Servo Requirements

Mark Broihier
June 19, 2021

Table of Contents

1 Introduction

The purpose of this document is to define the requirements for a servo daemon that runs on Raspberry PIs. There are many of these available and this one is not intended to necessarily be better. I'm writing this mainly to increase my knowledge and experience with Raspberry Pi architecture.

1.1 Scope

This document defines the interface and processing for an application that implements a multi channel servo control daemon with low CPU requirements and high resolution Pulse Width Modulation (PWM).

1.2 Features

"servod" will control multiple servos over a range from 1 msec to 2 msec with 1 usec resolution and a 20 msec period. These are pretty consistent control interface ranges for modern servos.

"servod" will read a FIFO pipe and use the input from the pipe to determine the servo to modify and to determine the time, in microseconds, of the servo PWM pulse.

"servod" will output the PWM signal to a GPIO pin associated with the servo.

1.3 Acronyms

DMA	Direct Memory Access
FIFO	First In First Out
GPIO	General Purpose Input Output
PWM	Pulse Width Modulation

2 General Overview

"servod" will be a Raspberry Pi service that, on boot, starts a daemon that begins sending servo PWM signals to configured servos. Clients can control the servos by writing to a FIFO pipe. Valid pulse width values are used to adjust current pulse width to the value desired by the client.

3 servod

The following paragraphs outline the operation of servod and define the requirements of it's behavior.

3.1 Theory of Operation

A Raspberry Pi running Raspbian will host this application.

When powered on, this application will start and begin sending PWM pulses to configured GPIO pins. It will then pend on a FIFO pipe waiting for new servo pulse widths to modify. When input is present in the FIFO, the servo selection is verified and the pulse width is verified and then operations are performed to modify the width of the target servo. Once complete, the daemon will again pend on the FIFO.

When requested to power down, the daemon will send zero to the configure GPIO pins and tear down any system allocated resources.

3.2 External Interface Requirements

ID::SERVO-EXT-IF-010 Servo control shall be through a FIFO pipe.

ID::SERVO-EXT-IF-020 Servo output shall be 3.3 volt PWM signals to configurable GPIO pins.

3.3 Processing Requirements

The following paragraphs identify the processing to be performed by servod.

3.3.1 On Boot

ID::SERVO-PR-BOOT-010 On boot, the servod shall read its configuration and set the servos to their initial state.

ID::SERVO-PR-BOOT-020 The initial state of servod shall continuously send initial pulse widths to all the configured servos.

ID::SERVO-PR-BOOT-030 servod shall initialize a FIFO pipe at location /dev/servo_fifo.

3.3.2 On FIFO Input

ID::SERVO-PR-FIFOI-010 servod shall accept a line of text input from the FIFO.

> ID::SERVO-PR-FIFOI-010.001 servod shall confirm the input conforms to the following format: %d, %d\n.
>
> ID::SERVO-PR-FIFOI-010.002 servod shall confirm that the first integer digit string identifies a zero based servo number from 0 to the number of possible servos minus one.
>
> ID::SERVO-PR-FIFOI-010.003 servod shall confirm that the second integer digit string identifies a positive pulse width in microseconds from 1000 to 2000.
>
> ID::SERVO-PR-FIFOI-010.004 Invalid FIFO input (input that does not conform to the previous requirements), shall be discarded.

ID::SERVO-PR-FIFOI-020 servod shall process valid text input from the FIFO.

ID::SERVO-PR-FIFOI-020.001 servod shall use the first integer as an identifier of the servo to modify.

ID::SERVO-PR-FIFOI-020.002 servod shall use the second integer to set the servo selected to the pulse width that matches the integer.

3.4 Provisioning Requirements

ID::SERVO-PROV-010 Configuration of servod shall be done with the use of a text file, servoDefinitionFile.txt read when started.

ID::SERVO-PROV-010.001 Each line of the text file defines on servo and servos will be defined sequentially starting at zero.

ID::SERVO-PROV-010.002 servod shall confirm the input conforms to the following format: %d, %d\n.

ID::SERVO-PROV-010.003 servod shall confirm that the first integer digit string identifies a unique GPIO pin in the BCM format.

ID::SERVO-PROV-010.004 servod shall confirm that the second integer digit string identifies a DMA channel between 0 and 15.

ID::SERVO-PROV-010.005 Invalid file input shall cause servod termination.

3.5 Performance Requirements

ID::SERVO-PERF-010 servod shall control at least 8 servos.

ID::SERVO-PERF-020 servod shall be able to set pulse widths with at least 1 usec resolution.

ID::SERVO-PERF-030 servod shall require less that 1% CPU utilization after initialization.