Aidan Johnson
16 January 2018
Dissection of C Programs

DT7816 Calibration (`dt7816-calibration`)

`main.c`: calibrates analogue input or output.

1. Global constants (default):
  1.1. Sample rate: 200000 Hz
  1.2. Number of samples: 1000
  1.3. Maximum offset count: 10
  1.4. Nominal +9 V nominal count: 29358
  1.5. +9 V maximum/minimum count: +293/−293
  1.6. Wiper minimum: 0
  1.7. Wiper maximum: 255
  1.8. Wiper middle: 128
2. Acquires 1 buffer, returning its average value (for 8 channels).
3. Analogue input (AIN):
  3.1. Sets wiper of analogue input gain potentiometer.
  3.2. Initialisation:
    3.2.1. Creates 8-channel mask (for AIN0-AIN7).
    3.2.2. Configures sample rate with internal clock.
    3.2.3. Sets channel gain, DC couples, and then turns off current source.
    3.2.4. Configures software trigger.
    3.2.5. Buffer sized to contain about the specified number of samples.
  3.3. Calibration:
    3.3.1. Gain potentiometer wiper (AIN0) set to middle.
    3.3.2. Analogue input 0 through 7 shorted (AIN0-AIN7) to ground (AGND), then measures offset.
    3.3.3. Applies +9 V to AIN0, reading and averaging 1 buffer (must be within nominal maximum/minimum range).
    3.3.4. Gain potentiometer adjusted with the wiper, reading and averaging 1 buffer until the average equals the nominal +9 V count plus the offset.
    3.3.5. Saves the gain potentiometer wiper as a calibration constant.
4. Analogue output (AOUT):
  4.1. Adjusts wiper for either: (1) offset; (2) gain potentiometer.
  4.2. Calibration:

4.2.1. Sets gain potentiometer wiper to middle.

4.2.2. Outputs −9.9 V, directing user to measure output voltage with a digital voltmeter.

4.2.3. Outputs +9.9 V, again directing user to measure output voltage.

4.2.4. Saves gain potentiometer calibration after user adjusts voltage to 19.8 V plus reading in *3.2.3.*

4.2.5. Saves offset potentiometer calibration after user adjusts output voltage to be with in ±20 μV.

5. Prints gain calibration of the 8 analogue inputs (failure or success).

6. Prints gain calibration of the 2 analogue outputs.

6.1. Prints gain potentiometer wiper (*3.1*).

6.2. Prints gain potentiometer user calibration (*3.2*).

6.3. Prints offset potentiometer wiper (*3.1*).

6.4. Prints offset potentiometer user calibration (*3.2*).

Signal Analyser (`sig-analyzer`)

`main.c`: web application that acquires data from the board's analogue input channels, performs the FFT, and saves the results to an embedded web server that can be accessed in a web browser.
`mongoose.c`: header and library used by `main.c` to create HTTP web server and sockets.

1. Global constants (default):

1.1. Number of samples: 1024

1.2. Number of buffers: 1

1.3. Trigger level: 0.0

1.4. Gain: 1 or *log*(10)

1.5. Listening port: 8080

1.6. Sampling rate: 400000 Hz

2. Initialises data types:

2.1. Web socket.

2.2. Buffer (number of samples and values).

2.3. Server parameters (sampling rate, number of samples, and buffer).

2.4. FFT window (coherent gain and width sized for number of samples).

3. Initialisation:

  3.1. Establishes terminal command line arguments.

  3.2. Creates and initialises the web server.

  3.3. Creates and opens input stream; opens analogue input.

  3.4. Returns sampling rate.

  3.5. Writes channel mask; sets channel gain, DC coupling, and turns current source off.

  3.6. Configures trigger on AIN0.

  3.7. Creates and initialises AIO structures; sizes buffer for each channel to hold the number of specified samples.

  3.8. Creates buffer to store raw values that are converted to V.

  3.9. Fast Fourier Transform (FFT):

    3.9.1. Creates buffer to store FFT spectral amplitudes

    3.9.2. Receives complex output from FFTW

4. Data acquisition (*ad infinitum*):

  4.1. Raw values from 16-bit input channels converted to voltage after stream ends with buffer completion.

  4.2. For all web connections, voltage data pushed to WebSocket clients.

  4.3. Applies window function to acquired data.

  4.4. Computes FFT: spectral magnitudes.

  4.5. FFT data pushed to WebSocket clients.

  4.6. Delay to prevent overwhelming web browser.

5. Handlers for:

  5.1. HTTP requests.

  5.2. Sending a `String` identifying the board.

  5.3. Sending sampling rate.

  5.4. Sending FFT data.

  5.5. Sending raw waveform data.

6. Function for creating window function used in *4.3*.