ECE 423 API Reference

Introduction

This document provides the description for the API used in ECE 423 labs for SD card and video display interfaces.

Include Files

SD Card: `#include "ece423_sd/ece423_sd.h"`

Video Display: `#include "ece423_vid_ctl/ece423_vid_ctl.h"`

SD Card API

Types

- `FAT_HANDLE`: handle to FAT file system
- `FAT_BROWSE_HANDLE`: handle to browse FAT directory
- `FAT_FILE_HANDLE`: handle to FAT file
- `FILE_CONTEXT`: structure to hold the information about the file

Functions

SDLIB_Init()

- Prototype: `bool SDLIB_Init(int base_addr)`
- Description: Initiate the SD card.
- Arguments: The address of the SD card device (`base_addr`), e.g. `SD_CONT_BASE` (refer to `system.h` in your BSP project).
- Return: `TRUE` when the SD card is initialized successfully.

Fat_Mount()

- Prototype: `FAT_HANDLE FAT_Mount()`
- Description: Mount the FAT file system.
- Arguments: None.
- Return: Handle to the FAT file system. If the mount failed, it returns `0`.

Fat_Unmount()

- Prototype: `void FAT_Unmount(FAT_HANDLE Fat)`
- Description: Unmount the FAT file system.
- Arguments: Handle to the mounted FAT system.
- Return: None.

Fat_FileBrowseBegin()

- Prototype: `bool FAT_FileBrowseBegin(FAT_HANDLE hFAT, FAT_BROWSE_HANDLE *pFatBrowseHandle)`
- Description: Start browsing the file system.
- Arguments: Handle to the file system (`hFAT`) - Pointer to a browsing handle (`pFatBrowseHandle`).
- Return: FALSE if the FAT system has not been mounted.

Fat_FileBrowseNext()

- Prototype: `bool Fat_FileBrowseNext(FAT_BROWSE_HANDLE *pFatBrowseHandle, FILE_CONTEXT *pFileContext)`
- Description: Browse the next file in the directory.
- Arguments: Pointer to browse handle (`pFatBrowseHandle`) – Pointer to file context (`pFileContext`)
- Return: `FALSE` if an error occurred.

Fat_CheckExtension()

- Prototype: `bool Fat_CheckExtension(FILE_CONTEXT *pFileContext, char* ext)`
- Description: Check the extension of the file.
- Arguments: Pointer to the file context (`pFileContext`) – An extension to compare against (`ext`).
- Return: TRUE if the file has the extension (`ext`).

Fat_GetFileName()

- Prototype: `char* Fat_GetFileName(FILE_CONTEXT *pFileContext)`
- Description: Get the file name.
- Arguments: Pointer to the file context (`pFileContext`).
- Return: The file name.

Fat_FileOpen()

- Prototype: `FAT_FILE_HANDLE Fat_FileOpen(FAT_HANDLE Fat, const char *pFilename)`
- Description: Open the file using its name.
- Arguments: Handle to the FAT system (`Fat`) – File name (`pFilename`).
- Return: Handle to the file, if the file is not found it returns.

Fat_FileRead()

- Prototype: `bool Fat_FileRead(FAT_FILE_HANDLE hFileHandle, void *pBuffer, const int nBufferSize)`
- Description: Read from the file to the memory.
- Arguments: Handle to the FAT file (`hFatHandle`) – Target memory address (`pBuffer`) – Number of bytes (`nBufferSize`).
- Return: `FALSE` if an error occurred.

Fat_ FileSeek()

- Prototype: `bool Fat_FileSeek(FAT_FILE_HANDLE hFileHandle, const FAT_SEEK_POS SeekPos, const int nOffset)`
- Description: Seek the file.
- Arguments: Handle to the FAT file (`hFatHandle`) – Position to seek from (`SeekPos`): `FILE_SEEK_BEGIN/FILE_SEEK_CURRENT/FILE_SEEK_END` – Number of bytes (`nBufferSize`).
- Return: `FALSE` if an error occurred.

Fat_FileClose()

- Prototype: `void Fat_FileClose(FAT_FILE_HANDLE hFileHandle)`
- Description: Close the file.
- Arguments: Handle to the file (`hFileHandle`).
- Return: None.

Video Display API

Types

`ece423_video_display`: structure for the display properties

Functions

ece423_video_display_init()

- Prototype: `ece423_video_display* ece423_video_display_init( char* sgdma_name, int width, int height, int num_buffers)`
- Description: Initialize the video display.
- Arguments: Video DMA name (`sgdma_name`), e.g. `VIDEO_DMA_NAME` (refer to `system.h` in your BSP project) – Video width (`width`) – Video height (`height`) – Number of buffered frames (`num_buffers`).
- Return: Pointer to video display structure, `0` when it fails.

ece423_video_display_buffer_is_available()

- Prototype: `int ece423_video_display_buffer_is_available(ece423_video_display* display)`
- Description: Check if there is an available buffer to write a new frame.
- Arguments: Video display (`display`).
- Return: `0` when the buffer is available.

ece423_video_display_register_written_buffer()

- Prototype: `void ece423_video_display_register_written_buffer(ece423_video_display * display)`
- Description: Register the last written frame to be displayed.
- Arguments: Video display (`display`).
- Return: None.

ece423_video_display_switch_frames()

- Prototype: `void ece423_video_display_switch_frames(ece423_video_display* display)`
- Description: Switch to the next frame to be displayed.
- Arguments: Video display (`display`).
- Return: None.

ece423_video_display_get_buffer()

- Prototype: `alt_u32* ece423_video_display_get_buffer( ece423_video_display* display)`
- Description: Get a pointer to the current buffer to be written.
- Arguments: Video display (`display`).
- Return: Pointer to the frame buffer.
