
Return value	Number of items read
Example	

1.3.19. PPAPI_FATFS_Write

Prototype	PP_S32 PPAPI_FATFS_Write(const PP_VOID* IN pvBuf, PP_S32 IN s32Size, PP_S32 IN s32NumItem, PP_VOID* IN pvHandle)
Description	Use this function to write data into a file at the current position. The file must be opened with "r+", “w”, “w+”, "a+" or “a”. The file pointer is moved forward by the number of bytes successfully written
Argument	pvBuf : Pointer to data to be written s32Size : Size of items to be written s32NumItem : Number of items to be written pvHandle : Handle of target file
Return value	Number of items written

Example	void myfunc(void)
	{
	PP_VOID *file;
	PP_CHAR *string=”ABC”;
	file= PPAPI_FATFS_Open (“myfile.bin”,”w”);
	if (!file)
	{
	printf (“File cannot be opened!”);
	return;
	}
	/* write 3 bytes */
	if(PPAPI_FATFS_Write (string,1,3,file)!=3)
	{
	printf (“Error: not all items written”);
	}
	PPAPI_FATFS_Close (file);
	}

1.3.20. PPAPI_FATFS_GetChar

Prototype	PP_CHAR PPAPI_FATFS_GetChar(PP_VOID* IN pvHandle)
Description	Use this function to read a character from the current position in the

	open target file.
Argument	pvHandle : Handle of open target file
Return value	value : Character read from the file -1 : Read failed

Example	int myreadfunc(PP_CHAR *filename, PP_CHAR *buffer, long buffsize)
	{
	PP_VOID *file= PPAPI_FATFS_Open (filename,”r”);
	while (buffsize--)
	{
	int ch;
	if((ch= PPAPI_FATFS_GetChar (file))== -1)
	break;
	*buffer++=ch;
	buffsize--;
	}
	PPAPI_FATFS_Close(file);
	return 0;
	}

1.3.21. PPAPI_FATFS_PutChar

Prototype	PP_CHAR PPAPI_FATFS_PutChar(PP_CHAR IN cChar, PP_VOID* IN pvHandle)
Description	Use this function to write a character to the specified open file at the current file position. The current file position is incremented.
Argument	cChar : Character to be written pvHandle : Handle of open target file
Return value	Value : Successfully written character -1 : write failed
Example	

1.3.22. PPAPI_FATFS_EOF

Prototype	PP_RESULT_E PPAPI_FATFS_EOF(PP_VOID* IN pvHandle)

Description	Use this function to check whether the current position in the open target file is the end of file (EOF).
Argument	pvHandle : Handle of open target file
Return value	0 : Not at end of file. Else : End of file or an error. See Error codes
Example	

1.3.23. PPAPI_FATFS_Tell

Prototype	PP_S32 PPAPI_FATFS_Tell(PP_VOID* IN pvHandle)
Description	Use this function to obtain the current read-write position in the open target file
Argument	pvHandle : Handle of open target file
Return value	Current read or write file position
Example	

1.3.24. PPAPI_FATFS_Seek

Prototype	PP_RESULT_E PPAPI_FATFS_Seek(PP_VOID* IN pvHandle, PP_S32 IN s32Offset, PP_S32 IN s32Whence)
Description	Use this function to move the stream position in the target file. The file must be open
Argument	pvHandle : Handle of open target file s32Offset : Byte position relative to s32Whence s32Whence : Where to calculate offset from The s32Whence parameter is one of the following: 0 – start of file. 1 – current position of file pointer. 2 – end of file.
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.25. PPAPI_FATFS_Delete

Prototype	PP_RESULT_E PPAPI_FATFS_Delete(const PP_CHAR* IN szFileName)
Description	Use this function to delete a file.
Argument	szFileName : Null-terminated string with the name of the file to be deleted, with or without path
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.26. PPAPI_FATFS_FindFirst

Prototype	PP_RESULT_E PPAPI_FATFS_FindFirst(const PP_CHAR* IN szFileName, PP_FNFIND_S* OUT pstFind)
Description	Use this function to find the first file or subdirectory in a specified directory. First call PPAPI_FATFS_FindFirst() and then, if the file is found, get the next file with PPAPI_FATFS_FindNext(). Files with the system attribute set will be ignored. **Note:** If this is called with "*.*" and it is not the root directory, then: ● the first entry found is ".", the current directory. ● the second entry found is "..", the parent directory.
Argument	szFileName : Name of file to find pstFind : Where to store the file information
Return value	eSUCCESS : Successful execution Else See Error codes

Example	void mydir(void)
	{
	PP_FNFIND_S find;
	if (!PPAPI_FATFS_FindFirst("subdir/*.*",&find))
	{
	do
	{
	printf ("filename:%s",find.filename);

	if (find.attr & FATFS_ATTR_DIR)
	{
	printf (“ directory\n”);
	}
	else
	{
	printf (“ size %d\n”,find.filesize);
	}
	} while (!PPAPI_FATFS_FindNext(&find));
	}
	}

1.3.27. PPAPI_FATFS_FindNext

Prototype	PP_RESULT_E PPAPI_FATFS_FindNext(PP_FNFIND_S* OUT pstFind)
Description	Use this function to find the next file or subdirectory in a specified directory
Argument	pstFind : File information (created by calling PPAPI_FATFS_FindFirst()).
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.28. PPAPI_FATFS_Move

Prototype	PP_RESULT_E PPAPI_FATFS_Move(const PP_CHAR* IN szFileName, const PP_CHAR* IN szNewName)
Description	Use this function to move a file or directory.The original file or directory is lost.
Argument	szFileName : File or directory name, with or without the path szNewName : New name of the file or directory, with or without the path
Return value	eSUCCESS : Successful execution

	Else See Error codes
Example	

1.3.29. PPAPI_FATFS_Rename

Prototype	PP_RESULT_E PPAPI_FATFS_Rename(const PP_CHAR* IN szOldName, const PP_CHAR* IN szNewName)
Description	Use this function to rename a file or directory. If a file or directory is read-only it cannot be renamed. If a file is open it cannot be renamed.
Argument	szOldName : File or directory name, with or without path szNewName : New name of file or directory
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.30. PPAPI_FATFS_GetAttr

Prototype	PP_RESULT_E PPAPI_FATFS_GetAttr(const PP_CHAR* IN szFileName, PP_U8* OUT pu8Attr)
Description	Use this function to get the attributes of a specified file
Argument	szFileName : The target file pu8Attr : Where to write the attributes Possible file attribute settings FATFS_ATTR_ARC - archived file or directory FATFS_ATTR_DIR - directory FATFS_ATTR_VOLUME - volume FATFS_ATTR_SYSTEM – system file or directory FATFS_ATTR_HIDDEN – hidden file or directory FATFS_ATTR_READONLY – read-only file or directory
Return value	eSUCCESS : Successful execution Else See Error codes
Example	void myfunc(void) {

	PP_U8 attr;
	/* find if myfile.txt is read only */
	if(!PPAPI_FATFS_GetAttr ("myfile.txt",&attr)
	{
	if(attr & FATFS_ATTR_DIR)
	printf("myfile.txt is read only");
	else
	printf("myfile.txt is writable");
	}
	else
	{
	printf("file not found");
	}
	}

1.3.31. PPAPI_FATFS_SetAttr

Prototype	PP_RESULT_E PPAPI_FATFS_SetAttr(const PP_CHAR* IN szFileName, PP_U8 IN u8Attr)
Description	Use this function to set the attributes of a file.
Argument	szFileName : The target file u8Attr : New attribute setting
Return value	eSUCCESS : Successful execution Else See Error codes

Example	void myfunc(void)
	{
	/* make myfile.txt read only and hidden */
	PPAPI_FATFS_SetAttr ("myfile.txt", FATFS_ATTR_READONLY |
	FATFS_ATTR_HIDDEN);
	}

1.3.32. PPAPI_FATFS_GetTimeDate

Prototype	PP_RESULT_E PPAPI_FATFS_GetTimeDate(const PP_CHAR* IN

<table>
<tr><td></td><td>szFileName, PP_U16* OUT pu16CTime, PP_U16* OUT pu16CDate)</td></tr>
<tr><td>Description</td><td>Use this function to get time and date information from a file or directory.</td></tr>
<tr><td>Argument</td><td>szFileName : Target file
pu16CTime : Where to store the creation time
pu16CData : Where to store the creation date
<table>
<tr><td>Argument</td><td>Valid values</td><td>Format</td></tr>
<tr><td>Day</td><td>0-31</td><td>(d & 001fH)</td></tr>
<tr><td>Month</td><td>1-12</td><td>((d & 01e0H) >> 5)</td></tr>
<tr><td>Years since 1980</td><td>0-119</td><td>((d & fe00H) >> 9)</td></tr>
</table>
<table>
<tr><td>Argument</td><td>Valid values</td><td>Format</td></tr>
<tr><td>2 second increments</td><td>0-30</td><td>(t & 001fH)</td></tr>
<tr><td>Minute</td><td>0-59</td><td>((t & 07e0H) >> 5)</td></tr>
<tr><td>Hour</td><td>0-23</td><td>((t & f800H) >> 11)</td></tr>
</table></td></tr>
<tr><td>Return value</td><td>eSUCCESS : Successful execution
Else See Error codes</td></tr>
<tr><td>Example</td><td></td></tr>
</table>

1.3.33. PPAPI_FATFS_SetTimeDate

<table>
<tr><td>Prototype</td><td>PP_RESULT_E PPAPI_FATFS_SetTimeDate(const PP_CHAR* IN szFileName, PP_U16 IN u16CTime, PP_U16 IN u16CDate)</td></tr>
<tr><td>Description</td><td>Use this function to set the time and date on a file or on a directory.</td></tr>
<tr><td>Argument</td><td>szFileName : Target file
u16CTime : Creation time of file or directory
u16CDate : Creation date of file or directory</td></tr>
<tr><td>Return value</td><td>eSUCCESS : Successful execution
Else See Error codes</td></tr>
<tr><td>Example</td><td>void myfunc(void)
{
PP_U16 ctime, cdate;
ctime = (15<<11)+(30<<5)+(22>>1);
/* 15:30:22 */</td></tr>
</table>

	cdate = ((2002-1980)<<9)+(11<<5)+(3);
	/* 2002.11.03. */
	PPAPI_FATFS_MakeDir “subfolder”); /* creating directory */
	PPAPI_FATFS_SetTimeDate(“subfolder”, ctime, cdate);
	}

1.3.34. PPAPI_FATFS_Flush

Prototype	PP_RESULT_E PPAPI_FATFS_Flush(PP_VOID* IN pvHandle)
Description	Use this function to flush an opened file to a storage medium. This is logically equivalent to performing a close and open on a file to ensure the data changed before the flush is committed to the medium.
Argument	pvHandle : Handle of target file
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.35. PPAPI_FATFS_Format

Prototype	PP_RESULT_E PPAPI_FATFS_Format(PP_VOID)
Description	Use this function to format the card as FAT32. If the media is not present, this function fails. If successful, all data are destroyed and any open files are closed.
Argument	none
Return value	eSUCCESS : Successful execution Else See Error codes
Example	

1.3.36. PPAPI_FATFS_Stat

Prototype	PP_RESULT_E PPAPI_FATFS_Stat(const PP_CHAR* IN szFileName, PP_FNSTAT_S* OUT pstStat)

Description	Use this function to get information about a file. This function retrieves information by filling the PP_FNSTAT_S structure passed to it. It sets the file size, creation time/date, last access date, modified time/date, and the drive number where the file is located.
Argument	szFileName : Name of the file pstStat : Pointer to F_STAT structure to be filled
Return value	eSUCCESS : Successful execution Else See Error codes

Example	void myfunc(void)
	{
	PP_FNSTAT_S stat;
	if (PPAPI_FATFS_Stat(“myfile.txt”,&stat))
	{
	printf (“error”);
	return;
	}
	printf (“filesize:%d”,stat.filesize);
	}

1.3.37. PPAPI_FATFS_FileLength

Prototype	PP_S32 PPAPI_FATFS_FileLength(const PP_CHAR* IN szFileName)
Description	Use this function to get the length of a file.
Argument	szFileName : File name, with or without path
Return value	Length of file. -1 : The requested file does not exist or has an error
Example	

1.3.38. Error codes

ERROR	Value	Meaning
eSUCCESS	0	Successful execution

eERROR_FATFS_INVALIDDRIVE	800	The specified drive does not exist.
eERROR_FATFS_NOTFORMATTED	801	The specified volume has not been formatted.
eERROR_FATFS_INVALIDDIR	802	The specified directory is invalid.
eERROR_FATFS_INVALIDNAME	803	The specified file name is invalid.
eERROR_FATFS_NOTFOUND	804	The file or directory could not be found.
eERROR_FATFS_DUPLICATED	805	The file or directory already exists.
eERROR_FATFS_NOMOREENTRY	806	The volume is full.
eERROR_FATFS_NOTOPEN	807	The file access function requires the file to be open.
eERROR_FATFS_EOF	808	End of file.
eERROR_FATFS_RESERVED	809	Not used.
eERROR_FATFS_NOTUSEABLE	810	Invalid parameters for PPAPI_FATFS_Seek().
eERROR_FATFS_LOCKED	811	The file has already been opened for writing/appending.
eERROR_FATFS_ACCESSDENIED	812	The necessary physical read and/or write functions are not present for this volume.
eERROR_FATFS_NOTEMPTY	813	The directory to be moved or deleted is not empty.
eERROR_FATFS_INITFUNC	814	No init function is available for a driver, or the function generates an error.
eERROR_FATFS_CARDREMOVED	815	The card has been removed.
eERROR_FATFS_ONDRIVE	816	Non-recoverable error on drive.
eERROR_FATFS_INVALIDSECTOR	817	A sector has developed an error.
eERROR_FATFS_READ	818	Error reading the volume.
eERROR_FATFS_WRITE	819	Error writing file to volume.
eERROR_FATFS_INVALIDMEDIA	820	Media not recognized.
eERROR_FATFS_BUSY	821	The caller could not obtain the semaphore within the expiry time.
eERROR_FATFS_WRITEPROTECT	822	The physical medium is write protected.
eERROR_FATFS_INVFATTYPE	823	The type of FAT is not recognized.
eERROR_FATFS_MEDIATOOSMALL	824	Media is too small for the format type requested.
eERROR_FATFS_MEDIATOOLARGE	825	Media is too large for the format type

		requested.
eERROR_FATFS_NOTSUPPSECTORSIZE	826	The sector size is not supported. The only supported sector size is 512 bytes.
eERROR_FATFS_UNKNOWN	827	Unspecified error has occurred.
eERROR_FATFS_DRVALREADYMNT	828	The drive is already mounted.
eERROR_FATFS_TOOLONGNAME	829	The name is too long.
eERROR_FATFS_NOTFORREAD	830	Not for read.
eERROR_FATFS_DELFUNC	831	The delete drive driver function failed.
eERROR_FATFS_ALLOCATION	832	Memory allocation faile
eERROR_FATFS_INVALIDPOS	833	An invalid position is selected.
eERROR_FATFS_NOMORETASK	834	All task entries are exhausted.
eERROR_FATFS_NOTAVAILABLE	835	The called function is not supported by the target volume.
eERROR_FATFS_TASKNOTFOUND	836	The caller's task identifier was not registered – normally because PPAPI_FATFS_EnterFS() has not been called.
eERROR_FATFS_UNUSABLE	837	The file system has become unusable, normally due to excessive error rates on the underlying media.
eERROR_FATFS_CRCERROR	838	A CRC error has been detected on the file.
eERROR_FATFS_CARDCHANGED	839	The card that was being accessed has been replaced with a different card.

2. Revision History

Version	Date	Description
V0.0	20180509	
V0.1	20180608	