GENERAL

and the stm32 digital signal processing ukhas

Published: September 19, 2026 | Index ID: #-

Introduction

High altitude ballooning has become an exciting platform for experimenting with embedded electronics, radio communications, and data acquisition. The UK High Altitude Space Society (UKHAS) provides a community and a set of guidelines for designing reliable payloads that can survive the extreme conditions of the stratosphere. Among the many hardware choices available, the STM32 family of microcontrollers stands out for its blend of performance, low power consumption, and robust software ecosystem. In this article we explore how the STM32 can be leveraged for digital signal processing (DSP) tasks in UKHAS projects, covering architecture, libraries, practical design tips, and real world examples.

Why STM32 for UKHAS Payloads?

UKHAS payloads often require:

- Real time data filtering and compression to reduce telemetry bandwidth.
- Signal demodulation for telemetry receivers.
- Fast, low latency processing for attitude control or sensor fusion.
- Robustness to temperature variations from –80 /°C to +50 /°C.
- Low power consumption to extend mission duration.

The STM32 series, particularly the Cortex M4 and M7 cores, delivers on all these fronts. With a dedicated floating point unit, hardware multiply accumulate (MAC) instructions, and a rich set of peripherals, it is ideally suited for embedded DSP.

Understanding the STM32 Architecture

Core Families

STM32 microcontrollers are grouped into several families, each tailored for a specific application domain:

- STM32F0 – Entry level, cost effective, limited DSP features.
- STM32F1 – General purpose, moderate performance.
- STM32F3 – Mixed signal with enhanced ADC and DAC.
- STM32F4 – Cortex M4 core with floating point unit, ideal for DSP.
- STM32F7 – Cortex M7 core, higher clock speeds, more memory.
- STM32H7 – Dual core Cortex M7/M4, highest performance.

For most UKHAS projects, the STM32F4 or STM32F7 families provide the optimal balance between performance and power consumption.

Key DSP Related Features

1. Floating Point Unit (FPU) – Enables fast double precision calculations, essential for high quality filtering.
2. Multiply Accumulate (MAC) – Accelerates convolution and FIR filter operations.
3. DSP Extensions – Includes instructions like VSQRT, VRECIP, and VQDSUB for efficient math.
4. High Performance Memory – SRAM up to 512 /KB, Flash up to 2 /MB, and optional external memory interfaces.

5. Peripheral Integration – Dedicated DMA controllers, ADCs with high sampling rates, and UART/SPI/I2C interfaces for sensor data.

The CMSIS DSP Library

The Cortex Microcontroller Software Interface Standard (CMSIS) provides a vendor agnostic API for ARM Cortex M processors. Its **CMSIS DSP** library offers a comprehensive set of optimized functions for:

- Fast Fourier Transform (FFT)
- Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filtering
- Signal statistics and correlation
- Mathematical utilities (matrix, vector operations)

Because the library is written in C and heavily optimized for ARM instruction sets, it runs efficiently on STM32 cores, making it a go to choice for UKHAS payloads that need real time DSP.

Installing CMSIS DSP on STM32CubeIDE

1. Open STM32CubeIDE and create a new STM32 project.
2. In the Project Manager tab, enable the CMSIS DSP component under Software Packs.
3. Generate code. The IDE automatically adds the necessary include paths and linker settings.
4. Include #include "arm_math.h" in your source files to access DSP functions.

Designing a UKHAS Payload with STM32 DSP

Mission Overview

Consider a typical UKHAS payload that measures atmospheric pressure, temperature, and collects high frequency audio data for post flight analysis. The payload must:

- Sample sensors at 1 /kHz.
- Filter out 50/60 /Hz mains interference.
- Compress data using a lightweight algorithm.
- Transmit telemetry at 433 /MHz with a bandwidth of 4 /kHz.

Hardware Selection

- Microcontroller: STM32F411CEU6 (Cortex M4, 100 /MHz, 512 /KB Flash, 96 /KB SRAM).
- ADC: 12 bit, 2 /MSPS, 6 channel (for sensors).
- Audio Interface: I2S DAC/ADC pair (e.g., PCM5102).
- Radio: RFM95W (LoRa) or a custom 433 /MHz transceiver.
- Power: 3.7 /V Li Po battery with voltage regulator.

Software Architecture

1. HAL Layer – Handles low level peripheral configuration.
2. FreeRTOS – Provides task scheduling for sensor acquisition, filtering, and transmission.
3. CMSIS DSP – Implements filtering and FFT.
4. Compression – Simple run length encoding or LZ4 for telemetry.
5. Telemetry – Packet framing, CRC, and error checking.

Signal Filtering Example

Below is a concise example of implementing a 4th order Butterworth low pass filter using CMSIS DSP. The filter removes high frequency noise while preserving the 1 /kHz sensor data.

#include "arm_math.h"\n\n#define NUM_TAPS 9\nfloat32_t coeffs[NUM_TAPS] = {\n -0.0044f, -0.0069f, -0.0076f, -0.0036f,\n 0.0052f, 0.0130f, 0.0139f, 0.0074f,\n 0.0000f\n};\n\nfloat32_t state[NUM_TAPS + 1];\narm_fir_instance_f32 S;\n\nvoid initFilter(void) {\n arm_fir_init_f32(&S, NUM_TAPS, coeffs, state, 1);\n}\n\nvoid filterSample(float32_t *input, float32_t *output) {\n arm_fir_f32(&S, input, output, 1);\n}\n\n\nFFT for Audio Analysis

During the flight, the payload can perform an on the fly FFT to detect specific acoustic signatures (e.g., rotor blades). The following snippet demonstrates a 256 point FFT using CMSIS DSP.

#include "arm_math.h"\n\n#define FFT_SIZE 256\nfloat32_t fftInput[FFT_SIZE];\nfloat32_t fftOutput[FFT_SIZE];\n\narm_cfft_radix4_instance_f32 fftInst;\n\nvoid initFFT(void) {\n arm_cfft_radix4_init_f32(&fftInst, FFT_SIZE, 0, 1);\n}\n\nvoid computeFFT(void) {\n arm_cfft_radix4_f32(&fftInst, fftInput);\n arm_cmplx_mag_f32(fftInput, fftOutput, FFT_SIZE);\n}\n\n\nTelemetry Packaging

To fit within the 4 kHz bandwidth, the payload compresses sensor readings using a simple delta encoding algorithm. Each packet contains:

- Timestamp (24 /bits)
- Sensor values (16 /bits each)
- CRC (16 /bits)

The resulting packet size averages 48 /bits, comfortably below the bandwidth limit.

Thermal Management for High Altitude Flight

At 30 km altitude, temperatures can drop below –80 °C. STM32 devices are rated down to –40 °C, so additional measures are required:

- Enclosure Insulation – Use aerogel or multi layer insulation (MLI) to reduce heat loss.
- Active Heating – A small resistive heater controlled by a temperature sensor can keep the MCU above its operating minimum.
- Component Placement – Place heat generating components (e.g., radio transceiver) near the enclosure surface to facilitate heat dissipation.
- Power Budgeting – Heaters consume power; balance with battery capacity.

Power Management Strategies

Efficient power usage extends flight time. Key strategies include:

1. Dynamic Frequency Scaling – Reduce the core clock when the payload is idle.
2. Peripheral Sleep Modes – Disable unused peripherals (e.g., I2C, UART) during low activity periods.
3. Battery Monitoring – Use the STM32's built in battery charger IC or an external fuel gauge to monitor state of charge.
4. Low Power Modes – Enter STOP or STANDBY modes when telemetry is not required.

Debugging and Validation

In Flight Telemetry Logging

During a test flight, real time telemetry can be streamed to a ground station for monitoring. Include a status packet that reports:

- Battery voltage
- Temperature readings
- Processor load
- Error flags

This helps identify issues such as overheating or memory corruption before the payload lands.

Post Flight Data Analysis

After recovery, compare the raw sensor data with the filtered output to verify filter performance. Plot the frequency spectrum of the audio signal to ensure the FFT implementation is correct.

Case Study: UKHAS 2023 Atmospheric Sounding

In the UKHAS 2023 competition, a team built a payload that used an STM32F746ZG (Cortex M7) to process real time GPS data and perform adaptive filtering of atmospheric pressure readings. The payload achieved a 30 % reduction in telemetry bandwidth by employing CMSIS DSP for FIR filtering and a custom run length encoder.

Key takeaways from this project:

Baca Juga (Artikel Terkait)

- **advanced engineering electromagnetics solution manual edition hardcover by balanis constantine a published by wiley**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-solution-manual-edition-hardcover-by-balanis-constantine-a-published-by-wiley.pdf
- **advanced engineering electromagnetics solutions manual**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-solutions-manual.pdf
- **advanced engineering electromagnetics wiley 1989 grading**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-wiley-1989-grading.pdf
- **advanced engineering fluid mechanics by biswas**
  URL: http://djjsearch.com/advanced-engineering-fluid-mechanics-by-biswas.pdf