**Title: STM32 Analog Signal Recording and SD Card Storage Project**

**Description:**

Are you a skilled embedded systems developer with experience in STM32 microcontrollers and signal processing? We have an exciting project that involves capturing and recording analog signals using an STM32 microcontroller and storing them onto an SD card. If you're passionate about creating efficient and reliable embedded solutions, this project is for you.

**Project Overview:**

In this project, we are working with an STM32 microcontroller that receives two analog signals: a Power Signal (PA0) and a Voltage Control Signal (PA1). The PA1 signal is a 1kHz Sawtooth waveform. Our goal is to develop a system that utilizes the PA1 signal as a trigger to initiate the recording of the PA0 signal at the beginning of each cycle. The SD card has been interfaced with the STM32. The biggest part of this project is debugging; hence you will have access to the hardware and an oscilloscope showing the expected signals.

**Project Requirements:**

1. **Signal Triggering:** The project involves using the 1kHz Sawtooth signal (PA1) as a trigger to initiate the recording of the PA0 signal. This synchronization ensures that the recording captures the relevant data during each cycle.
2. **Sampling and Recording:** The PA0 signal should be recorded at a rate of 4MSa/s (Mega-samples per second). Each cycle of the PA0 signal consists of 4096 samples. It's crucial to design an efficient sampling and recording mechanism to ensure accurate data capture.
3. **Storage:** The recorded PA0 signal data should be stored onto an SD card. The STM32 microcontroller has an SPI2 interface connected to the SD card for data storage. Only the PA0 signal data needs to be saved, utilizing DMA to save CPU power.
4. **Peak Frequency Calculation and UART Transmission:** After each cycle of PA0 signal data is saved onto the SD card, a peak detection calculation will be performed on the recorded data. This calculation will identify the peak frequency of the signal. The calculated peak frequency will then be sent to the UART interface for communication.

**Additional Information:**

The peak frequency calculation is carried out after the data from each PA0 cycle is saved in the SD card. This approach ensures that accurate and relevant data is used for the calculation. As part of the design, the project may involve utilizing the FreeRTOS operating system to manage tasks efficiently and maintain synchronization between data recording, peak frequency calculation, and UART communication.

**Skills Required:**

* Strong proficiency in STM32 microcontroller programming and interfacing.
* Experience in analog signal processing and trigger mechanisms.
* Familiarity with STM32 clock, sampling time, real-time data acquisition, DMA & TMM, FREERTOS, and storage onto SD cards.

**Deliverables:**

1. STM32 firmware that captures and records PA0 signal data triggered by the PA1 signal.
2. Efficient algorithm for managing data storage onto the SD card while optimizing space and write cycles.
3. Peak frequency calculation and UART transmission implementation.
4. A screen with a graph on it

   Description automatically generatedA white electronic device with a screen

   Description automatically generatedIntegration of FreeRTOS for managing concurrent tasks.

The Yellow signal is the PA1. The frequency is around 840Hz. We want this signal to act as a triggerfor data acquisition process, just like in the picture, where the Blue signal is saved to the SD Card every Cycle. The cycle time is about about 1.190ms. You are to determine if the trigger should be at the arising edge or falling edge of the Control Signal(PA1: Yellow Signal). Since we are using 12bits, we expect all 4096 samples to be acquired per trigger (i.e. each period of the saw-tooth (Yellow) signal). The detected peak of the blue signal need to be sent to the computer through the Uart2.

For the SD Card, the SPI2 is used to connect the card as follows: SPI2\_SCK(pb13), SPI2\_MISO (PB14), SPI2\_MOSI (PB15), GPIO OUT(PB1). You can rename the GPIO TO SD\_CS(PB1). Please be aware that the STM32F44RE does not use SDIO, it used the “user define”. Finally, all other required configurations including the clock, Prescalar, FreeRTOS, DMA and TMM will be left to you to decide.