Demonstration software

Overview

SCF4-SDK comes with open-sourced command line and GUI sample programs for rapid controller evaluation. A simple control software example is provided for testing and demonstration. Software is given "as is" to help with getting started and testing.

Source code is maintained on GitHub

There are several control software branches maintained in parallel for different cameras. See table below for details

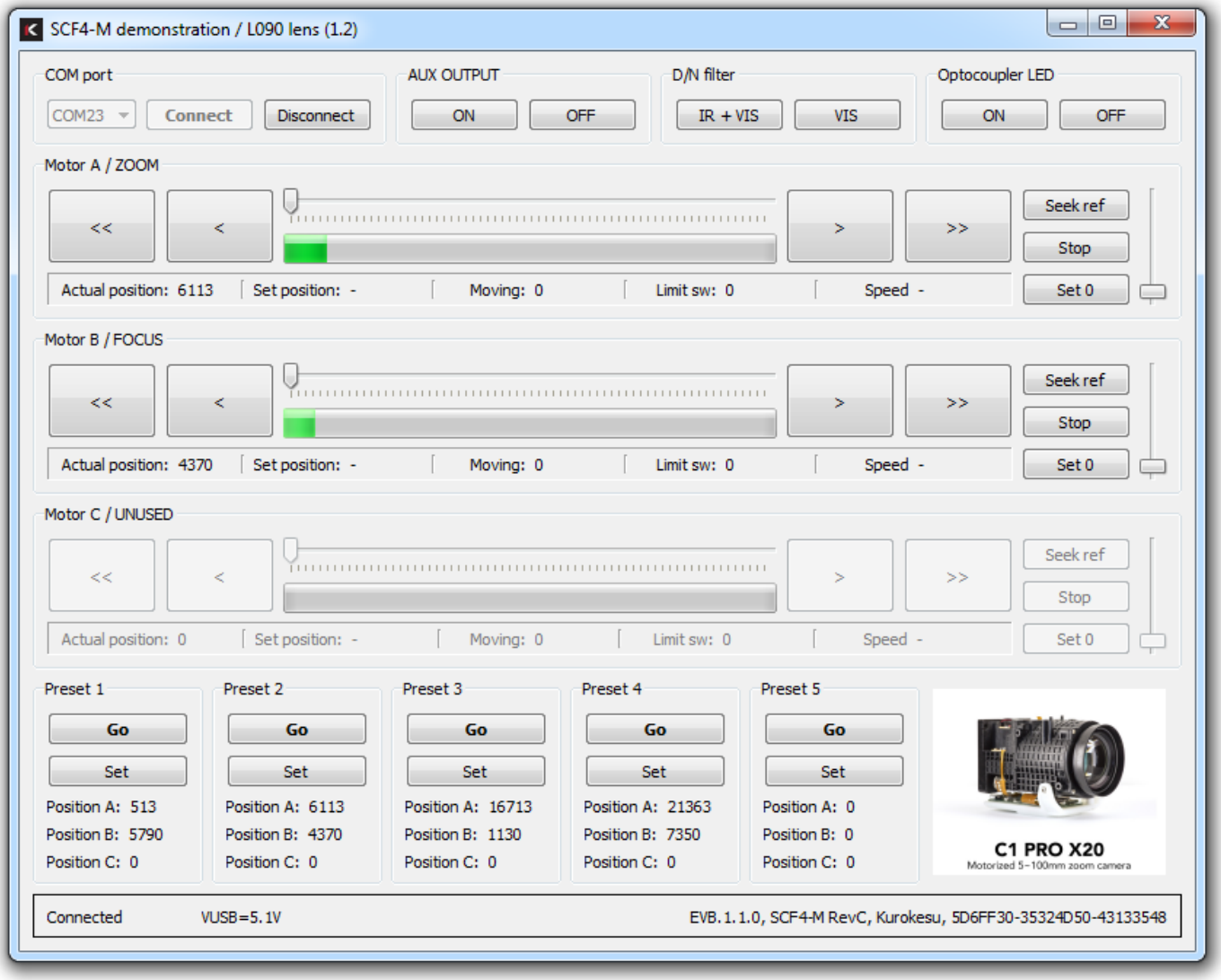
Camera / Product	Controller board	Notes
C1_PRO_X18	SCF4-L087	Camera block with 18x motorized zoom lens
C1_PRO_X20	SCF4-L090	Camera block with 20x motorized zoom lens
C1_PRO_X10	SCF4-L054	Camera with 10x motorized zoom lens
C1_PRO_X3	SCF4-L050	Camera with 3x motorized zoom lens
SCF4-L065-KIT	SCF4-EVB	5-50mm lens with M14/CS mount
SCF4-L002-KIT	SCF4-EVB	2.8-12mm lens with M14/CS mount

GUI

Python program with a graphical user interface allows many parameters to be controlled by an inexperienced user. To keep user interface uncluttered some parameters are not displayed. After connecting to SCF4 controller virtual serial port version string should be displayed at the bottom. Also, all controls will be activated. From now relative movement commands should be issued to check if connected motors turn properly (once button is pressed it will change positioning mode to absolute). When sliders operated positioning is switched to absolute mode, it makes sense after the axis is referenced.

Python scripting language with PyQt5 user interface was chosen because of rapid development

and clear syntax. Python 3.6 and a few dependencies should be installed before running any examples. Python and PyQt5 enables modern cross-platform user interface capable of running on high DPI monitors correctly.



Many lenses and motion systems can be connected to the controller and they can be split into two groups - with and without limit/reference switch/sensor. So there are two fundamentally different homing approaches.

1. Homing without limit switch is not very accurate but universal and works with all lenses. Idea is to rotate motor until the mechanical system hits its limit, then rotate a bit backwards and mark position as `0`. The lens should not sustain any irreversible damage, but this procedure should not be performed often.
2. The lens can incorporate various limit or reference switch configurations, therefore homing procedure should be adapted to a particular geometry. For example, the lens has reference opto-interrupter when it crosses the middle of its travel. By reading status register it becomes is clear at which section lens axis is and motor should be moved to center until switch changes status. To complicate things even more optocoupler or any switch has backlash/hysteresis and controller comparator has adjustable lower and upper thresholds. Good understanding of how particular lens behaves is a must and still, it can

take a few experiments to set optimal parameters. Some lenses depending on each axis position can have variable travel limits.

Current program version has following homing procedure implemented:

- Move motor by a fixed number of steps (lens may hit mechanical stop)
- Move opposite direction until the switch is actuated
- Move back by a fixed number of steps
- Set current position as 0

Program configuration is saved to `settings.json`. It's a read/write file with the purpose to provide default settings for some parameters like motion speed, jog steps, last used COM port, etc. and save settings and last position when program is closed.

Internal stepper motor driver has 16-bit position counter, absolute positioning is possible within a range of 0..65535 steps, (for 200 steps per revolution motor equals 20 full turns). In relative movement, if motion exceeds 16bit counter range it will overflow and continue the motion. The single move command is also 16 bits.

Command line

Command-line programs provide quick clean coding templates, examples to understand G-code usage. Programs explain how to:

- Read version string
- Initialize and perform relative movements
- Perform lens homing
- Change motion speed
- Perform an emergency stop
- And more

For the full list see the examples directory

Terminal

Control commands also can be sent directly from a terminal program of your choice. Baudrate for virtual COM port is irrelevant and communication speed over Full-speed USB 2.0 is 12 Mbit/s.

- Each command must be terminated by a newline
- Each command returns status code or requested information

Revision #13
Created 27 July 2020 17:56:29 by Saulius

Updated 6 February 2021 19:39:55 by Saulius