A SCOMP Peripheral for Hobby Servo Motor Positioning and Oscillation

Suhel Keswani

Submitted
April 23, 2024

Introduction

This document discusses the functionality and design of the SCOMP Peripheral for Hobby Servos. This peripheral controls servos with appropriate pulse width modulation and provides a layer of abstraction for programmers interested in positioning and autonomous oscillation. An intuitive API, degree granularity of positioning, and independent servo control were prioritized due to customer requirements. By using a Gantt chart and assigning individual tasks to team members across weeks, all proposed features were implemented successfully.

Device Functionality

Capabilities

Each servo can be controlled using two modes -- one in which the position of a servo is set, and one in which the position of the servo oscillates from 0 to 180 degrees. Across both modes, pulses are continuously generated to set and hold servo positions appropriately.

The end-user is able to control each of the servos across modes independently and in tandem with one another. Additionally, hobby servos are always safely controlled, as hardware level mechanisms prevent the generation of pulses under 6 ms or over 24 ms.

Application Platform Interface

To send instructions to the peripheral, programmers simply use the OUT instruction to write signed integer values to the peripheral. Position register types interpret values as degrees from 0 to 180; note that values under 0 are interpreted as 0, and values over 180 are interpreted as 180. Oscillation register types interpret values as an amount of oscillations to perform, where an oscillation represents one iteration of moving to 0 degrees, holding this position for approximately 1 second, moving to 180 degrees, and then holding this position for at least 1 second. Oscillation values under 1 are interpreted as 1. Note that new instructions to the peripheral will always override previous/current instructions as needed (e.g. an incoming position instruction for servo 4 will overwrite any existing oscillation of servo 4).

TABLE I

Output Register Map of SCOMP Peripheral For Hobby Servo Motor Positioning And Oscillation

I/O Addr.	Servo Pulse	Register Type	B 15	B 14	B 13	B 12	B 11	B 10	B 09	B 08	B 07	B 06	B 05	B 04	B 03	B 02	B 01	B 00
0x50	0	position	position in degrees															
0x51	0	oscillation	number of full range oscillations															
0x52	1	position	position in degrees															
0x53	1	oscillation	number of full range oscillations															
0x54	2	position	position in degrees															
0x55	2	oscillation	number of full range oscillations															
0x56	3	position	position in degrees															
0x57	3	oscillation	number of full range oscillations															

Design Decisions and Implementation

In the design of this peripheral, programmer utility and usability were prioritized. Degree granularity, independent control, and automation were deemed to be the most necessary and versatile features. In order to achieve precise pulse width modulation for degree granularity, a faster, 100kHZ clock is used. By assigning a pair of I/O addresses for each pulse output, simultaneous and independent control across modes is allowed across the four servos. To support this, the peripheral requires eight unique chip select inputs from an appropriately fitted I/O decoder to control four distinct pulse generators capable of producing pulses for both position and oscillation modes.

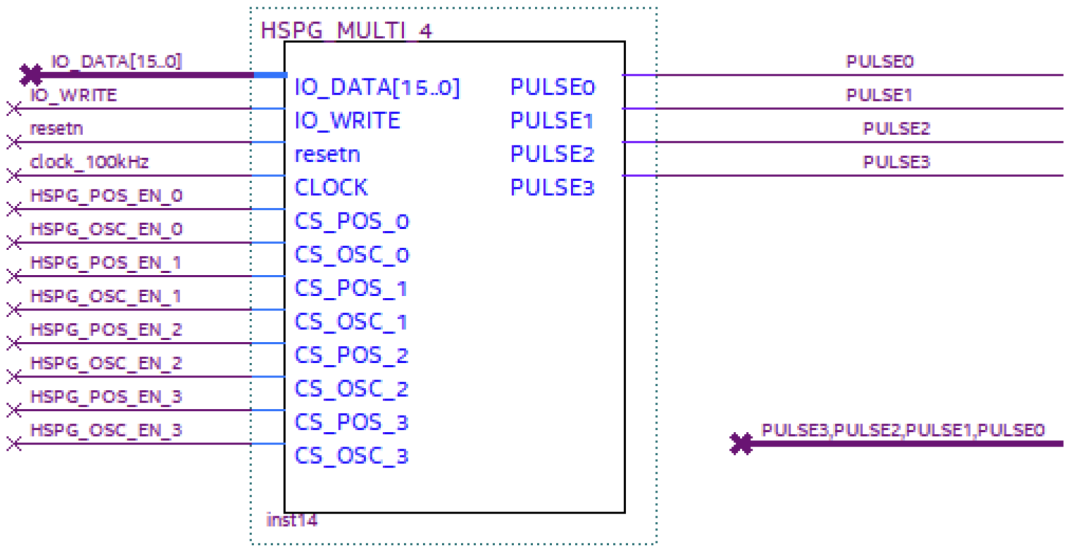

Figure 1. Block diagram of the SCOMP Peripheral for Hobby Servos. The 100kHz clock and chip select inputs are shown.

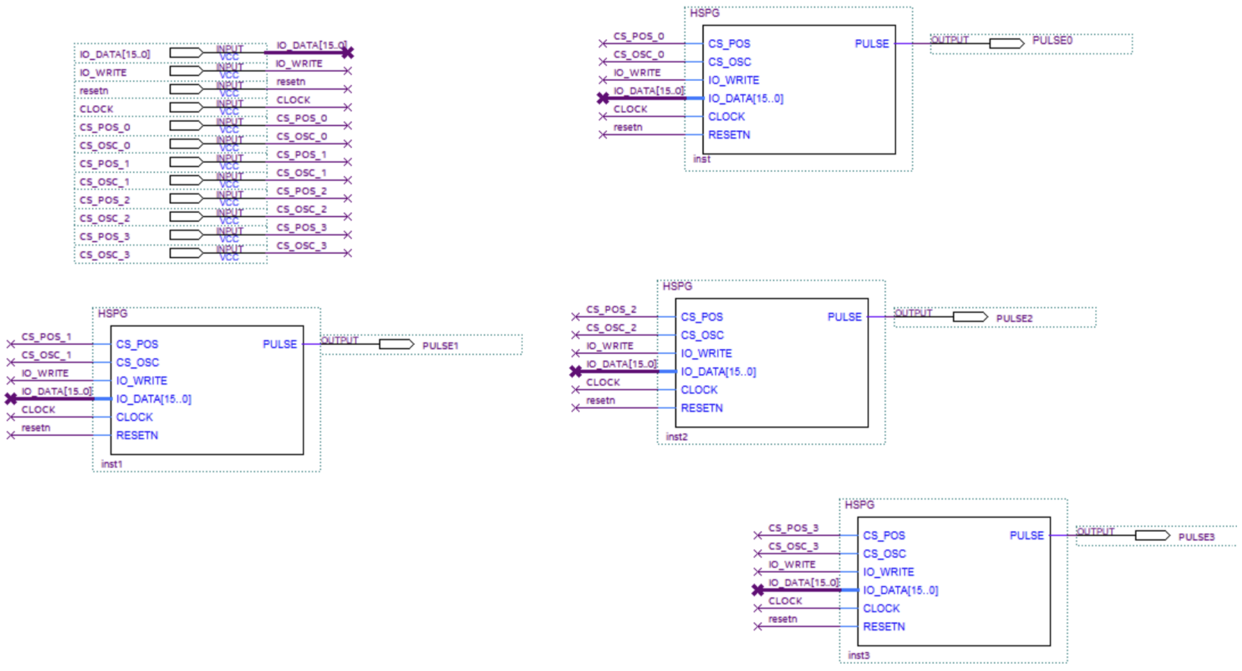

Figure 2. Block diagram showing the four internal pulse generators within the SCOMP Peripheral for Hobby Servos.

Conclusions

By providing programmers with precise positioning and automatic oscillation in an effective API, the SCOMP Peripheral for Hobby Servos creates easy and effective control of four servos. While automatic oscillation functionality is useful, as it simplifies programming while freeing up the processor to perform other tasks, customization can provide more value. Additional desirable functionality for autonomous oscillation might include an option for infinite oscillation, configurable oscillation positions, and/or configurable delays between movements. In order to maintain a usable API, such parameters can be written to the peripheral separately across I/O addresses. Additionally, improvements to the existing design that better facilitate future development could have been made. Namely, the use of a mode input vector signal alongside four chip selects can allow for individual servo mode control without requiring a separate chip select input for each servo and mode.