MoToScope User Manual

Author: Lauren Kosub
Date: December 17th, 2019

Abstract: *The MoToScope software is running on the STM32l476 Discovery kit which provides backend support for the digital storage oscilloscope. A user can fetch and process waveform data using MoToScope. MotoScope is controlled from the PC USB Port using the Micro-on-Tether (MoT) command protocol. The MotoScope allows users to initialize their own Oscilloscope with their own values such that they can gather wave data the way they want to. This manual serves to help future software engineers who wish to write PC software for the MotoScope user interface.*

Table of Contents

Communicating with the MoToScope

The MoToScope is run via MoT Commands in the following format:
:<Device ID (1 byte)><function command (1 byte) ><Arguments>

The semicolon specifies the start of a command. The device ID byte is equal to three (0x03) because it is the third device in the MoT admin device table. Each of the commands will also be followed by a checksum. This checksum can be calculated using the hex2MoTcmd.c script. Within this file, these checksums will be represented by XX. When creating your own commands, be sure to run the command inputs in the hex2MoTcmd.c script and append the checksum to your input command. All commands end in a new line character.

The function byte is a number between 0 and 5 and maps to a specific functionality within the MoToScope.

Configuring the MoToScope

In order to use the MoToScope, a user must first send the initialization command. The function code 0x0 **initializes the scope** and does not take any arguments. The command a user would send to the device via MoT would be: "*:0300XX".* By running the configuration command, the program sets up all of its hardware components (TIM2, ADC, DMA) such that it is ready to process data. The default configuration has TIM2 as the trigger for the ADC. The DMA is triggered by the completion of the ADC. This triggering from TIM2 to ADC to DMA initially runs continuously in a repetitive waveform sampling mode. The data being collected is recorded by the software in main through the adc_data value. adc_data is a circular buffer of size 2048. This can be configured later on. A user can read waveform data points from the adc_data value and perform additional computations on this data as discussed in the analysis section of this manual.

Oscilloscope Variables

The MoToScope program is focused around the following variables: mode, trigger, trigger delay, sample size, report size, and sample rate.

The **mode** of the MoToScope specifies if the Oscilloscope in scan through mode, repetitive waveform sampling mode, or single-shot mode. Scan through mode allows the user to start and stop the sampling of the oscilloscope at will as well as step through the values of the waveform in chunks of size sample size at 00 their own pace. This is configured to work with the PA1 as trigger so that the user can more accurately trigger. Repetitive waveform sampling mode is the default mode and it simply grabs waveform data continuously and reports that data to main through the adc_data variable. Single-shot mode wait for the trigger interrupt and then take a single sample of size sample size and reports it to main through adc_data. I.E. the trigger is started and then immediately stopped after one round of data collection. This enables the user to process one chunk of data and only one chunk of data.

The **trigger** of the MoToScope can either be TIM2 or PA1. TIM2 in an internal trigger while PA1 is an external trigger. The trigger enables the timebase to start its scan at the same point on each repetition of the waveform.

The **trigger delay** is the number of seconds the trigger should wait before enabling the timebase to start its scan. With PA1, this is always zero (future work would include linking PA1 with a timer such that it can also have a trigger delay or introducing a new timer to handle trigger delays independent of the type of trigger used). With TIM2, the trigger delay is appending to the sample rate such that the trigger is enabled every (sample rate + trigger delay).

The **sample size** specifies how much data is collected every time the trigger is enabled. This value specifies the size of the adc_data circular buffer.

The **report size** is how much data is actually reported to the user. Due to time constraints with project implementation, the report size is equal to the sample size. For future work, one could make it such that you can configure sample size and report size independently.

The **sample rate** is how often adc_data is updated by the ADC/DMA. The ADC/DMA are activated when the trigger interrupts, so the sample rate is equal to the count value of the timer (TIM2). PA1 does not have a sample rate, since its interrupt is caused by external behavior.

The **watchdog** is internal to the system and is stored within main. The watchdog ensures that the read data does not exceed certain bounds.

Modifying Variables within the MoToScope

The function code 0x1 **modifies a setting within the scope**. It has an additional 1 byte argument to specify which variable to modify and a four byte value that represents the new value's data.

The byte argument 0x0 modifies the oscilloscope's mode. The next argument represents which mode the oscilloscope should switch to. The new mode (next argument) can either be 0x0, 0x1, or 0x2. 0x0 is scan through mode, 0x1 is repetitive waveform sampling mode, and 0x2 is single-shot waveform sampling mode. An example of modifying the mode to be scan through mode would be ":030100XX". An example of modifying mode to be repetitive waveform sampling mode would be ":030101XX". An example of modifying mode to be single-shot waveform sampling mode would be ":030102XX".

The byte argument 0x1 modifies the trigger type. This command takes in another 1 byte parameter that specifies which trigger to use. If the parameter is 0x0 then use the TIM2 for the trigger. If the parameter is 0x1 then use the PA1 for the trigger. The command for modifying the trigger would be formatted as follows: ":030101<new trigger>"

The byte argument 0x2 modifies the trigger delay. This command takes in another 4 byte argument that represents the new trigger delay value. The command for modifying the trigger delay would be formatted as follows: ":030102<new trigger delay>"

The byte argument 0x3 modifies the sample size. This command takes in another 4 byte argument that represents the new sample size value. The command for modifying the trigger delay would be formatted as follows: ":030103<new sample size>"

The byte argument 0x4 modifies the report size. This command takes in another 4 byte argument that represents the new report size value.The command for modifying the trigger delay would be formatted as follows: ":030104<new report size>"

The byte argument is 0x5 modifies the sample rate. This command takes in another 4 byte argument that represents the new sample rate value.The command for modifying the trigger delay would be formatted as follows: ":030105<new sample rate>"

Running the MoToScope

The MoToScope takes in commands via MoT, so to run the MoToScope you must first send it an initialization command. After you can either configure values with the modify commands or start data processing with the scan through, repetitive waveform sampling, or single-shot commands.

The function code 0x3 **runs the scan through mod**e on the MoToScope. It takes in a 1 byte parameter that specifies if the MoToScope should start, stop, or step.

If the parameter byte is 0x0, the MoToScope starts collecting data and continues running until told to stop. This command would be formatted as ":030300XX".

If the parameter byte is 0x1, the MoToScope stops collecting data and waits until it is either told to start again or step. This command would be formatted as ":030301XX".

If the parameter byte is 0x2, the MoToScope collects sample size amount of data and then stops. This is considered a step. Another call to step restart the data collection and then stop it once more. This way you are "stepping" through the data one chunk at a time. This command would be formatted as ":030303XX".

The function code 0x3 **runs the repetitive waveform sampling mode**. This command does not take any inputs and will continuously run and report data back to main via adc_data. This command would be formatted as ":0303XX"

The function code 0x4 **runs the single-shot waveform sampling mode**. This command reads in sample size amount of data and then quits. It takes no arguments. This command would be formatted as ":0304XX"

Analyzing Data with the MoToScope

Data is passed into main through adc_data. A user should use adc_data to perform certain calculations on the waveform data within adc_data in a C program. Users can write their own C functions that take in a uint8_t array and run different monitoring scripts or analysis scripts on the waveform. In this project there are examples of such processing operations already implemented.

The function waveformAverage(uint8_t *) takes in the adc_data and determines what the average value of the waveform is as well as the root-mean-square. This information is stored in a struct and returned to the program. The function checkBounds(uint8_t *) takes in the adc_data and monitors it to ensure that it has not gone over or under a certain limit. The limits (HIGH and LOW) are #defined values within the program that can be modified as necessary. The Red LED is enabled in main whenever the waveform surpases expected bounds defined in software. The surpassing bounds warning is also sent to MoT to be displayed back to the user. Users can add assessment functionality by writing more processing functions.