

Title:	Second Generation Photogrammetry Project
Author:	J.A. Crawford
Abstract:	Although the friction drive approach reported on in Part 1 worked quite well, there were still some aspects of the design I did not care for. The first part of this installment discusses a re-design of the friction drive with which I am genuinely pleased with. This design is much easier to align and is probably about one-half the size of the original design. In addition, this installment looks at the software/firmware required to operate the AS5048A angular encoder. With the 3-phase motor drive software already in hand, the entire mount was exercised to evaluate angular slew performance, and most notably, the angular re-trace precision capabilities of the right ascension axis.
Date:	Dec 23, 2025
Version:	1.0

1 Friction Drive Re-Design

The main issues I had with the original design included:

1. physical size of the drive mechanics was too large in my opinion
2. the arm assembly used for the main drive axles used very small bearings which on occasion were broken when too much squeezing force was exerted on the main friction disk
3. the 6mm axles incurred a fair amount of bending due to the forces involved which invited second-order impairments to come into play, including questions about how much contact area between the main friction disk and axles was actually being achieved
4. other stresses were present in the drive which led to *creeking sounds* on occasion as the mechanics abruptly shifted (less than 0.001 inches most likely) to accommodate the stresses

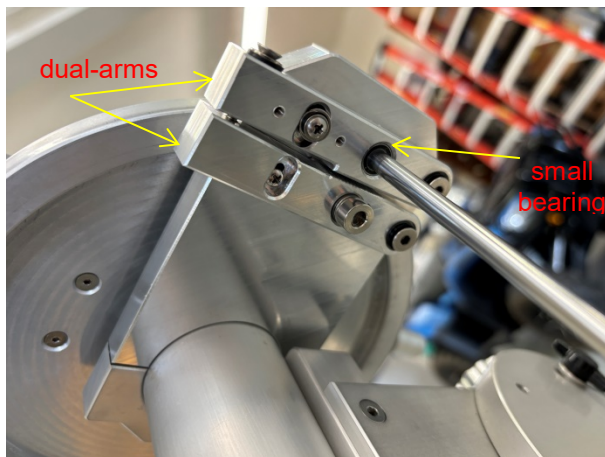


Figure 1 Original friction drive arrangement using a dual-arm arrangement



Figure 2 Original design which used a *spanner* below the main bearings to compensate for axle-bending and slight shifting of the (small) main bearings

The redesign is different in that:

1. the 6mm axles involved with the main friction disk were traded out for 8mm axles thereby greatly reducing axle bending and 2nd order stresses. At the same time, considerably larger forces could be accommodated without breaking or warping the main bearings.
2. the dual-arm arrangement used to hold the main axle bearings was abandoned for a pair of larger bearings situated along a true radial line. The main bearings moved *together* due to the mounting method shown in Figure 3.

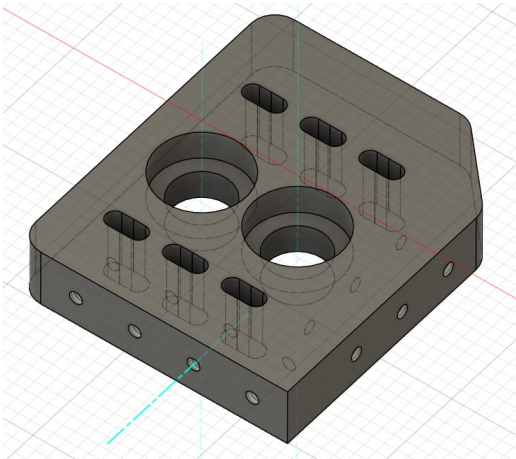


Figure 3 The redesign uses a bearing mount which keeps the bearings together as a pair, but still allowing the pair to adjust radially by using the adjustment slots (and bolts)

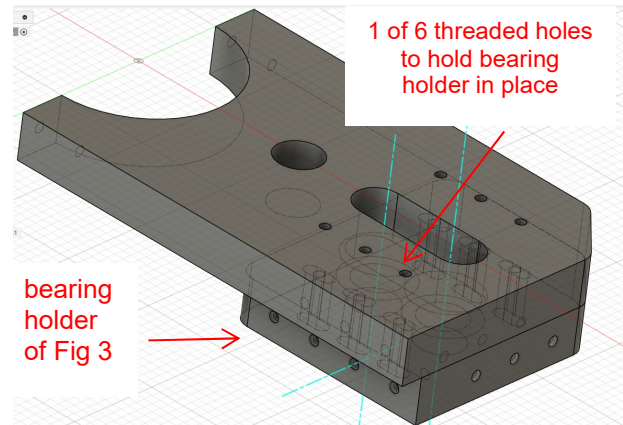


Figure 4 Bearing holder of Figure 3 held radially in place with the top baseplate affixed to the associated telescope axis. Holder is held in place using 6 thru-bolts from the under-side

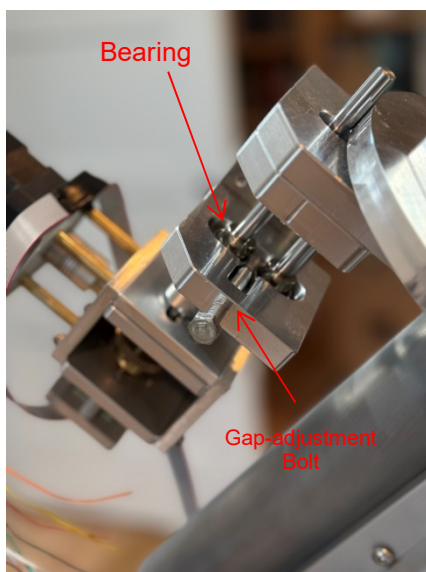


Figure 5 One of the most successful aspects of the new design is replacement of the so-called spanner with a second bearing arrangement in which the gap between bearings is adjusted using a simple tapered bolt. Adjusting the bolt changes the force exerted by the axles on the friction disk while keeping the bearings along a radial line.

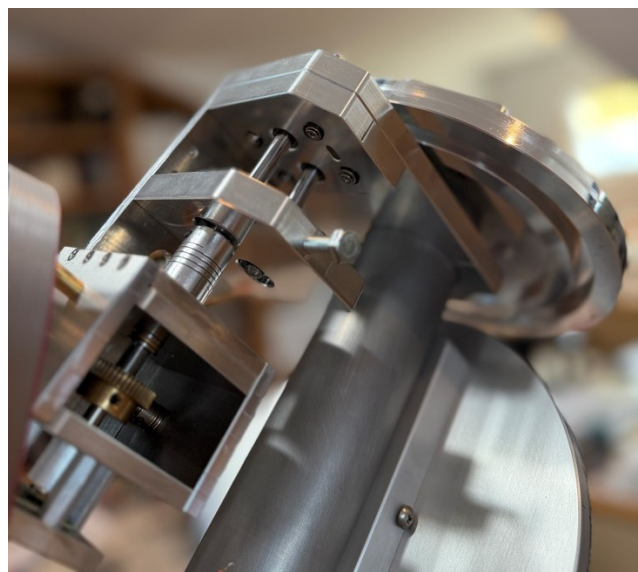


Figure 6 Upward looking view of the Dec axis arrangement

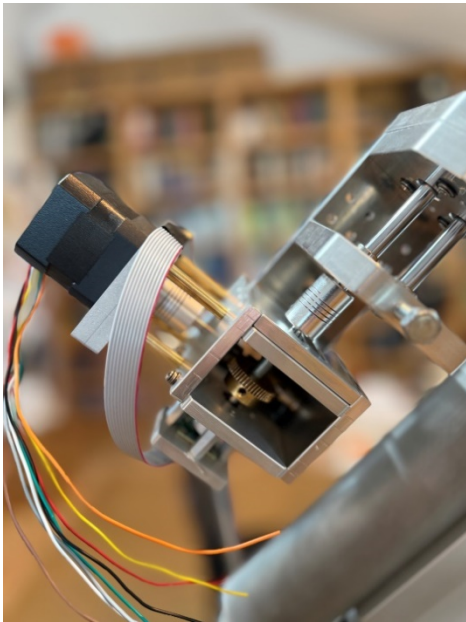


Figure 7 Side-view of the Dec axis clearly showing the worm gear box and axial couplers

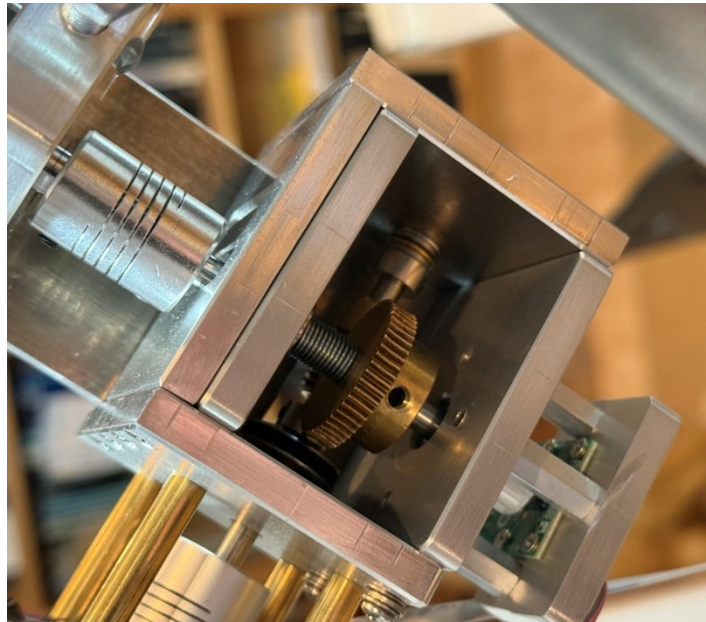


Figure 8 Close-up of Figure 7. Angular encoder shown mounted below the worm gear assembly.

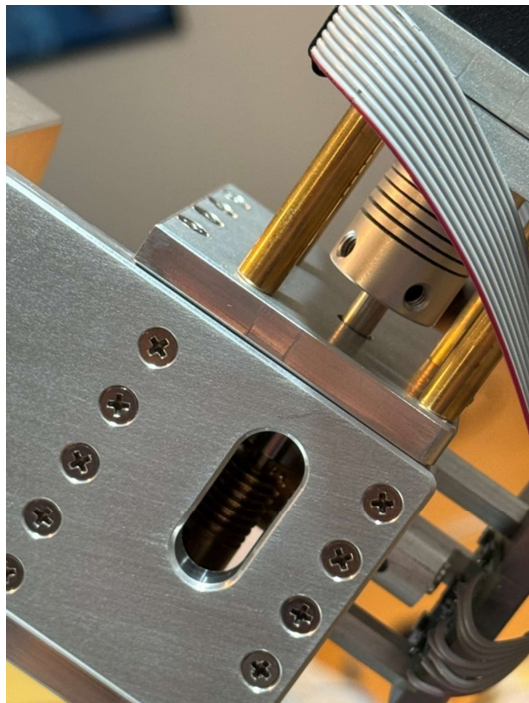


Figure 9 Opposite side of the Dec axis arrangement. Milled window aides worm gear alignment.



Figure 10 View of complete Dec axis drive

This new mechanical design of Figure 10 will be replicated for the RA axis.

2 Overview

Early on, I spent a fair amount of time looking into re-hosting the project on a Raspberry Pi 5 (RP5) while retaining the Texas Instruments TMS320F28379D-based Launchpad for real-time motor control as shown in Figure 12. However, I ultimately found the RP5 software maturity level was insufficient for my needs, and I have consequently decided to push this aspect of the project off to a future date. The high-level configuration for my efforts therefore remains that shown in Figure 11.

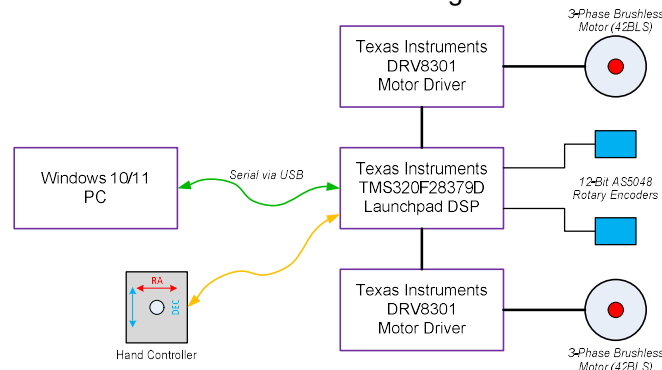


Figure 11 Development platform (short-term)¹

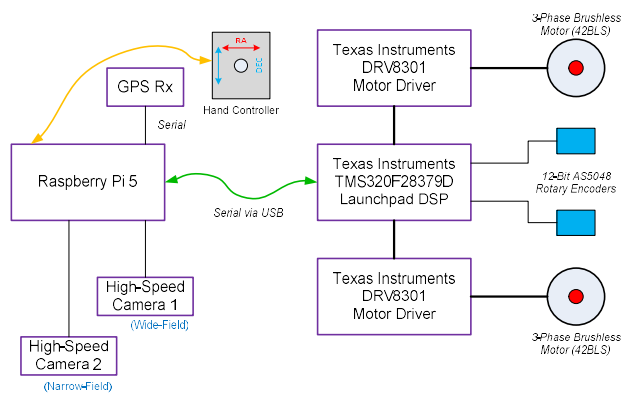


Figure 12 Long-term project evolution replaces the PC platform with a Raspberry Pi 5, thereby providing support for (i) GPS receiver, (ii) wide-field CMOS camera, and (iii) narrow-field CMOS camera

2.1 TMS320F28379D Usage

The TI TMS320F28379D consists of two nearly identical CPUs, each with its own control loop accelerator (CLA). CPU01 is the master CPU of the two and takes care of:

- PC – DSP interface
- Ping-pong reading of the two angular encoders (one for the RA axis, the second for the Dec axis)
- Hand-controller interface
- Motor control for the RA axis

CPU02 takes care of:

- Motor control for the Dec axis

¹ From U29152.

Although the two CPUs (and their associated CLA) are virtually identical, joint memory allocation between the two CPUs must be done carefully.

A picture of the Header Board which connects to the backside of the TI Launchpad DSP board² (TMS320F28379D) is shown in Figure 13. Ribbon cable interfaces to the RA and Dec angular encoders are provided, along with 8 GPIOs which can be made available for future uses as needed.

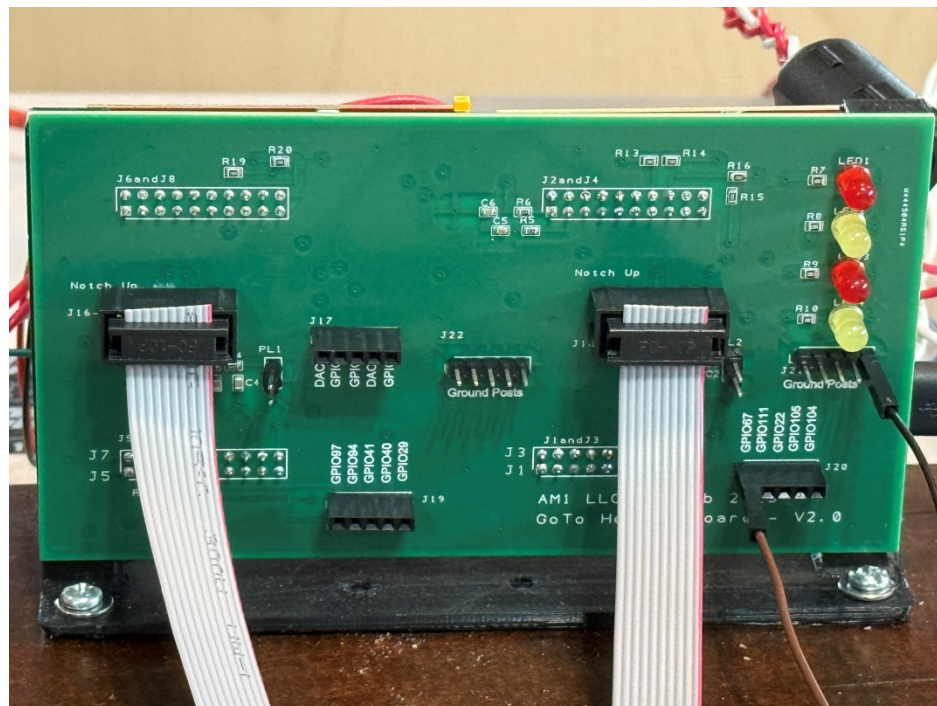


Figure 13 DSP backside Header Board which hosts ribbon cable interfaces to the RA and Dec angular encoders. Board space permitted the inclusion of 4 LEDs which are tied to GPIO pins, as well as 8 additional GPIOs which can be programmatically made inputs or outputs. Each LED draws 10 mA when lit.

² PCB made by Pcbway. I hand assembled the PCB in my shop.

3 Graphical User Interface

Graduation to the 2nd-generation mount design necessitated a substantial re-write of the control interface to the TI DSP since the rotary encoder was changed. The 1st-generation software had a large amount of code devoted to calibrating the rotary encoder and assessing its performance, but none of this code is required for the 2nd-generation mount. The top-level GUI as of this writing is shown in Figure 14.

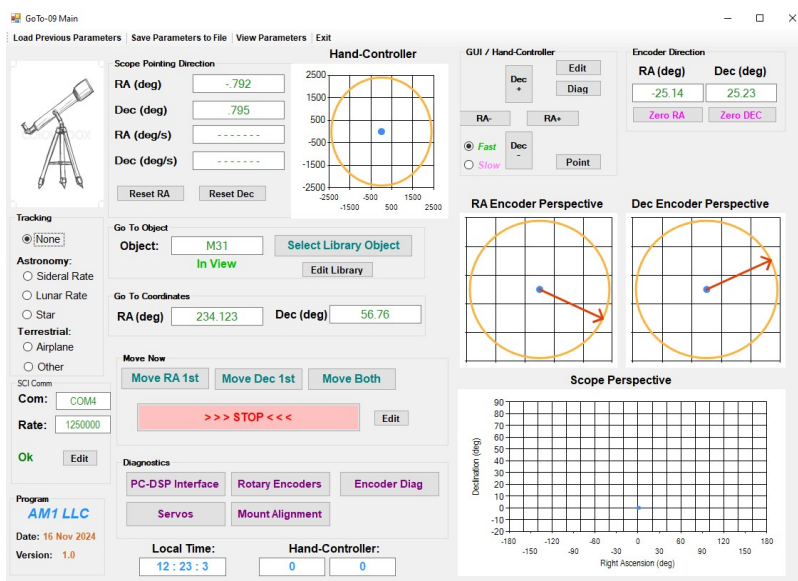


Figure 14 Top-level GUI (Right Ascension/Declination Mode)

The 1st-generation mount had to anticipate cogging and other non-ideal torque behaviors especially with its RA axis. This necessitated a closed-loop control system involving a high-speed optical angular encoder working with a 3-phase servo motor control system. The much smoother operation of the 2nd-generation mount combined with the zero-backlash of its friction drive design dramatically simplifies the servo motor control, making it essentially an *open-loop* system. Although the 2nd-generation mount also includes rotational encoders, these will primarily be used for calibration purposes rather than closed-loop servo control.

4 Mount Rotational Motion

Smooth angular motion for the mount was initially discussed quite some time ago in §8 of [2], but quite a few design details have changed since that material was written. Most notably, both mount axes (i) use friction drive, (ii) a GUI-based manual motion control has been added, (iii) an analog hand-controller has also been added, and (iv) provision for a variety of tracking options has been included.

The *gear ratio* for each axis is a combination of (i) a friction drive portion having an advertised ratio of 7.5 in / 8 mm = 23.81254763 and (ii) a worm gear portion having a ratio of 30:1. Combined, the overall gear ratio is about $30 \times 31.7500635 = 714.37643 : 1.0$. An axial rotation of 1 arc-second translates into a nominal servo motor rotation of 714.376 arc-seconds which is equivalent to 11.9063 arc-minutes or 0.198 arc-degrees for the servo motor. This is a severe accuracy requirement because any required torque will

no doubt erode the achieved accuracy as well³. This would be equivalent to about 1814 *ticks* per revolution if a stepper motor were to be used⁴.

The worm gear ratio is *exact* whereas the friction drive portion will have a very small error which will need to be quantified. The 8mm axle is specified to fall within 7.995mm to 7.97mm resulting in a total gear ratio value ranging from 714.8232 to 717.0654. Although the axle-radius can affect the scaling factor appreciably, careful measurements will be required in order to determine the *roundness* factor of the axle, and this will directly translate into *periodic error*.

Once the mount is properly aligned with the celestial pole, gear ratio calibration should be straight forward. A star well above the horizon can be tracked for a given amount of time, the associated Right Ascension change easily computed based upon the length of a sidereal day, and that quantity compared with the angular reading of the angular encoder involved. Since the angular encoder's absolute accuracy is not perfect, angular position measurements at multiple time values should be captured and a numerical curve fit applied for calibration purposes⁵.

If star tracking is invoked for the astronomy use-case, most of these issues are substantially mitigated. In the photogrammetry-triangulation use-case, however, pointing angles will be measured open-loop based upon the voltage waveform phases presented to the servo motors and these imprecision issues will be very important to assess.

4.1 DSP-Based Processing

The DSP-based signal processing necessarily comes first because all of the real-time processing is done by the DSP. The PC C# code will preprocess DSP commands as needed, but none of this activity will have a real-time nature.

A high-level conceptual sketch for the control of one mount axes is shown in Figure 15, with considerably more detailed material to follow.

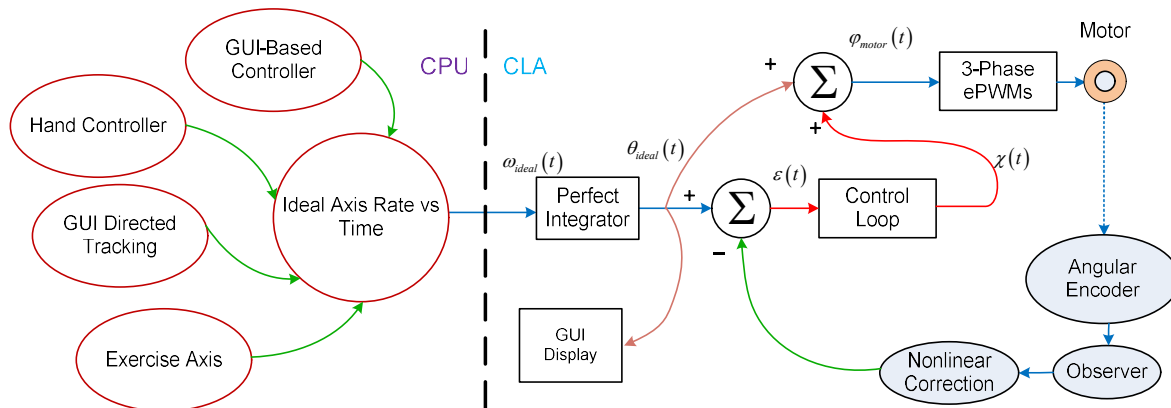


Figure 15 High-level sketch⁶ for the control of one mount axis. Observer function is described in §0. The major 3-phase ePWM algorithms are described in §9.

³ Measurements need to be made of the completed mount to determine how pointing accuracy is degraded by weight imbalances which may be present in the mount.

⁴ This is unrealistic for most stepper motors since their native resolution is normally 200 steps per revolution which would have to be increased to roughly $200 \times 16 = 3200$ steps using interpolation. Motor torque drops off significantly as the interpolation factor is increased.

⁵ See Figure 15 of [1].

⁶ U29152_Figures.vsd.

4.1.1 DSP CPU

The DSP's CPU handles communications between the PC and DSP using one SCI port. Most of the real-time signal processing is done in the control loop accelerator (CLA) associated with each respective CPU (the TMS320F28379D has two CPUs, each CPU having its own CLA.) These communications primarily entail parameter changes to be invoked in the signal processing algorithms. Some diagnostic information from each CLA is relayed back to each CPU and subsequently back to the PC using the same SCI port.

CPU-related processing is discussed later in §4.2.

4.1.2 Hand Controller Processing

A hand-controller has been integrated into the overall design as shown in Figure 11. This will make it possible to manually direct the pointing direction of the telescope with a minimum degree of complexity.

The TI TMS320F28379D processor includes a 16-input multiplexed SAR ADC which can be configured for 12-bit resolution (single-ended) or 16-bit (differential). Combined with the hand-controller as sketched in Figure 16, a simple manual control for slewing the mount in RA and DEC directions can be implemented.

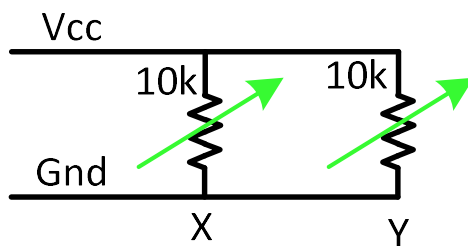


Figure 16 Hand-controller is based upon a pair of joy-sticks which use 10 k Ω variable resistors as shown. Vcc is connected to a convenient DC voltage (nominal 3.3V) from the TMS320.

A joy-stick style hand-controller is attached to the DSP electronics in order to facilitate manual control over the telescope's pointing direction. The 2-axis joy-stick is implemented as shown in Figure 16 with the potentiometer center-tap voltages provided back to the DSP's ADC-D input. DSP ADC-A, ADC-B, and ADC-C are dedicated to ePWM use by way of the two DRV8301 3-phase motor controllers and are consequently not available. The ADC-D input is toggled back and forth between the two joy-stick voltages in a ping-pong fashion.

The output voltage range provided from the joy-stick outputs is limited to between 0V and 3.3V. ADC-D operates as a single-ended 12-bit ADC and its output values range from 0 to 4095. This is the input to the hand-controller algorithm shown in Figure 17.

It is highly desirable for the hand-controller input to exhibit a slower attack time than the decay time. Imagine using the hand-controller to slew the telescope's pointing direction. For a number of reasons, the starting motion should not be too great even if the hand-controller is slammed to one extreme of its physical range. Similarly, when trying to steer the pointing direction it would be annoying to grossly overshoot the desired direction. Using a shorter decay time help avoid this issue while at the same time not abruptly halting the motion which could cause other problems.

A block diagram for the overall controller (for one axis) is shown in Figure 17. This design will have to be modified once camera-based tracking is included in the project, but there is no point in over complicating the project at this point in time with that detail.

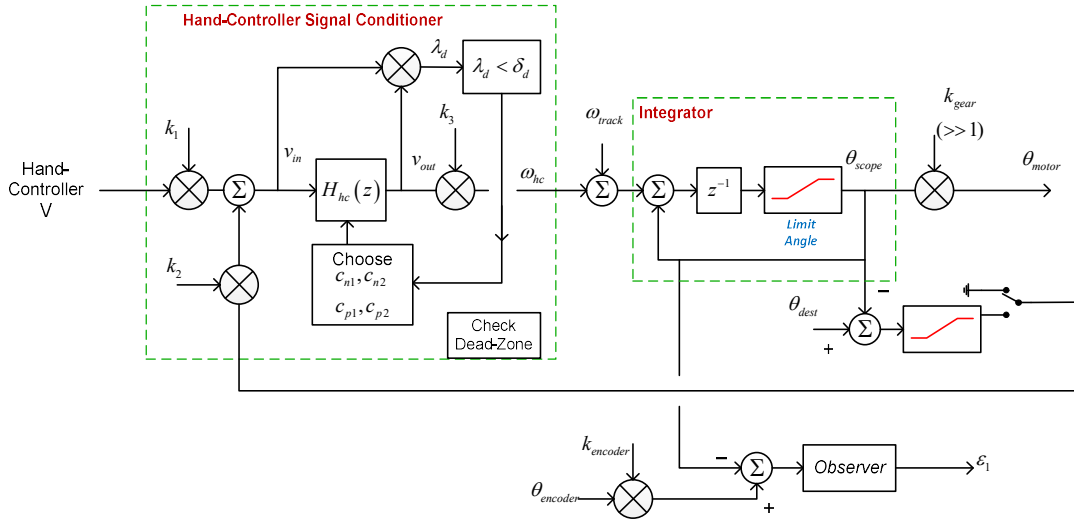


Figure 17 Motion controller concept for one axes⁷

The hand-controller signal conditioning block is based upon a variant of a simple one-pole low-pass filter which has as its Laplace transform transfer function

$$H(s) = \frac{1}{1 + s\tau} \quad (1)$$

The bilinear transform $s \approx (2/T)(1 - z^{-1})(1 + z^{-1})^{-1}$ can be used to convert (1) into a discretized digital form as

$$v_{out} = \frac{(1 + z^{-1})v_{in} - z^{-1}\left(1 - \frac{2\tau}{T}\right)v_{out}}{\left(1 + \frac{2\tau}{T}\right)} = \frac{(1 + z^{-1})v_{in} - z^{-1}c_n v_{out}}{c_p} \quad (2)$$

In order to implement different attack and decay times as advocated in Figure 17, the c_n and c_p coefficients in (2) can be adjusted in real-time, each coefficient set having a different time constant value. More specifically,

$$\begin{aligned} c_{pfast} &= 1 + \frac{2\tau_{fast}}{T} \\ c_{nfast} &= 1 - \frac{2\tau_{fast}}{T} \\ c_{pslow} &= 1 + \frac{2\tau_{slow}}{T} \\ c_{nslow} &= 1 - \frac{2\tau_{slow}}{T} \end{aligned} \quad (3)$$

Parameter δ_d is a fairly arbitrary small positive constant, and T is the time between digital samples (updates). The *Check Dead-Zone* box is included to prevent an imperfectly-centered hand-controller to cause the pointing direction to drift. It simply outputs zero unless the absolute value of the hand-controller's output is larger than a small constant.

⁷ U29152_Figures.vsd.

The raw hand-controller outputs are integers ranging from 0 to 4095 as already stated. The hand-controller inputs to Figure 17 are immediately shifted to lie in the range of -2048 to 2047. Gain parameter $k_1 = 1$. The discrete-time transfer function $H_{hc}(z)$ is implemented using IEEE-754 single-precision floating-point numbers throughout so no internal scaling is required per say.

Since $H_{hc}(z)$ is a simple first-order lowpass filter, there should be no over-shoot or under-shoot at the v_{out} output. Axis rotational rates ω_{hc} and ω_{track} are referenced to the telescope pointing direction. Although the maximum slew rate for the axes is important, a more important limitation for this design is the maximum motor RPM value used since the gear-ratio is so large (1587.503175 : 1.0). Therefore, the maximum motor RPM is used as the independent quantity and the maximum axis rotational rate follows as

$$\omega_{scope_max} = \max(|\omega_{hc} + \omega_{track}|) = \frac{RPM_{max}}{60} \frac{2\pi}{k_{gear}} \quad (4)$$

Figure 18 through Figure 21 show the results of a MATLAB time-domain simulation where Figure 18 is the assumed hand-controller input for one axis. The MATLAB script is provided in §11.

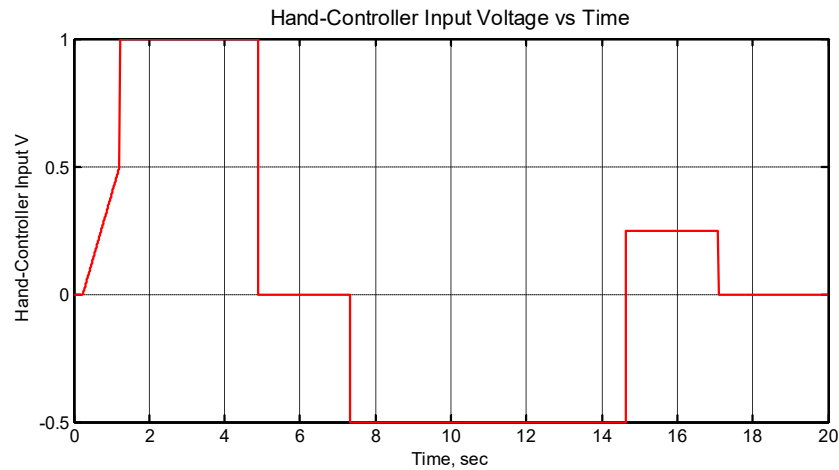


Figure 18 Example hand-controller input⁸ to Figure 17

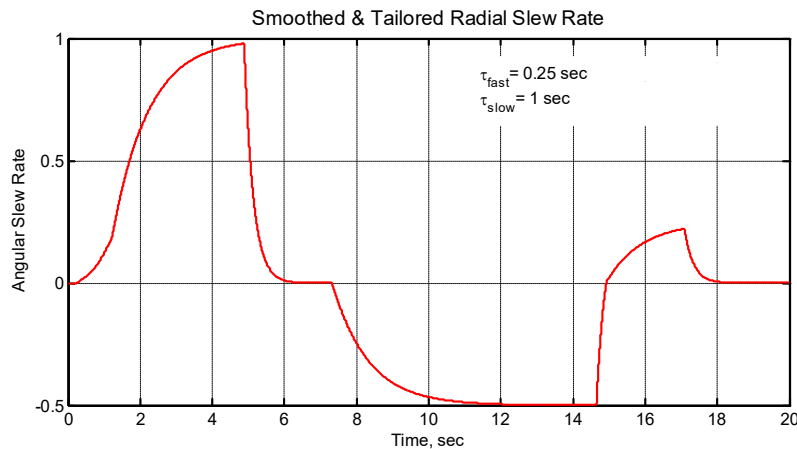


Figure 19 Output from hand-controller signal conditioner in Figure 17 exhibiting different attack and decay time behavior

⁸ Computed using u29443_handcontroller.m.

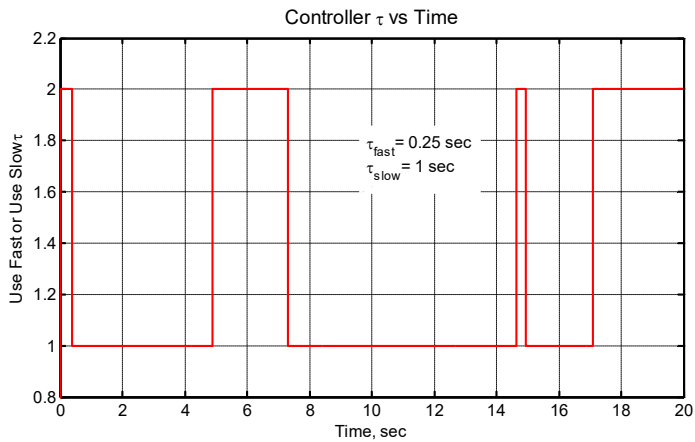


Figure 20 Time constant choice versus time associated with Figure 18 and Figure 19

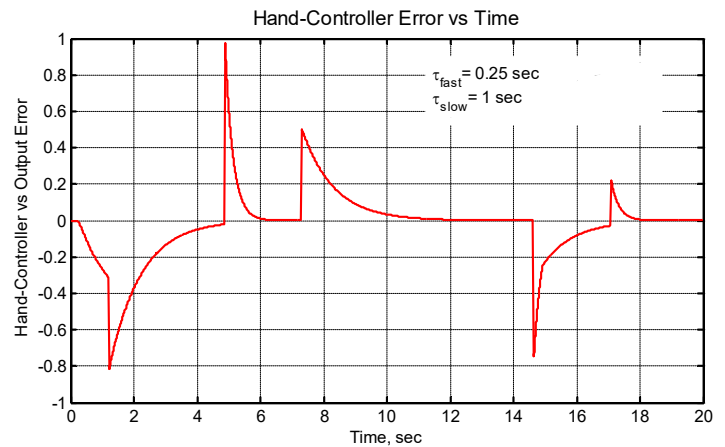


Figure 21 Controller output error versus time associated with Figure 18, Figure 19, and Figure 20

One Step Backwards

Once the hand-controller algorithm was implemented on the DSP, it was very obvious that the ADC-D inputs were being seriously contaminated by EMI from the ePWMs. Either the order of the lowpass filter given by (1) needed to be increased, or additional pre-filtering ahead of Figure 17 was needed. Increasing the order of (1) would have likely introduced ringing in the transient response, so the pre-filtering option was chosen.

The pre-filtering was implemented as a simple *boxcar filter* which is synonymous with using a *N-tap* FIR filter having tap weights of $1/N$. This filter can be efficiently implemented using a pointer-based circular buffer.

It also proved very helpful to (temporarily) allocated one GPIO pin (J1-8, GPIO-22) to CLA1 so that CLA algorithm execution time could be directly measured using an oscilloscope.

Only a few GPIO lines can be reached by the CLA(s), specifically GPIO-16 through GPIO-23. Most of these are allocated to ePWM usage. Fortunately, many different types of interrupts can be used to trigger different CLA tasks as needed without explicitly dedicating GPIO pins to the CLA(s).

4.2 CPU-Based Processing

The commands available from the top-level GUI (Figure 14) which are conveyed over the PC/DSP interface (using the SCI bus) to the DSP (CPU1) are summarized in Table 1. Commands intended for CPU2 (and its CLA) must pass through CPU1 first.

Table 1 GUI Commands in Figure 11

Command	Cmd to DSP	Description
	\$\$\$\$	
Test string	_X	X is followed by a random test string which is echoed back by DSP to the PC
Set tracking mode	_set_trk_mode	Parameter: mode na lun Lunar rate, only affects RA axis sid Sidereal, only affects RA axis air Airplane, TBD oth Other, TBD
Set local time	_set_time	Parameters: hrs, min, sec Within DSP, converted to total seconds since midnight (i.e., $3600 \times hrs + 60 \times min + sec$)
Get local time	_get_time	Parameters: hrs, min, sec Computed from internal total seconds since midnight
Reset DSP	_reset_dsp	Resets the DSP. No parameters
Reset RA	_reset_ra	Resets the internal program variable for Right Ascension to zero for the present pointing direction. Used to set home position. Parameter: None
Reset Dec	_reset_dec	Resets the internal program variable for Declination to zero for the present pointing direction. Used to set home position. Parameter: None
Go to coordinates	_goto	Mount is directed to the RA / Dec direction as manually entered into the GUI. Parameters: (in degrees) RA Dec Move type (1:RA first, 2:Dec first, 3: together)
Stop motion	_stop	Immediately halts motion of both axes
Return to home position	_go_home	Single parameter mode: 1:RA first, 2: Dec first, 3: together
>>>> Hand Controller <<<<		
Set HC sampling rate	_set_hc_fs	Sets up timer2 period for CPU1.
Read back Joystick values	_get_joy_vs	Read back joystick voltages, V

Command	Cmd to DSP	Description
Set dead-zone	<code>_set_xdeadzone</code>	Set joystick dead-zone parameter for RA axis
	<code>_set_ydeadzone</code>	Set joystick dead-zone parameter for Dec axis
Set time constants	<code>_set_hc_xcoeffs</code>	Sets RA axis time-constants <code>cp_fast</code> , <code>cp_slow</code> , <code>cn_fast</code> , <code>cn_slow</code> time constants
	<code>_set_hc_ycoeffs</code>	Sets Dec axis time-constants <code>cp_fast</code> , <code>cp_slow</code> , <code>cn_fast</code> , <code>cn_slow</code> time constants
>>>> Motor Control <<<<		
Set A0 limits for RA axis	<code>_set_RA_A0</code>	Sets A0min and A0max for RA axis motor control
Set A0 limits for Dec axis	<code>_set_Dec_A0</code>	Sets A0min and A0max for Dec axis motor control
Set RA ePWM parameters	<code>_set_xepwm_parms</code>	Set ePWM timer period (in ticks), motor-to-axis step-down ratio <code>kaxis</code> , maximum motor RPM
Set Dec ePWM parameters	<code>_set_yepwm_parms</code>	Set ePWM timer period (in ticks), motor-to-axis step-down ratio <code>kaxis</code> , maximum motor RPM
Set RA axis limits	<code>_set_RA_limits</code>	min and max limits, degrees
Set Dec axis limits	<code>_set_Dec_limits</code>	min and max limits, degrees
Enable/disable ePWMs	<code>_enable_epwms</code>	Enable or disable ePWMs
>>>> Angular Encoders <<<<		
Get RA encoder angle	<code>_get_ra_coder_ang</code>	
Get Dec encoder angle	<code>_get_dec_coder_ang</code>	
Get all encoder details	<code>_get_encoder_all</code>	Retrieves RA encoder angle, AGC, and Mag values Single input parameter, 1 for RA axis, 2 for Dec axis
Set EPWM sampling rate	<code>_set_fs</code>	Set sampling rate for EPWMs
Set EPWM MASH order	<code>_set_mash_order</code>	Set order for MASH algorithm
RA+ Slew Rate	<code>_slew_ra_pos</code>	Pertains to GUI buttons Parameters, RA+ maximum slew rate, degrees/sec
RA- Slew Rate	<code>_slew_ra_neg</code>	Pertains to GUI buttons Parameters, RA- maximum slew rate, degrees/sec
DEC+ Slew Rate	<code>_slew_dec_pos</code>	Pertains to GUI buttons Parameters, DEC+ maximum slew rate, degrees/sec
DEC- Slew Rates	<code>_slew_dec_neg</code>	Pertains to GUI buttons Parameters, DEC- maximum slew rate, degrees/sec

5 Angular Encoder

The magnet-based angular encoder, AS5048A⁹, is new to my project. As described previously in [1], this encoder is much smaller, and much easier to align than the previous optically-based encoder. Although it only provides 14-bits of resolution, combined with the 31.75:1 ratio from the friction drive portion of the drive chain, the theoretical resolution is $360^\circ / 31.75 / 2^{14} = 0.0006921^\circ$ or equivalently about 2.5 arc-seconds. This is more than adequate for my terrestrial usage plans, as well as initial pointing for my astronomical pursuits.

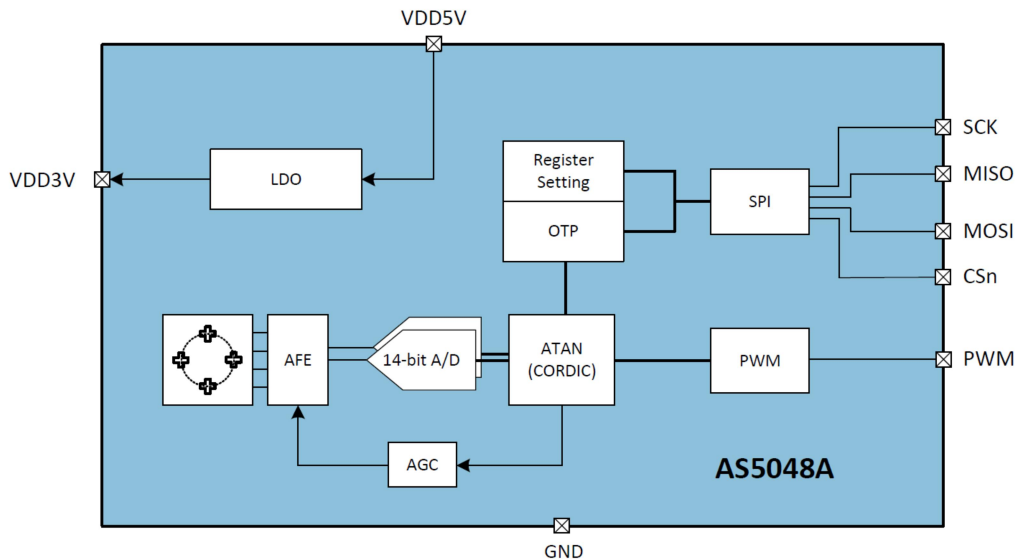


Figure 22 High-level block diagram of the AS5048A magnetic encoder

Pertinent performance details for the AS5048A are given in Table 3. Additional details were provided in §4 of [1]. One of the more interesting specifications in this table is the limited range for the sampling rate f_{sample} .

From Table 6 and Table 7, the SPI interface to the magnetic encoders is supported as given in Table 2. The AS5048A accepts either 5V or 3.3V power with 5V being chosen for convenience with the Launchpad DSP board.

Table 2 SPI Interface to Magnetic Encoders

Signal	Connector	GPIO	Power/Gnd	Connector
SOMIC	J2-18	123		
MOSIC	J2-17	122		
SPICLKC	J5-48	52		
ENCEN1 (RA)	J6-58	131		
ENCEN2 (Dec)	J6-57	130		
Encoder-1 Power			+5V	J3-21
Encoder-1 Ground			Gnd	J3-22
Encoder-2 Power			+5V	J7-61
Encoder-2 Ground			Gnd	J7-62

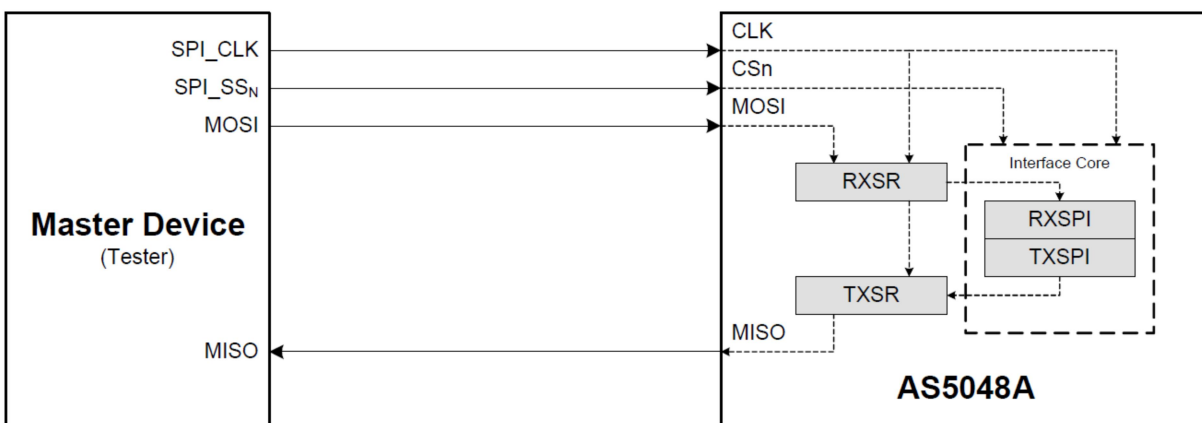
⁹ The AS5048A uses a SPI interface whereas the AS5048B uses a I²C interface.

Table 3 AS5048 System Performance [13]

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
B _z	Magnetic input field		30	50	70	mT
f _{sample}	Output sampling rate		10.2	11.25	12.4	kHz
RES	Output resolution		14			Bit
Noise	Sensor output noise	2.73LSB@14bit, rms value			0.06	deg
t _{prop}	System propagation delay		90.7	100	110.2	μs
f _{PWM}	PWM frequency		0.907	1	1.102	kHz
INL _{OPT} @25°C	Non-linearity, optimum placement of the magnet				± 0.8	deg
INL OPT+TEMP	Non-linearity optimum placement of the magnet over the full temperature range				±1	deg
INL DIS+TEMP	Non-linearity @ displacement of magnet and temperature -40°C to 150°C	Assuming N35H magnet (D=8mm, H=3mm) 500μm displacement in x and y z-distance @ 2000μm			±1.2	deg
t _{startup}	Startup time				10	ms

Different interface options are possible when using two AS5048A encoders as shown in Figure 23 through Figure 25 with the separate chip-enable configuration adopted for this project as shown in Figure 24.

The separate chip-enable configuration of Figure 24 has been chosen for this GoTo project.

**Figure 23** 4-wire mode with a single AS5048A encoder

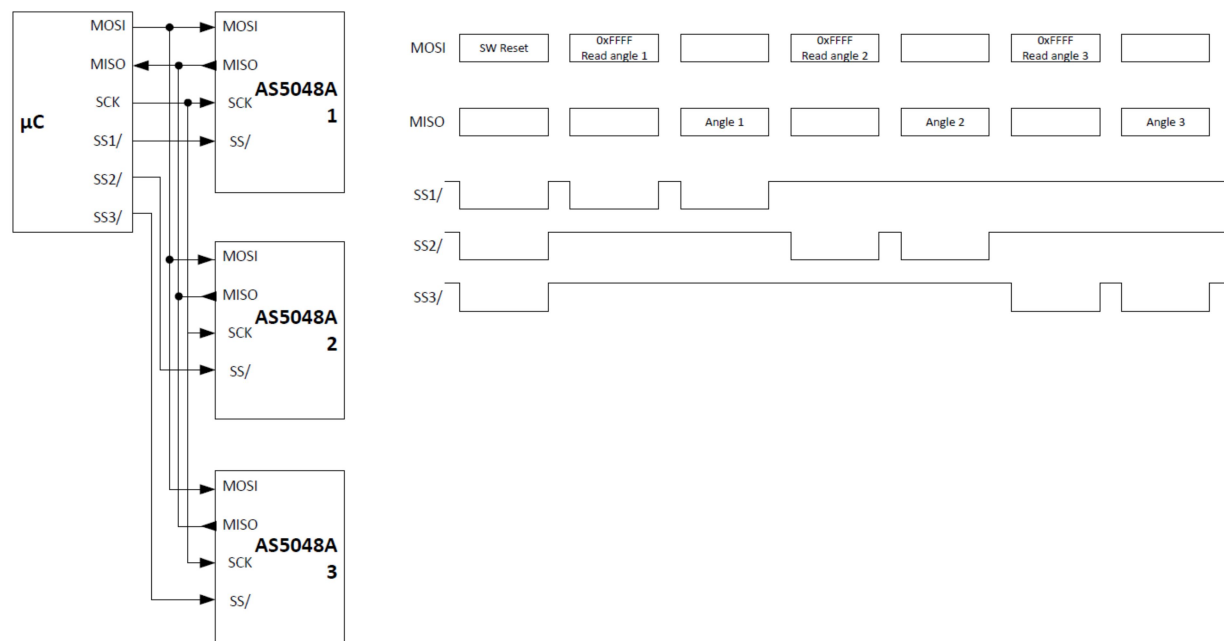


Figure 24 Multiple encoder arrangement using separate chip-select signals

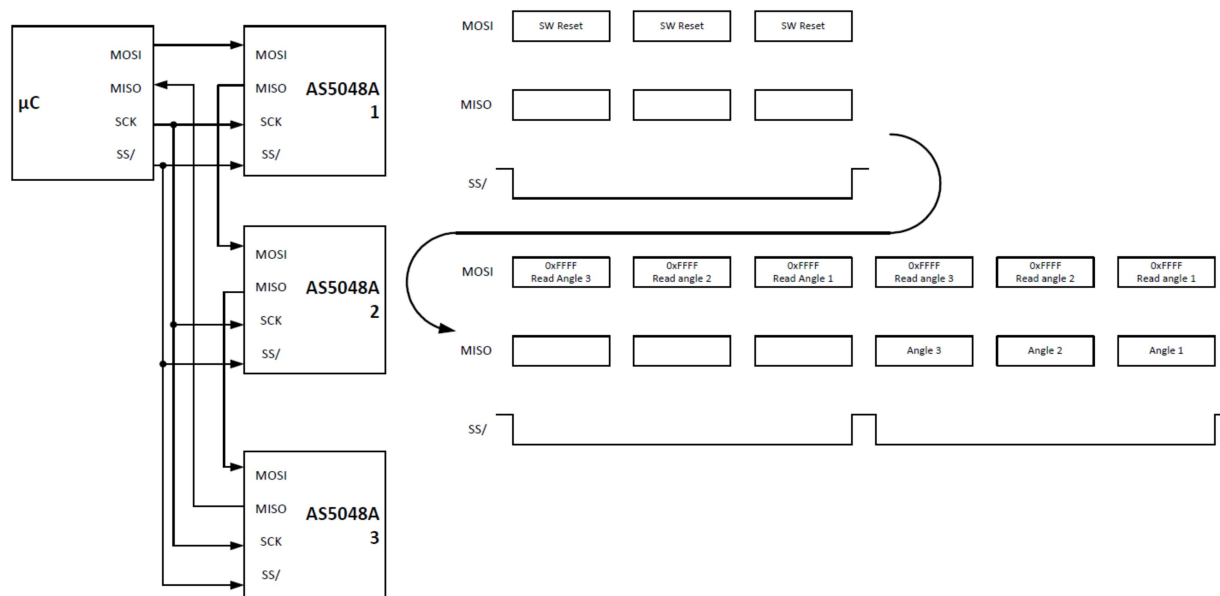


Figure 25 Daisy chain configuration for multiple encoders. A single encoder failure likely interrupts all encoder operation, however. The GoTo project only requires two encoders so the daisy-chaining route would only save one wire. This methodology condenses the encoder-pair query down to basically one *larger* read operation which is attractive but directing a command to an individual encoder does not appear to be supported.

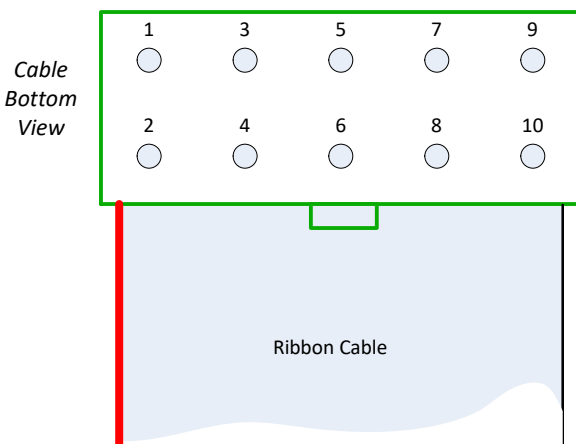


Figure 26 Encoder ribbon cable, bottom side¹⁰

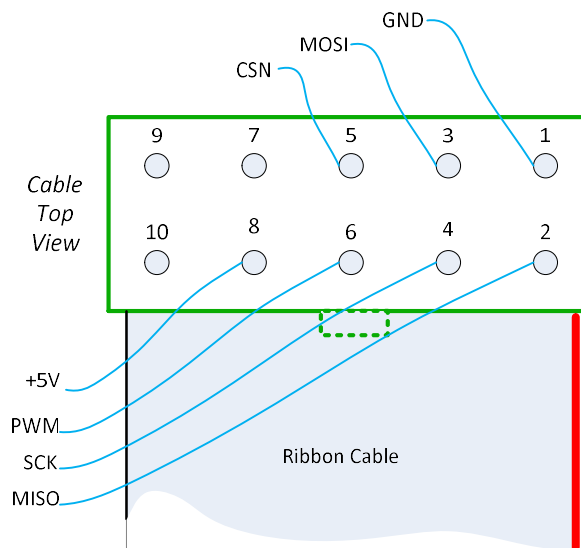


Figure 27 Encoder ribbon cable, top side¹¹

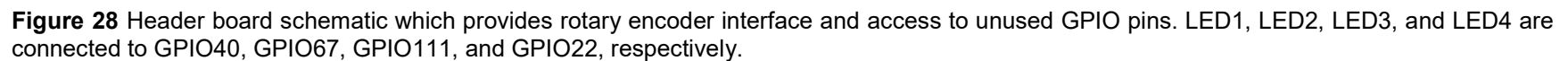
Table 4 DSP GPIO Signal Mapping to Rotary Encoder Headers

Encoder	Encoder Signal	Cable Pin No.	DSP GPIO	DSP Board Pin Number ¹²
RA	Gnd	1 (Red)	NA	
RA	MISO	2	123	J2-16 P123
RA	MOSI	3	122	J2-14 P122
RA	SCK	4	52	J5-5 P52
RA	CSn	5	131	J6-16 P131
RA	PWM	6	NA	
RA	3.3V	7	NA	
RA	5.0V	8	NA	
RA	---	9		
RA	---	10		
DEC	Gnd	1 (Red)	NA	
DEC	MISO	2	123	J2-16 P123
DEC	MOSI	3	122	J2-14 P122
DEC	SCK	4	52	J5-5 P52
DEC	CSn	5	130	J6-16 P130
DEC	PWM	6	NA	
DEC	3.3V	7	NA	
DEC	5.0V	8	NA	
DEC	---	9		
DEC	---	10		

¹⁰ U29849_Encoder_Connector_Wiring.vsd

¹¹ U29849_Encoder_Connector_Wiring.vsd

¹² From J-columns in Table 7



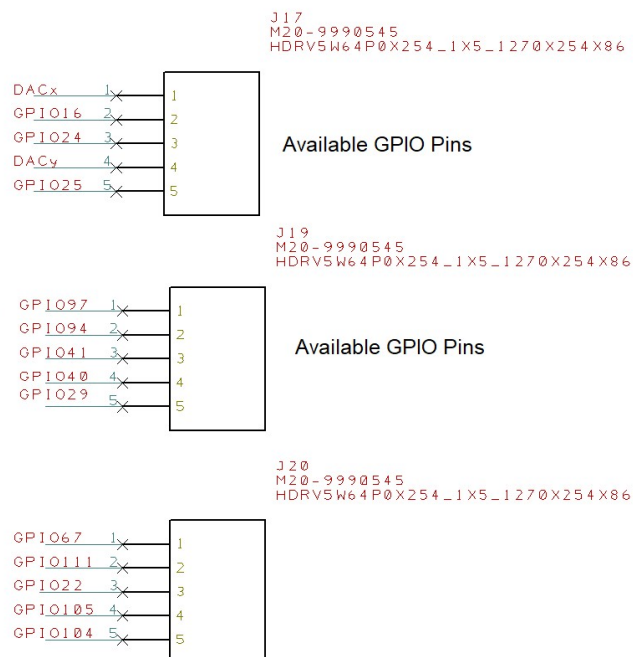


Figure 29 Close-up of header board GPIO connectors

Table 5 DSP Header Board Signal Access

Connector	Pin	GPIO #	Function	Comments
15	1	DACx	Filtered output from DSP DAC1	
	2	16	OUTPUTXBAR7 / GPIO16 / SPISIMOA	
	3	24	OUTPUTXBAR1 / GPIO24 / SPISIMOB	
	4	DACy	Filtered output from DSP DAC2	
	5	25	GPIO25 / OUTPUTXBAR2 / SPISOMIB	
19	1	97	GPIO97	
	2	94	GPIO94	
	3	41	GPIO41 / SCLB	
	4	40	GPIO40 / SDAB	Also connected to LED1
	5	29	GPIO29 / OUTPUTXBAR6 / SCITXDA	
20	1	67	GPIO67	Also connected to LED2
	2	111	GPIO111	Also connected to LED3
	3	22	GPIO22 / SCITXDB / SPICLKB	Also connected to LED4 Available from CPUs and CLAs.
	4	105	GPIO105 / SCLA / SCIRXDD	
	5	104	GPIO104 / SDAA / SCITXDD	

Rotary Encoder Headers

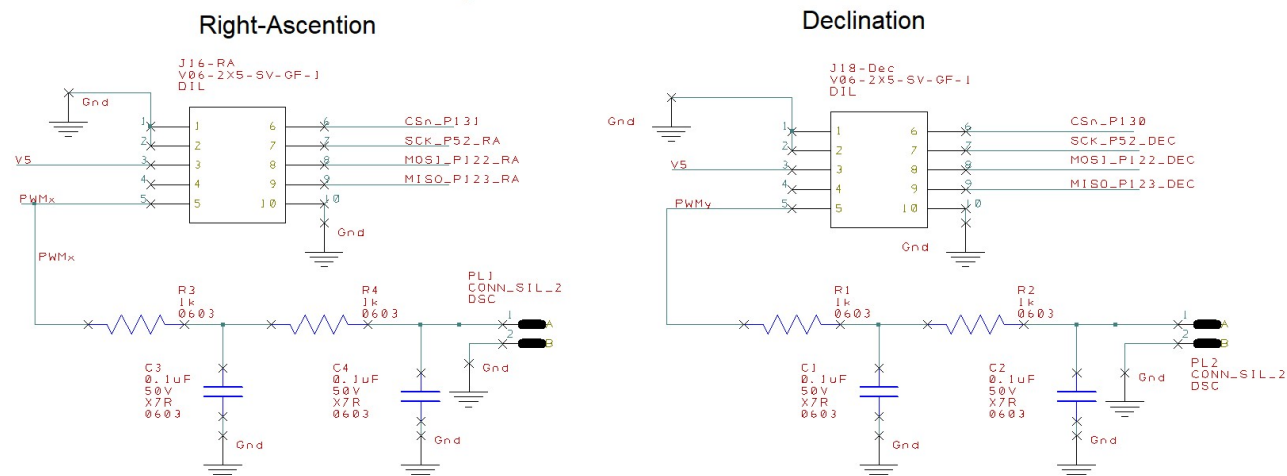


Figure 30 DSP header board connector pinout for rotary encoders



GoTo_Header_Board
_V2 - 24Feb2025.pdf

6 Late Breaking News

Software development on the TMS320F28379D has been both invigorating as well as challenging. Long time-gaps during the code development seriously impacted my efficiency. Using 3-phase motors and motor-controllers seriously increased project complexity while at the same time presenting new learning opportunities for me.

As I look toward the additional goals of this project, it is with some sadness that I have decided to abandon the TMS320F28379D in favor of the Raspberry Pi 5 as the computational and control heart of this project.

Even though I have a substantial time and effort invested into the TMS320, the full scope of my project has out grown the capabilities of this device as suggested by the complexity level shown in Figure 12. More specifically, the following points supported my decision:

- The TMS320F28379D (aka DSP) is a fairly old device, having been initially introduced into the marketplace by Texas Instruments in 1983.
- The DSP has only 2 cores, and a maximum clock rate of about 200 MHz. The core processors primarily support 32-bit computations (floats). Although double-precision computations are also supported, these come at considerable computational expense.
- Although I was able to skirt around memory size issues on the DSP thus far in my project, the future jump to video image processing would have invalidated this comment.
- Staying with the DSP would have forced some other processor (e.g., PC, Raspberry Pi) to host the graphical user interface (GUI)
- Software development efforts are generally increased as the number of device interfaces gets larger. This can especially be true with debugging issues.
- The cost of an individual 3-phase motor controller (DRV8301) has increased now to almost \$100, and the 2-axis telescope mount requires two controllers.
- Communication with my sensors in Figure 12 was often corrupted by the high electromagnetic interference (EMI) associated with the PWM motor control driver signals (6). Final packaging of the electronics for completely glitch-free operation has been a concern because it was going to involve something beyond tight wiring and ferrite-based noise suppression.

Looking forward toward additional project elements :

- I had already decided to do the video image processing in a Raspberry Pi 5 (RP5) for simplicity, as well as cost-point targets
- Although I really liked the GUI capabilities of running C# on a PC, needing a PC was a real detriment to portability as well as cost. I subsequently found that Python-based GUI capabilities on the RP5 to be more than sufficient for my needs.
- Since the friction drive step-down ratio is quite large¹³, the resolution of simple NEMA17 stepper-motors is sufficient for my project.
 - Controlling stepper-motors is far more simple and less expensive than controlling 3-phase motors
 - Stepper motors should result in less EMI and consequently fewer integration issues
 - Without using any micro-stepping of the motors, the pointing resolution is about 5.44 arc-seconds. Invoking 8:1 micro-stepping improves the resolution further to 0.68 arc-seconds
 - Adopting NEMA17 motors equipped with harmonic-drive reduction can provide pointing resolutions of less than 0.18 arc-seconds without any reliance on micro-stepping.

¹³ Main disk is 7.5" in diameter, with a drive-axle diameter of 8mm which translates to a step-down ratio of about 23.8125:1. In addition, the worm-gear arrangement provides another 50:1 step-down ratio resulting in an aggregate step-down ratio for the friction drive portion of the mount being about 1,190.627:1.

- The ultra-fine resolution provided by the NEMA17+harmonic-drive would, however, compromise the maximum telescope angular slew rates substantially.
- Even though the RP5 does not have computational hardware blocks like the DSP's CLAs, (i) the RP5 is 64-bit technology rather than 32-bit, (ii) high-speed memory resources of the RP5 dwarf the DSP capabilities, (iii) the RP5 has 4 cores rather than 2, (iv) the baseline RP5 clock rate is 1.5 GHz, and (v) the RP5 natively supports two video cameras whereas the DSP has no such capability.
- Accommodating stepper-motors in the friction-drive rather than 3-phase motors only involves milling slightly different motor mounts.
 - NEMA17 stepper-motors are about 4x cheaper than 3-phase motors
 - Many stepper-motor controller options are available whereas the DSP route only has the single TI controller as an option.
- RP5 supports multiple programming languages (e.g., C/C++/Python) whereas the DSP is limited to a subset of C++.
- The user community for the RP5 is much larger than for the DSP.
- Many more software libraries and tools are available for the RP5.

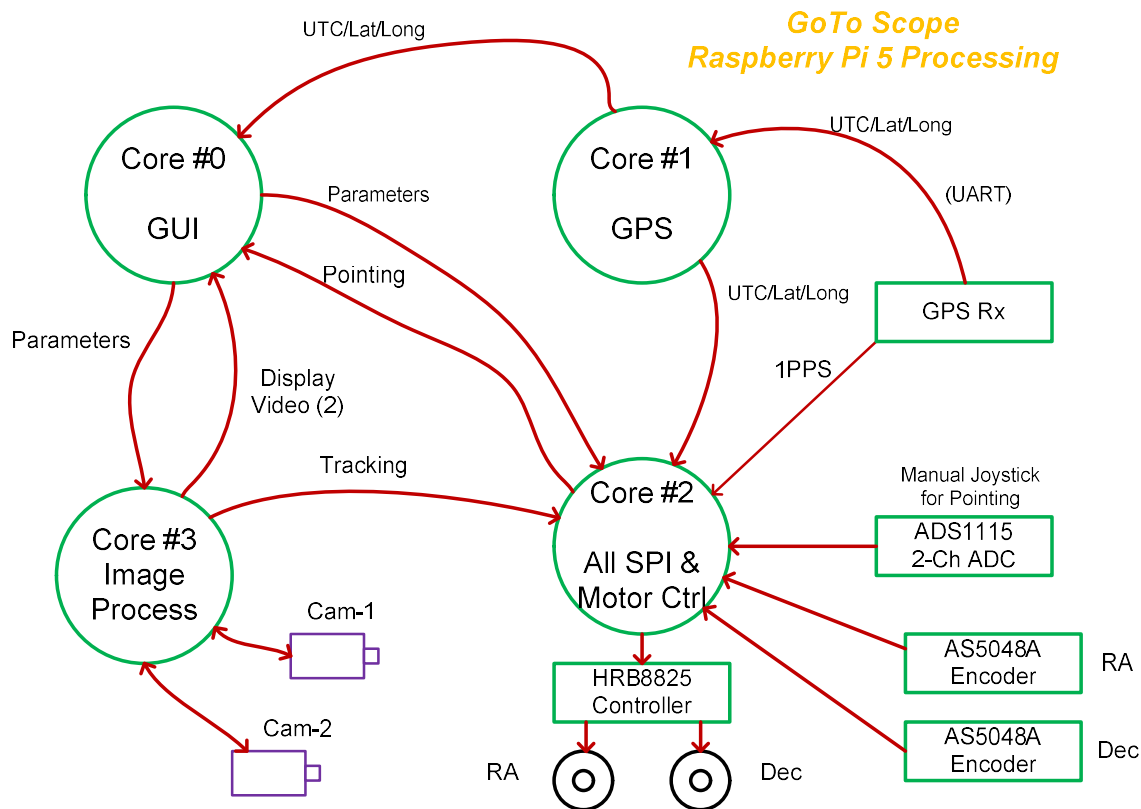


Figure 31 GoTo software plan for Rapsberry Pi 5 multiple cores

7 References

1. J.A. Crawford, "My Go-To Mount," Part 1, Version 2.0, 20 July 2024, U28639.
2. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part 7," August 2023.
3. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part 6AB," October 2022.
4. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part 6A," January 2022.
5. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part V: Optical Encoder Completion and Transition to TI DSP," January 2021, U24933.
6. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part V," U24933, January 2021.
7. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part IV: Optical Encoder and Elevation Drive," U24933, August 2020.
8. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part III," U24933, November 2019.
9. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part II, Direct-Drive Motors," U24933, April 2019.
10. _____, "Photogrammetry for Non-Invasive Terrestrial Position/Velocity Measurement of High-Flying Aircraft- Part I," 2017, U24866.
11. Ziaoyin Zhu, "Tutorial on Hertz Contact Stress," 1 Dec. 2012, OPTI 521, U29017.
12. ams OSRAM, "AN5000 Rotary Magnetic Position Sensors," Sept. 28, 2016.
13. ams, "AS5048A/AS5048B Magnetic Rotary Encoder (14-Bit Angular Position Sensor)," [v1-11] 2018-Jan-29, file: AS5048_DS000298_4-00_ZZZZZZZ_Rotary_Encoder.pdf.
14. _____, *Advanced Phase-Lock Applications- Synthesis*, Chapter 5, unpublished.
15. Benedict, T.R., and G.W. Bordner, "Synthesis of an Optimal Set of Radar Track-While-Scan Smoothing Functions," *IRE Trans. on Automatic Control*, July 1962.
16. Texas Instruments, "Debugging Multiple Cores," U29501.
17. _____, "DRV8301 Three-Phase Gate Driver with Dual Current Shunt Amplifiers and Buck Regulator," Jan. 2016, U29623.
18. grok, "TMS320F28379D Memory Allocation," U29512.

8 Appendix: Launchpad I/O with Dual DRV8301s Installed

SPI-A and SPI-B are dedicated to the two DRV8301, only leaving SPI-C available to communicate with the *two* magnetic encoders. Separate chip-select lines can, however, be accommodated. I/O ports in *green* are newly assigned for this project. Available I/O ports are denoted in *light blue*.

Table 6 DRV8301 I/O Mapping¹⁴ to F28379D Configured for Booster Pack Situated Closest to USB Connector for Elevation Axis (EPWM1-3)

DRV8301 Signal	F28379D Pins				DSP GPIO	F28379D Signal	Non-DRV8301 Usage
	J1	J3	J4	J2			
3.3V	1					3.3V	
x	2				32	GPIO32 / SDAA	LED1 Power
FAULT	3				19	GPIO19 / SCIRXDB / SPISTEA*	
OCTW	4				18	GPIO18 / SCITXDB / SPICLKA	
x	5				67	GPIO67	Available on DSP J1
x	6				111	GPIO111	Available on DSP J1
SCLK	7				60	GPIO60 / SPICLKA / SPISIMOB	
x	8				22	GPIO22 / SCITXDB / SPICLKB	Used by CLA1 during development for measuring algorithm execution times. Also drives LED4 on Figure 13
x	9				105	GPIO105 / SCLA / SCIRXDD	Available on DSP J1
x	10				104	GPIO104 / SDAA / SCITXDD	Available on DSP J1
x		21				5V	5V_ENCODER_2
GND		22				GND	GND_ENCODER_2
DC-V-FB		23				ADCIN14 / CMPIN4P	
VA-FB		24				ADCINC3 / CMPIN6N	
VB-FB		25				ADCINB3 / CMPIN3N	
VC-FB		26				ADCINA3 / CMPIN1N	
IA-FB		27				ADCINC2 / CMPIN6P	
IB-FB		28				ADCINB2 / CMPIN3P	
IC-FB		29				ADCINA2 / CMP1N1P	
x		30				ADCINA0 / DACOUTA	
PWM-AH			40		0	EPQM1A / GPIO0	
PWM-AL			39		1	EPWM1B / GPIO1	
PWM-BH			38		2	EPWM2A / GPIO2	
PWM-BL			37		3	EPWM2B / GPIO3	
PWM-CH			36		4	EPWM3A / GPIO4	
PWM-CL			35		5	EPWM3B / GPIO5	
x			34		24	OUTPUTXBAR1 / GPIO24 / SPISIMOB	Available on DSP J4
x			33		16	OUTPUTXBAR7 / GPIO16 / SPISIMOA	Available on DSP J4
x			32			DAC1	
x			31			DAC2	
GND				20		GND	

¹⁴ From U26355 LAUNCHXL-F28379D Overview, SPRUI77C, August 2016, March 2019. Not all GPIO etc pins are mapped to J-connectors. See U27334 F28379D Datasheet sprs880k for details. Originally appeared as Table 6-2 in [3]. Non-DRV8301 Usage column modified here, however.

DRV8301 Signal	F28379D Pins				DSP GPIO	F28379D Signal	Non-DRV8301 Usage
	J1	J3	J4	J2			
SCS				19	61	GPIO61 / SPISIMOB / SPISTEA*	
x				18	123	GPIO123 / SD1_C1 / SPISOMIC	SPISOMIC used for both encoders
x				17	122	GPIO122 / SD1_D1 / SPISIMOC	SPISIMOC used for both encoders
x				16		RST	
SDI				15	58	GPIO58 / SPISIMOA	
SDO				14	59	GPIO59 / SPISOMIA	
EN-GATE				13	124	GPIO124 / SD1_D2	
DC-CAL				12	125	GPIO125 / SD1_C2	
x				11	29	GPIO29 / OUTPUTXBAR6 / SCITXDA	Available on DSP J2

Table 7 DRV8301 I/O Mapping to F28379D Configured for Booster Pack situated furthest from USB connector for RA axis (EPWM4-6)¹⁵

DRV8301 Signal	F28379D Pins				DSP GPIO	F28379D Signal	Non-DRV8301 Usage
	J5	J7	J8	J6			
3.3V	41					3.3V	
x	42				95	GPIO95	1PPS from GPS Receiver
FAULT	43				139	GPIO139 / SCIRXDC	
OCTW	44				56	GPIO56 / SCITXDC	
x	45				97	GPIO97	Available on DSP J5
x	46				94	GPIO94	Available on DSP J5
SCLK	47				65	GPIO65 / SPICLKB	
x	48				52	GPIO52 / SPICLKC	SPICLKC for Magnetic Encoders <i>Must</i> be shared by the two encoders because SPICLKA and SPICLKB are dedicated to the DRV8301's
x	49				41	GPIO41 / SCLB	Available on DSP J5
x	50				40	GPIO40 / SDAB	Available on DSP J5
x		61				5V	5V_ENCODER_1
GND		62				GND	GND_ENCODER_1
DC-V-FB		63				ADCIN15 / CMPIN4N	
VA-FB		64				ADCINC5 / CMPIN5N	
VB-FB		65				ADCINB5	
VC-FB		66				ADCINA5 / CMPIN2N	
IA-FB		67				ADCINC4 / CMPIN5P	
IB-FB		68				ADCINB4	
IC-FB		69				ADCINA4 / CMPIN2P	
x		70				ADCINA1 / DACOUTB	
PWM-AH			80		6	EPWM4A / GPIO6	
PWM-AL			79		7	EPWM4B / GPIO7	
PWM-BH			78		8	EPWM5A / GPIO8	
PWM-BL			77		9	EPWM5B / GPIO9	
PWM-CH			76		10	EPWM6A / GPIO10	
PWM-CL			75		11	EPWM6B / GPIO11	
x			74		14	GPIO14 / OUTPUTXBAR3 / SCITXDB	SCITXDB for RP4 UART
x			73		15	GPIO15 / OUTPUTXBAR4 / SCIRXDB	SCIRXDB for RP4 UART

¹⁵ Originally appeared as Table 6-3 in [3]. Non-DRV8301 Usage column modified here, however.

DRV8301 Signal	F28379D Pins				DSP GPIO	F28379D Signal	Non-DRV8301 Usage
	J5	J7	J8	J6			
x			72			DAC3	
x			71			DAC4	
GND				60		GND	
SCS				59	66	GPIO66 / SPISTEB*	
x				58	131	GPIO131 / SD2 C1	ENCEN1 – Encoder 1 Enable
x				57	130	GPIO130 / SD2 D1	ENCEN2 – Encoder 2 Enable
x				56		RST	
SDI				55	63	SPISIMOB / GPIO63	
SDO				54	64	SPISOMIB / GPIO64	
EN-GATE				53	26	GPIO26 / SD2 D2	
DC-CAL				52	27	GPIO27 / SD2 C2	
x				51	25	GPIO25 / OUTPUTXBAR2 / SPISOMIB	Available

9 Appendix: 3-Phase Interpolation Algorithms

Unlike most motor control algorithms in the literature, I need to have very precise (e.g., 10 arc-second) precision pointing making jitter or wandering of either axis critical concerns. Since the associated algorithms must be executed at the sampling rate (i.e., perhaps as high as 160 ksps), execution efficiency is critical as well. Making use of the build-in CLA's within the F28379D should help ease memory as well as execution resource concerns.

For 3-phase control, the voltage waveforms given by

$$\begin{aligned} V_A &= A_A \sin(\theta) \\ V_B &= A_B \sin\left(\theta + \frac{4}{3}\pi\right) = A_B \left[\frac{\sqrt{3}}{2} \cos(\theta) - \frac{1}{2} \sin(\theta) \right] \\ V_C &= A_C \sin\left(\theta - \frac{4}{3}\pi\right) = A_C \left[\frac{\sqrt{3}}{2} \cos(\theta) + \frac{1}{2} \sin(\theta) \right] \end{aligned} \quad (5)$$

must be efficiently computed. A sine (and cosine) table of sufficiently small granularity along with numerical interpolation is the most economical way to proceed. The 2nd order interpolation method developed in [14] approximates the in-phase and quadrature-phase terms as¹⁶

$$\begin{aligned} I &\cong \sim I_{table} \left(1 - \frac{\phi^2}{2} \right) - Q_{table} \phi \\ Q &\cong Q_{table} \left(1 - \frac{\phi^2}{2} \right) + I_{table} \phi \end{aligned} \quad (6)$$

where the angle of interest is given by $\theta = k \Delta\theta + \phi$ where the angular step between table look-up values is given by $\Delta\theta$ and $|\phi| \leq |\Delta\theta / 2|$. Interpolated performance using (6) versus ideal are shown in Figure 32 through Figure 37 for an effective table lookup length¹⁷ of 160.

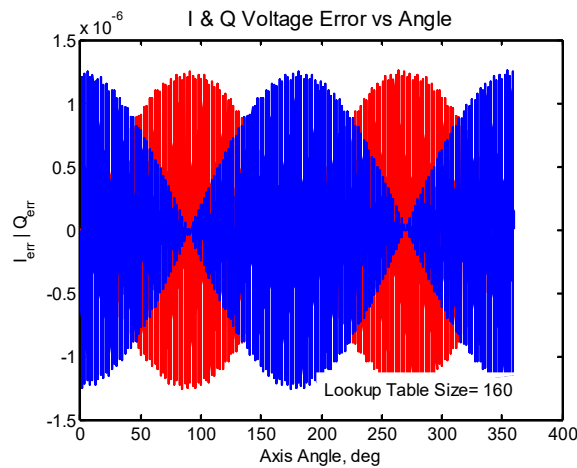
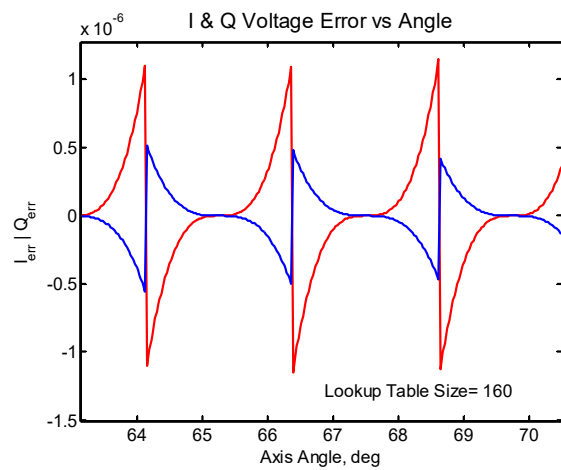
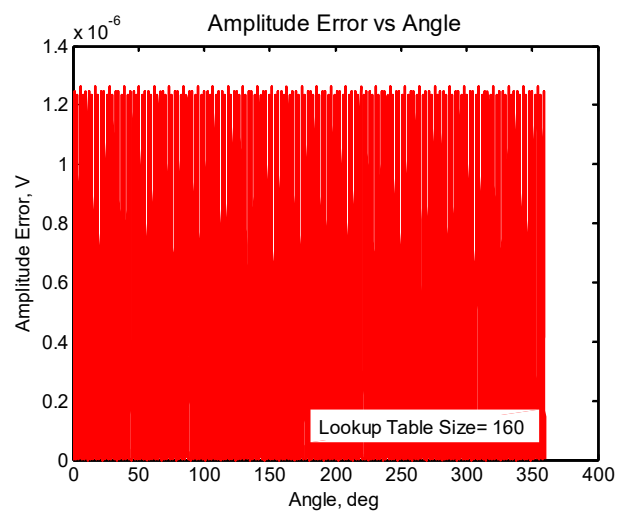
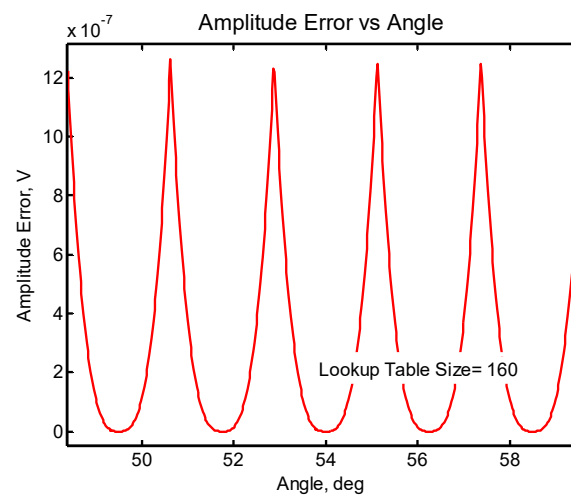


Figure 32 In-phase and quadrature-phase errors

¹⁶ Ideally, $I = \cos(\theta)$ and $Q = \sin(\theta)$.

¹⁷ Due to symmetry arguments, the physical lookup table length needs to only be 1/8 of the effective table length.

**Figure 33** Close-up of Figure 32**Figure 34** Amplitude error**Figure 35** Close-up of Figure 34

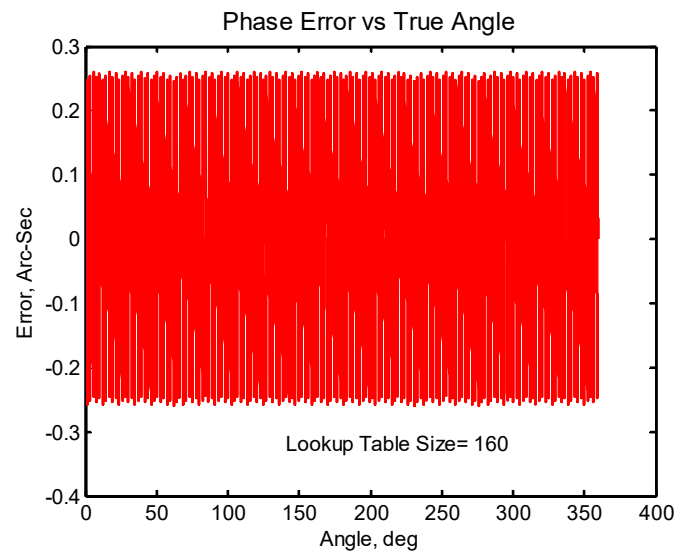


Figure 36 Phase error

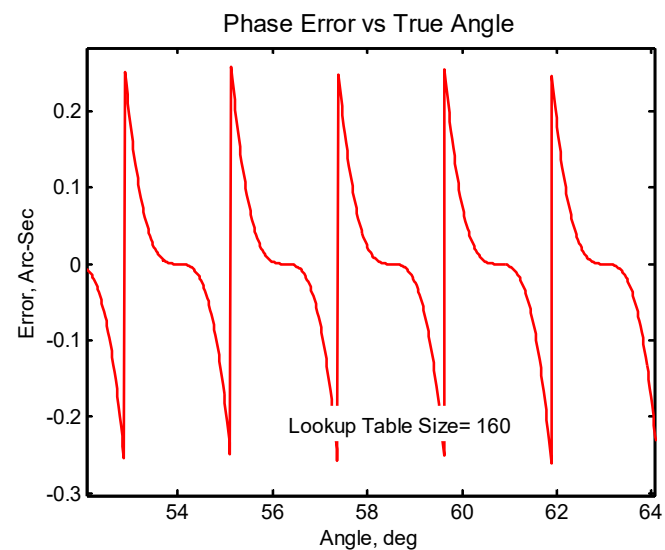


Figure 37 Close-up of Figure 36

Table 8 Interpolation Error Performance vs Table Lookup Size

Effective Lookup Table Size	Pk-Pk Amplitude Error, V	Pk-Pk Phase Error, Arc-Seconds
64	1.97e-5	8.13
128	2.46e-6	1.02
160	1.26e-6	0.52
192	7.30e-6	0.298
256	3.08e-7	0.127

9.1 ePWM Algorithms

9.1.1 ePWM Drive Voltage Limitations

Precision motor control is required for both the RA and Dec axes. This is accomplished using 3-phase motors where the driving voltages are created using pulse-width modulation techniques. Multiple constraints must be taken into account with using the ePWM approach. More specifically, this project uses the DRV8301 add-on device from Texas Instruments to create the high-voltage / high-current driving waveforms. The DRV8301 is limited to:

- Maximum pulse rate = 200 kpps¹⁸
- Maximum peak-current = 14 A
- MOSFET switch on-resistance = 0.013Ω
- Maximum V_{cc} = 24V

The 3-phase motors present more constraining requirements, however, as detailed in Table 9. Note in particular the peak-current limits which if exceeded, can result in damage to the motor!

Table 9 42BLSxx 3-Phase Motor Parameters

Model		42BLS01	42BLS02	42BLS03	42BLS04
Number of poles		8			
Number of phase		3			
Rated voltage	VDC	24			
Rated speed	RPM	4000			
Continuous stall torque	N.m	0.075	0.15	0.22	0.3
Rated torque	N.m	0.0625	0.125	0.185	0.25
Rated power	W	26	52.5	77.5	105
Peak torque	N.m	0.19	0.38	0.56	0.75
Peak current	A	6	10.8	15.5	21.7
Line to line resistance	ohms	1.5	0.8	0.43	0.3
Line to line inductance	mH	2.1	1.2	0.71	0.5
Torque constant	N.m/A	0.033	0.036	0.037	0.035
Back E.M.F	Vrms/Krpm	2.45	2.71	2.74	2.62
Rotor inertia	g.cm ²	24	48	72	96
Body length	mm	40.3	60.3	80.3	100.3
Mass	Kg	0.3	0.45	0.65	0.8

When the 3-phase motors are being used for positioning (i.e., ≈ 0 RPM), the voltages applied to the motor are basically DC in nature. Referring to Figure 39, when one of the voltage waveforms is at its peak, the other two voltage waveforms are at one-half of the peak value since

$$\begin{aligned}
 \sin(90^\circ) &= 1 \\
 \sin(90^\circ + 120^\circ) &= -0.50 \\
 \sin(90^\circ - 120^\circ) &= -0.50
 \end{aligned}
 \tag{7}$$

This results in the line-to-line voltage across each winding pair being 1.5 times the peak-value which equates to $1.5 \times V_{pk}$. This result along with Ohm's Law can be used to show that

¹⁸ kpps is kilo-pulses per second.

$$V_{pk} \leq \frac{4}{3} I_{pk} R_w \quad (8)$$

Using the information in Table 9 leads to the important results

$$\begin{aligned} V_{pk} &\leq 6V && \text{for 42BLS01} \\ V_{pk} &\leq 5.76V && \text{for 42BLS02} \\ V_{pk} &\leq 4.44V && \text{for 42BLS03} \end{aligned} \quad (9)$$

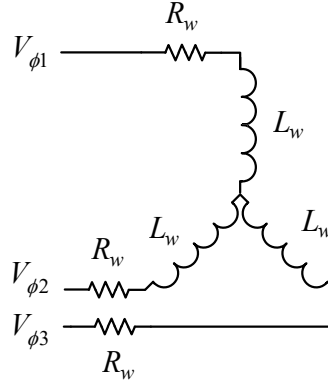


Figure 38 3-phase motor winding schematic¹⁹

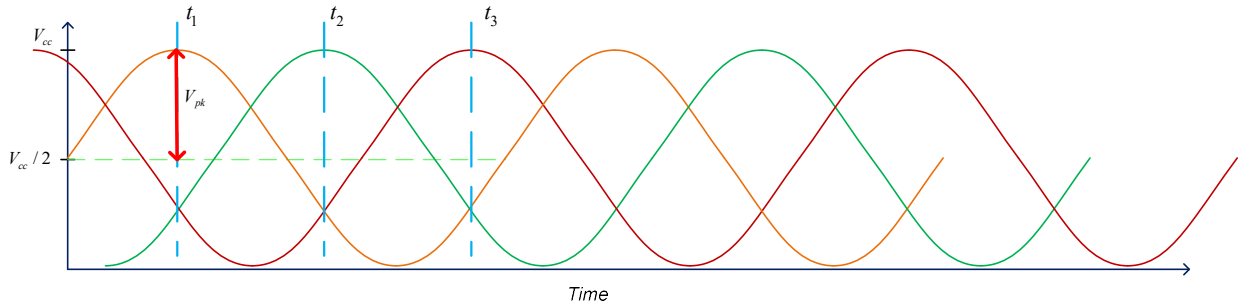


Figure 39 Full-scale 3-phase voltage waveforms²⁰

ePWM linearity is improved if the supply voltage $V_{cc} > 2V_{pk}$ because the resulting pulse-width duty factor spans a smaller range given by

$$\frac{\frac{V_{cc}}{2} - V_{pk}}{V_{cc}} \leq d \leq \frac{\frac{V_{cc}}{2} + V_{pk}}{V_{cc}} \quad (10)$$

which simplifies to

$$\frac{1}{2} - \frac{V_{pk}}{V_{cc}} \leq d \leq \frac{1}{2} + \frac{V_{pk}}{V_{cc}} \quad (11)$$

¹⁹ From U29152_Figures.vsd.

²⁰ From U29152_Figures.vsd.

Equation (11) is tabulated for several different supply voltage values and three different motor choices in Table 10. These guidelines for duty cycle must be followed in order to prevent damage to the motor and or excessive heating.

Table 10 Duty Cycle Ranges Versus 3-Phase Motor and Supply Voltage²¹

PWM Duty Factors									
		Vcc, V	42BLS01		42BLS02		42BLS03		
			dmin	dmax	dmin	dmax	dmin	dmax	
		12	0.00	1.00	0.02	0.98	0.13	0.87	
		18	0.17	0.83	0.18	0.82	0.25	0.75	
		24	0.25	0.75	0.26	0.74	0.32	0.69	

9.1.2 ePWM Pulse Rate

The TMS320F28379D in combination with the DRV8301 3-phase motor drivers are limited to a pulse repetition rate of about 200 kpps. The period for one ePWM cycle is programmatically given by the parameter xxxxxx where

$$T_{period} = 40nsec \times asdfasd \quad (12)$$

Since the period is given in terms of 40 nsec segments, the associated number of quantization bits is given by

$$N_{bits} = \frac{\log_e(\quad)}{\log_e(2)} \quad (13)$$

The ePWMs normally operate as a straight-forward pulse-width modulator. As such, synthesizing low-frequency sine waves leads to an output having many discrete spurious tones and appreciable jitter. In order to lessen this potential problem, the ePWM algorithm includes a second-order (1-1) Multi-stage-noise-Shaping (MASH) algorithm.

A classical MASH 1-1 block diagram is shown in Figure 40. This model can be linearized by replacing each quantizer with a summer and independent quantization noise source as shown in Figure 41. It is straight forward to use z-transforms to derive

$$\begin{aligned} y_1 &= xz^{-1} + q_1(1 - z^{-1}) \\ y_2 &= -q_1z^{-1} + q_2(1 - z^{-1}) \end{aligned} \quad (14)$$

from which

$$y_{out} = y_1z^{-1} + y_2(1 - z^{-1}) \quad (15)$$

²¹ From U29450_epwm_duty_factors.xlsx.

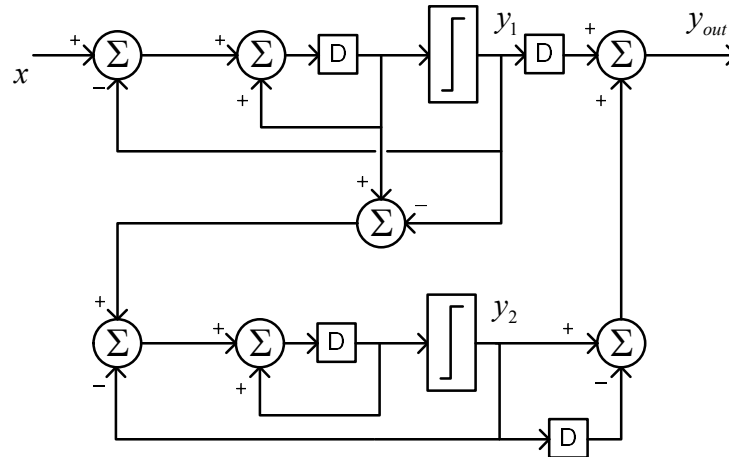


Figure 40 MASH (1-1) block diagram²²

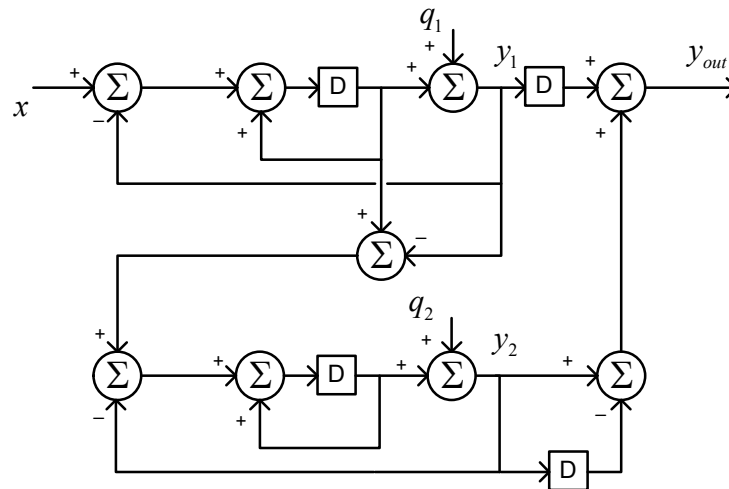


Figure 41 Linearized version²³ of Figure 40 using new quantization noise sources q_1 and q_2

Following through with (15) finally results in

$$y_{out} = xz^{-2} + q_2(1 - z^{-1})^2 \quad (16)$$

Processing

CLA1

- ePWM4, 5, 6 – int2

CLA1 Alternate:

- read encoder RA via SPIA
- read encoder Dec via SPIA

CPU1 Alternate:

- read ADC-D RA
- read ADC-D Dec

²² From U29152_Figures.vsd.

²³ From U29152_Figures.vsd.

Appendix: Observer²⁴

An observer is discussed in Dave Wilson's material on field oriented motor control [9] as shown in Figure 42. This filter configuration is also referred to as an *alpha-beta filter* by others and is actually a 2nd-order IIR filter as shown in Figure 43. Modern control system design based upon state-variable techniques routinely utilize the observer concept and relevant material is widely available.

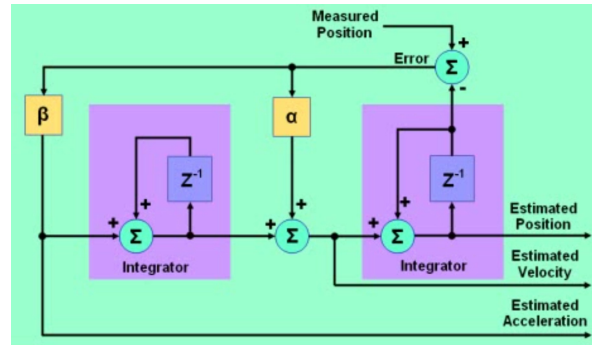


Figure 42 Observer by Dave Wilson [9] for use with an optical encoder, recast as a 2nd order IIR filter in Figure 43

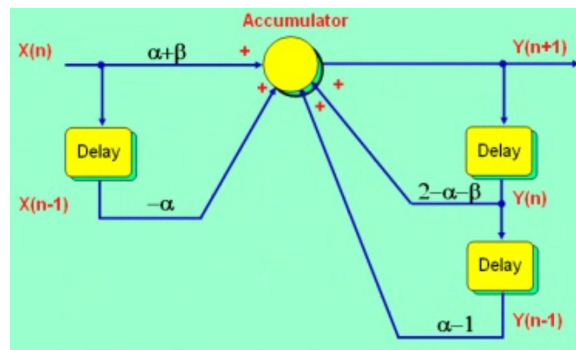


Figure 43 Alpha-beta filter of Figure 42 recast as a second-order IIR filter [9]

The transfer function between estimated position $Y(n)$ and input position $X(n)$ is given by

$$H(z) = X(z) \frac{(\alpha + \beta)z^{-1} - \alpha z^{-2}}{1 + z^{-1}(-2 + \alpha + \beta) + z^{-2}(1 - \alpha)} \quad (17)$$

The α and β values are generally kept small in order to suppress noise. Stability dictates that²⁵

$$\begin{aligned} 0 < \alpha < 1 \\ 0 < \beta \leq 2 \\ 0 < 4 - 2\alpha - \beta \end{aligned} \quad (18)$$

Parameter β must be < 1 or else noise will be amplified. These filters have a connection to Kalman filters but this will not be discussed further at this time.

²⁴ From [3].

²⁵ Wikipedia

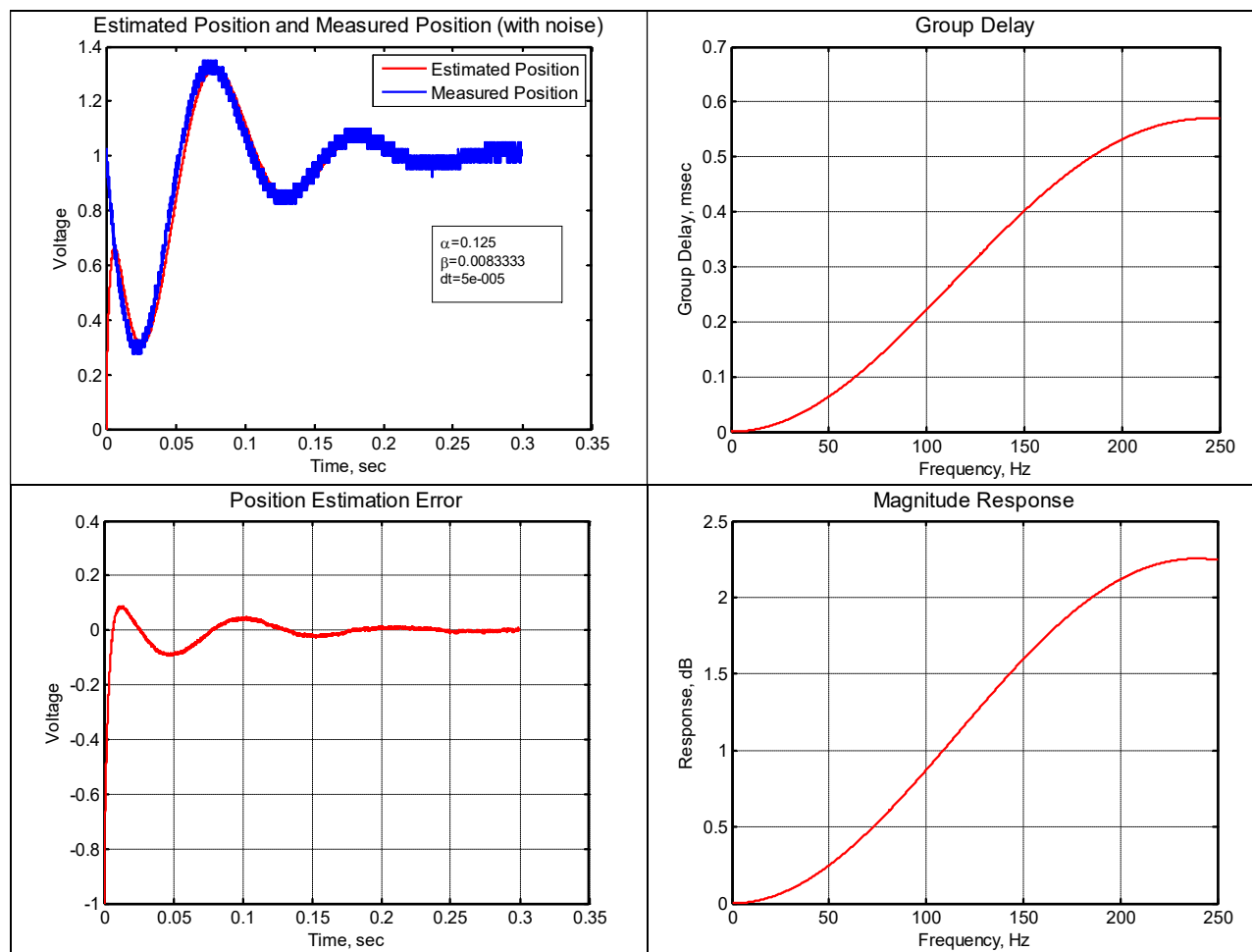
Benedict and Bordner came up with a relationship for β thereby simplifying parameter choices in 1962 [15]. Given a noisy system wherein the desired signal of interest is changing at a constant rate, they stipulated

$$\beta = \frac{\alpha^2}{2 - \alpha} \quad (19)$$

For situations where near-critical damping is desired, they recommended

$$\beta = 0.80 \left[\frac{2 - \alpha^2 - 2\sqrt{1 - \alpha^2}}{\alpha^2} \right] \quad (20)$$

Table 11 Example of Stand-Alone Observer Response to Quantized Damped Sinusoid Input²⁶ Using (19)



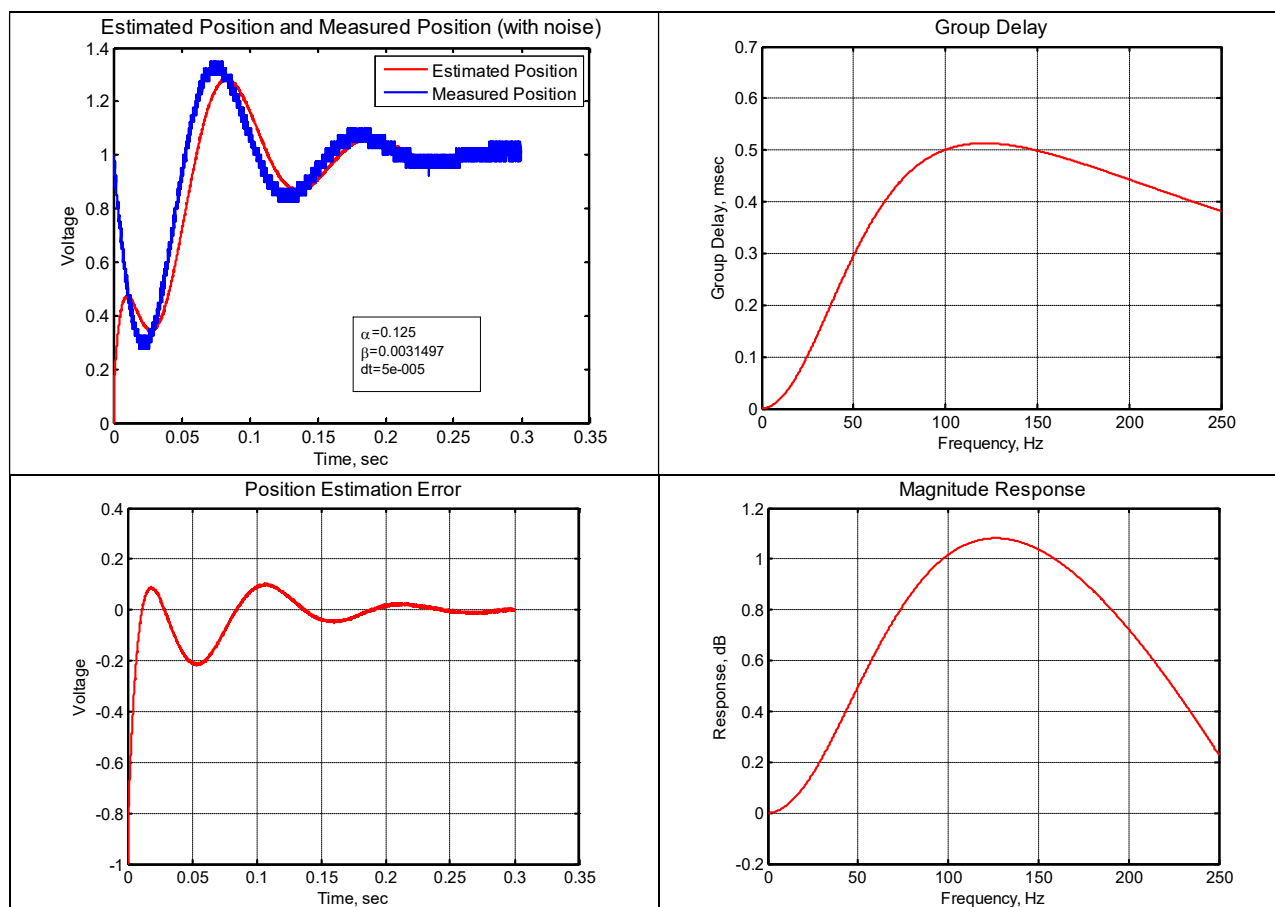
A comparison of results for $\alpha = 0.125$ in a system using a sampling rate of 20 kbps, quantization steps of 0.025V and uniformly distributed noise, these two approaches for β -choice are compared in Table 11 and Table 12. Based upon these abbreviated results, the baseline approach will be built around using (19) for now. In the context of Figure 42, the transfer functions of interest are

²⁶ Using u25863_tracking_loop.m.

$$H_{\theta}(z) = \frac{\theta_{est}(z)}{\theta_{meas}(z)} = \frac{(\alpha + \beta) - \alpha z^{-1}}{1 + z^{-1}(\alpha + \beta - 2) + z^{-2}(1 - \alpha)} \quad (21)$$

$$\begin{aligned} H_{\omega}(z) &= \frac{\omega_{est}(z)}{\theta_{meas}(z)} = H_{\theta}(z)(1 - z^{-1}) \\ &= \frac{(\alpha + \beta) - (2\alpha + \beta)z^{-1} + z^{-2}}{1 + z^{-1}(\alpha + \beta - 2) + z^{-2}(1 - \alpha)} \end{aligned} \quad (22)$$

Table 12 Example of Stand-Alone Observer Response to Quantized Damped Sinusoid Input²⁷ Using (20)



²⁷ Using u25863_tracking_loop.m.

10 Appendix: Smooth Telescope Move Commanding²⁸

10.1 Method 1

Smooth motion of the telescope mount in azimuth and elevation is important, particularly for large angular steps. One method was sketched out in §8 of [9] which will serve as the starting point for a later discussion in this section.

I originally considered using a lowpass-filtered frequency-step function for positioning the telescope (in azimuth and elevation) but the resultant movement was not at all smooth at the beginning of the associated move as shown in Figure 44, but rather abrupt. The associated Laplace transfer function for the 2nd-order all-pole lowpass filter is given by

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (23)$$

This can be converted to sampled-digital form using the bilinear transform leading to

$$\theta_o(z) = \theta_i(z) \frac{\omega_n^2 (1 + z^{-1})^2}{a + bz^{-1} + cz^{-2}} \quad (24)$$

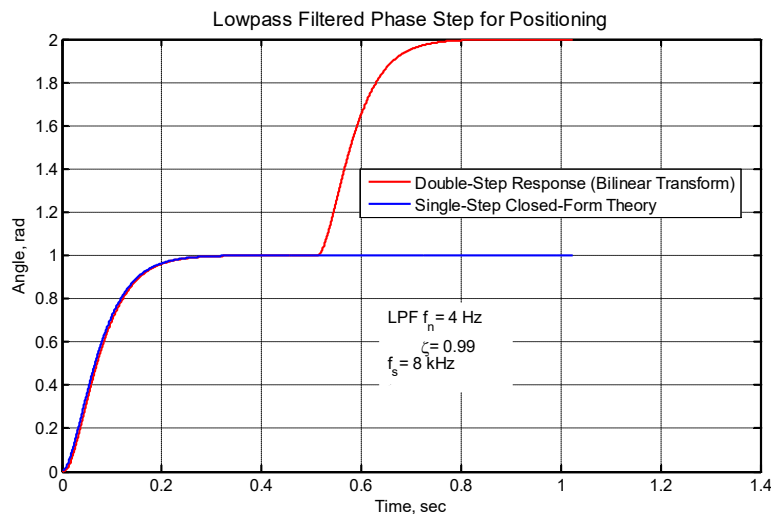


Figure 44 Lowpass-filtered phase step response²⁹

with

²⁸ From [10].

²⁹ u28404_smooth_positioner.m.

$$\begin{aligned}
 a &= \left(\frac{2}{T}\right)^2 + \frac{4\zeta\omega_n}{T} + \omega_n^2 \\
 b &= 2\omega_n^2 - \frac{8}{T^2} \\
 c &= \left(\frac{2}{T}\right)^2 - \frac{4\zeta\omega_n}{T} + \omega_n^2
 \end{aligned} \tag{25}$$

10.2 Method 2 (Originally from §8 of [9])

If an axle is to be rotated θ_{cmd} radians, this needs to be done (i) within a prescribed amount of time, (ii) with a specified degree of smoothness, and (iii) with a radian velocity no greater than a specified limit. These requirements are not mutually independent, however. For structural safety, it is better to think in terms of (iv) maximum angular velocity allowed and (v) maximum angular acceleration allowed since (v) limits the applied current and consequently the applied torque and (vi) limits the total rotational inertia allowed in the system.

With items (v) and (vi) in mind for large angular changes, the methodology outlined in Figure 45 achieves both objectives while being relatively simple. In this figure, the (positive) angular acceleration is limited to $a_{\theta_{max}} = 2\omega_{max} / T_r$ rad/sec² and the radian velocity is limited to ω_{max} rad/sec. A computed example³⁰ is shown in Figure 46.

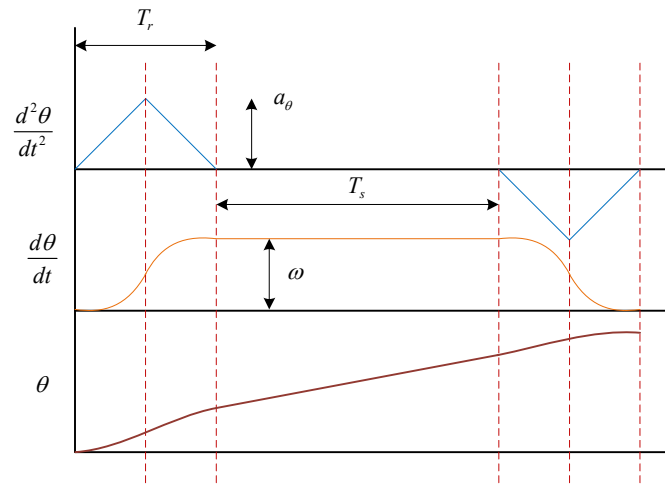


Figure 45 Angular acceleration and angular velocity should be constrained in order to limit the torque required and rotational speed used

³⁰ u25930_rotation_trajectory.m.

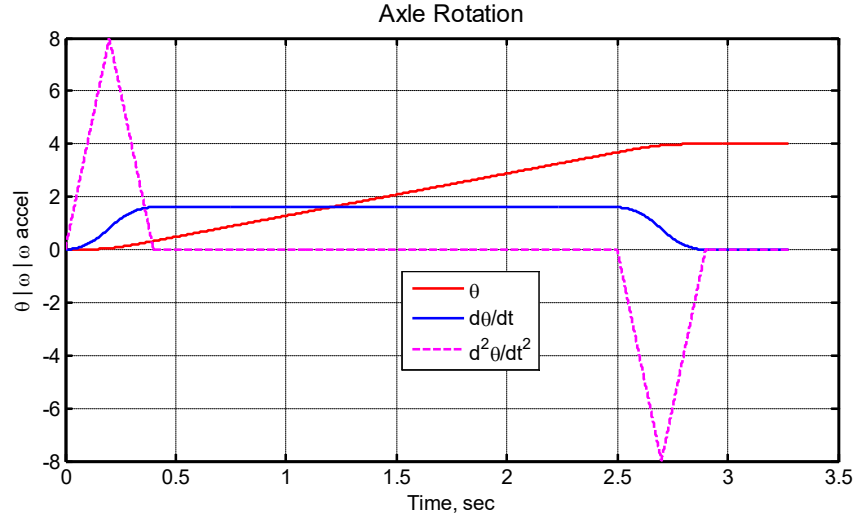


Figure 46 Computed example using the concepts of Figure 45. Maximum radian acceleration limited to 8 rad/sec² and maximum radian velocity limited to 1.6 rad/sec. Time required to traverse 4 radian change is about 2.5 sec.

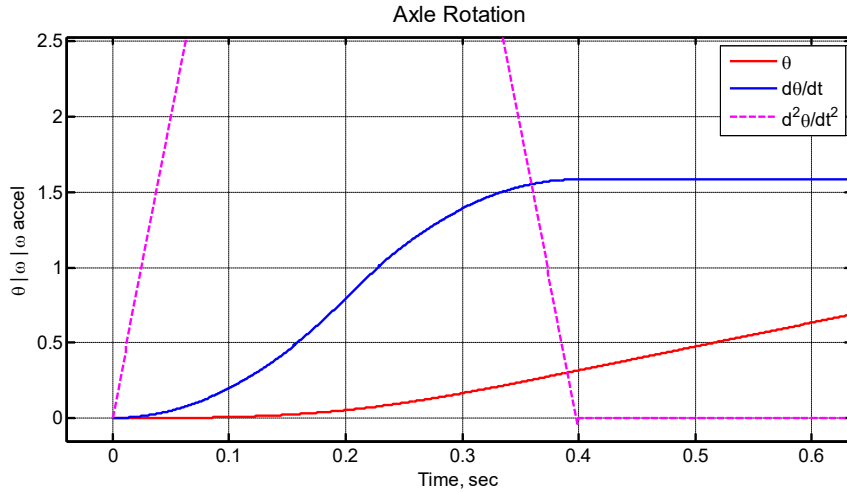


Figure 47 Close-up of Figure 46

More concisely in the context of Figure 45 and Figure 46, denote the maximum angular acceleration as $a_{\theta \max}$. Assuming that the angular rotation is in the positive sense, denote the maximum allowed angular velocity as $\omega_{\max}$. From Figure 45,

$$T_r = \frac{2\omega_{\max}}{a_{\theta \max}} \quad (26)$$

For simplicity, the same value of T_r can be used for all cases. Denote the maximum possible phase change impressed during the rotational acceleration as θ_{pmax} where

$$\theta_{pmax} = \frac{1}{2} \omega_{\max} T_r \quad (27)$$

Denote the commanded rotational phase change by θ_{cmd} .

$$\begin{aligned}
 & \text{if } (\theta_{cmd} > 2\omega_{max} T_r) \\
 & \quad T_s = \frac{\theta_{cmd} - 2\theta_{pmax}}{\omega_{max}} = \frac{\theta_{cmd} - \omega_{max} T_r}{\omega_{max}} = \frac{\theta_{cmd}}{\omega_{max}} - T_r \\
 & \text{else} \\
 & \quad T_s = 0 \\
 & \quad \theta_p = a_\theta \frac{T_r^2}{4} \\
 & \quad a_\theta = \frac{2\theta_{cmd}}{T_r^2} \\
 & \text{end}
 \end{aligned} \tag{28}$$

where θ_{cmd} is the desired rotation angle change in radians. From the previous two figures, in the case where $T_s \equiv 0$,

$$a(t) = \begin{cases} a_\theta & 0 < t \leq \frac{T_r}{2} \\ -a_\theta & \frac{T_r}{2} < t \leq \frac{3}{2}T_r \\ a_\theta & \frac{3}{2}T_r < t \leq 2T_r \end{cases} \tag{29}$$

and otherwise zero. In the case where $T_s \neq 0$,

$$a(t) = \begin{cases} a_\theta & 0 < t \leq \frac{T_r}{2} \\ -a_\theta & \frac{T_r}{2} < t \leq T_r \\ -a_\theta & T_r + T_s < t \leq T_s + \frac{3}{2}T_r \\ a_\theta & T_s + \frac{3}{2}T_r < t \leq T_s + 2T_r \end{cases} \tag{30}$$

but otherwise equal to zero. Given $a(t)$, the radian velocity and angle are given by

$$\begin{aligned}
 \omega(t) &= \int_0^t a(u) du \\
 \theta(t) &= \int_0^t \omega(u) du
 \end{aligned} \tag{31}$$

10.3 Additional Smoothing

The phase trajectory shown in Figure 46 and Figure 47 is very smooth as desired, but it can be made even more smooth by replacing the triangular acceleration pulses shown in Figure 46 with raised-cosine pulses as shown in Figure 48. In this approach, the triangular acceleration pulse is replaced by a raised-cosine pulse where the angular acceleration is given by

$$a(t) = a_{\theta} \frac{1 - \cos(\gamma t)}{2} \quad \text{for } 0 \leq t \leq T_r \quad (32)$$

and the maximum angular acceleration is given by a_{θ} .

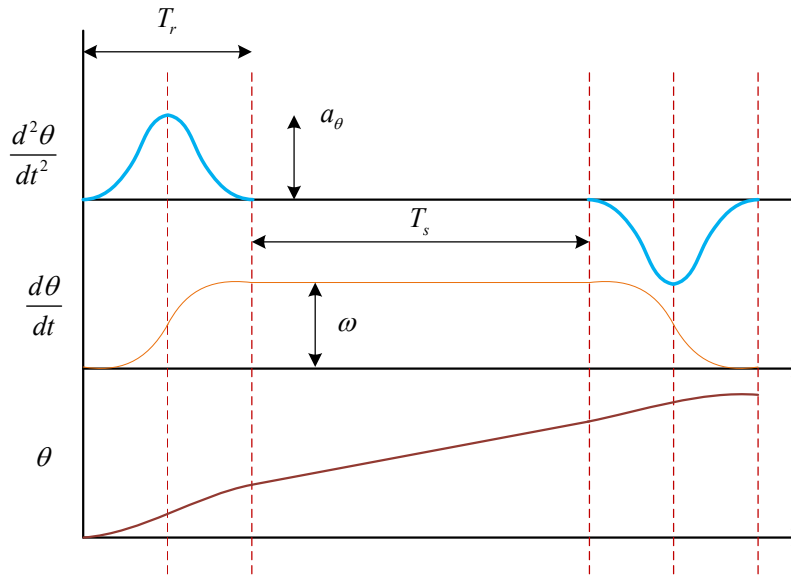


Figure 48 Smooth control waveform making use of raised-cosine acceleration pulses³¹

The radian frequency for $0 \leq t \leq T_r$ is the integral of (32) and given by

$$\omega(t) = \frac{a_{\theta}}{2} \left[t - \frac{\sin(\gamma t)}{\gamma} \right] \quad \text{for } 0 \leq t \leq T_r \quad (33)$$

with

$$\omega_{\max} = \frac{a_{\theta} T_r}{2} \quad (34)$$

This result makes it possible to rewrite (33) in a more convenient form as

$$\omega(t) = \frac{\omega_{\max}}{T_r} \left[t - \frac{\sin(\gamma t)}{\gamma} \right] \quad \text{for } 0 \leq t \leq T_r \quad (35)$$

The instantaneous phase during the ramping interval follows as given by

³¹ U28405 Photogrammetry Part 7 Figures.vsd.

$$\begin{aligned}
\theta(t) &= \frac{\omega_{\max}}{T_r} \left[\frac{t^2}{2} - \frac{\cos(\gamma t)}{\gamma^2} + \frac{1}{\gamma^2} \right] \quad \text{for } 0 \leq t \leq T_r \\
&= \frac{\omega_{\max}}{2T_r} \left[t^2 - \left(\frac{2}{\gamma} \right)^2 \sin^2 \left(\frac{\gamma t}{2} \right) \right] \quad \text{for } 0 \leq t \leq T_r \\
&= \frac{\omega_{\max}}{2T_r} \left\{ t^2 - \left[\frac{2}{\gamma} \sin \left(\frac{\gamma t}{2} \right) \right]^2 \right\} \quad \text{for } 0 \leq t \leq T_r
\end{aligned} \tag{36}$$

In this context,

$$\gamma = \frac{2\pi}{T_r} \tag{37}$$

and

$$\theta_{pulse} = \theta(T_r) = \frac{\omega_{\max} T_r}{2} \tag{38}$$

In order to make use of these results, assume that the pointing direction is to be changed by $\Delta\theta$ (assumed to be positive).

If $\Delta\theta \leq 2\theta_{pulse}$, T_s in Figure 48 can be made zero by choosing

$$\omega = \frac{\Delta\theta}{T_r} \leq \omega_{\max} \tag{39}$$

and using this value for $\omega_{\max}$ in the previous equations. If, on the other hand, $\Delta\theta > 2\theta_{pulse}$, use (38) along with

$$T_s = \frac{\Delta\theta - 2\theta_{pulse}}{\omega_{\max}} \tag{40}$$

An example start-up segment using these equations is shown in Figure 49 where the acceleration interval $T_r = 3$ seconds and the maximum rotational rate is limited to 36° per second which is quite fast! Adopting $T_r = 5$ seconds with a maximum rotational rate of 15° per second is shown in

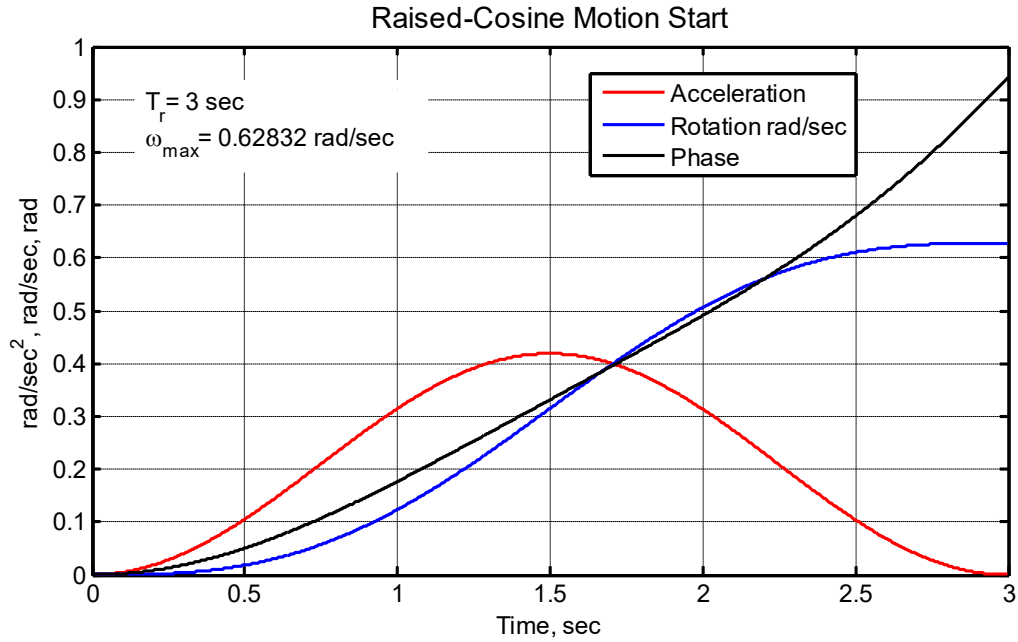


Figure 49 Start-up motion using raised-cosine trajectory shaping³²

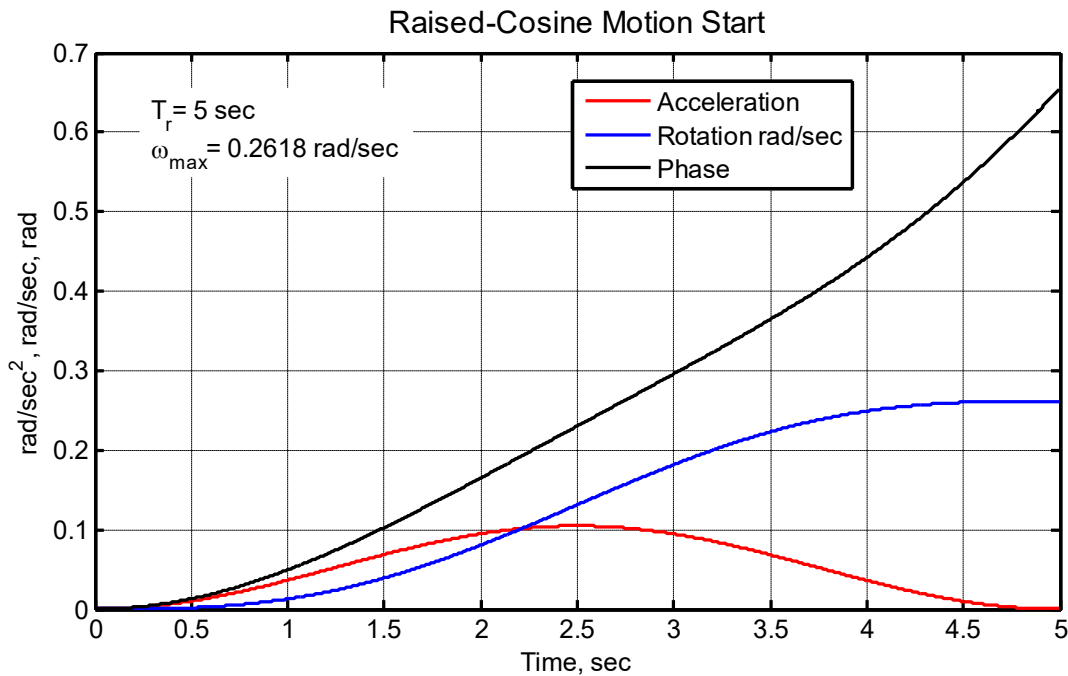


Figure 50 Start-up motion using raised-cosine trajectory shaping³³ with lower acceleration than in Figure 49

³² u25930_rotation_trajectory.m

³³ u25930_rotation_trajectory.m

11 Appendix: Hand-Controller MATLAB Script

```

===== u29443_handcontroller.m =====
%
%
%   J.A. Crawford
%   9 August 2024
%
%   Tailor the output voltage from the hand-controller to result
%   in different time constants for input changes which would further
%   increase the abs( angle ) versus return to the no-slewing condition
%
tauSlow= 1;
tauFast= 0.25;

npts= 8192;
dt= 20/npts;
tm= (0:npts-1)*dt;

nn= 1:100;
handcontroller(nn)= 0;

nn= 101:500;
handcontroller(nn)= 0.5*(nn-101)/(500-101);

nn= 501:1000;
handcontroller(nn)= 1;

nn= 1001:2000;
handcontroller(nn)= 1;

nn= 2001:3000;
handcontroller(nn)= 0;

nn= 3001:4000;
handcontroller(nn)= -0.5; % -0.5*(nn-3001)/(4000-3001);

nn= 4001:6000;
handcontroller(nn)= -0.5;

nn= 6001:7000;
handcontroller(nn)= 0.25;

nn= 7001:8192;
handcontroller(nn)= 0;

ttuse= zeros(1,npts);

fig1= figure(1);
clf;
axes( 'FontName', 'Arial', 'FontSize', 12 );
p1= plot( tm, handcontroller, 'r' );
set( p1, 'LineWidth', 2 );
grid on
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Time, sec', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Hand-Controller Input V', 'FontName', 'Arial', 'FontSize', 12 );
title( 'Hand-Controller Input Voltage vs Time', 'FontName', 'Arial', 'FontSize', 14 );

vinOld= 0;
vin= 0;

vout= zeros(1,npts);

cn1= 1 - 2*tauFast/dt;
cn2= 1 - 2*tauSlow/dt;
cp1= 1 + 2*tauFast/dt;
cp2= 1 + 2*tauSlow/dt;

tau2use= 1;

for ii=2:npts

    vin= handcontroller(ii);
    tau2use= floor( ( vout(ii-1)*vin < 0.001 ) + 1 );

    ttuse(ii)= tau2use;

    switch( tau2use )
        case 2
            vout(ii)= ( (vin+vinOld) - cn1*vout(ii-1) )/cp1;
        case 1
            vout(ii)= ( (vin+vinOld) - cn2*vout(ii-1) )/cp2;
    end
end

```

```

        vinOld= vin;
    end

    fig2= figure(2);
    clf;
    axes( 'FontName', 'Arial', 'FontSize', 12 );
    pl= plot( tm, vout, 'r' );
    set( pl, 'LineWidth', 2 );
    grid on
    xlabel( 'Time, sec', 'FontName', 'Arial', 'FontSize', 12 );
    ylabel( 'Angular Slew Rate', 'FontName', 'Arial', 'FontSize', 12 );
    title( 'Smoothed & Tailored Radial Slew Rate', 'FontName', 'Arial', 'FontSize', 14 );

    txt1= strcat( ['\tau_f_a_s_t= ', num2str(tauFast), ' ', 'sec' ] );
    txt2= strcat( ['\tau_s_l_o_w= ', num2str(tauSlow), ' ', 'sec' ] );

    annotation(fig2,'textbox','String',{txt1, txt2},'FontSize',12,...
        'FontName','Arial',...
        'FitBoxToText','off',...
        'LineStyle','none',...
        'BackgroundColor',[1 1 1],...
        'Position',[0.5639 0.7286 0.2908 0.1548]);
    h= gca;
    set( h, 'LineWidth', 2 );

    fig3= figure(3);
    clf;
    axes( 'FontName', 'Arial', 'FontSize', 12 );
    pl= plot( tm, vout - handcontroller, 'r' );
    set( pl, 'LineWidth', 2 );
    grid on
    xlabel( 'Time, sec', 'FontName', 'Arial', 'FontSize', 12 );
    ylabel( 'Hand-Controller vs Output Error', 'FontName', 'Arial', 'FontSize', 12 );
    title( 'Hand-Controller Error vs Time', 'FontName', 'Arial', 'FontSize', 14 );

    annotation(fig3,'textbox','String',{txt1, txt2},'FontSize',12,...
        'FontName','Arial',...
        'FitBoxToText','off',...
        'LineStyle','none',...
        'BackgroundColor',[1 1 1],...
        'Position',[0.5639 0.7286 0.2908 0.1548]);

    h= gca;
    set( h, 'LineWidth', 2 );

    fig4= figure(4);
    clf;
    axes( 'FontName', 'Arial', 'FontSize', 12 );
    pl= plot( tm, ttuse, 'r' );
    set( pl, 'LineWidth', 2 );

    grid on
    xlabel( 'Time, sec', 'FontName', 'Arial', 'FontSize', 12 );
    ylabel( 'Use Fast or Use Slow \tau', 'FontName', 'Arial', 'FontSize', 12 );
    title( 'Controller \tau vs Time', 'FontName', 'Arial', 'FontSize', 14 );

    axis( [0, 20, 0.8, 2.2] );

    annotation(fig4,'textbox','String',{txt1, txt2},'FontSize',12,...
        'FontName','Arial',...
        'FitBoxToText','off',...
        'LineStyle','none',...
        'BackgroundColor',[1 1 1],...
        'Position',[0.5639 0.7286 0.2908 0.1548]);

    h= gca;
    set( h, 'LineWidth', 2 );

```

12 Appendix: Measuring EPWM Drain Currents with DRV8301

The GoTo mount uses two DRV8301 booster packs which are plugged into the TMS320F28379D digital signal processor. The relevant DRV8301 schematic for the EPWM voltage and current measurements are shown in Figure 51. The header pins associated with the TMS320F28379D are detailed in Table 6 (excerpt in Table 13) for the Dec axis, and in Table 7 (excerpt in Table 14) for the RA axis. ADCA, ADCB, and ADCC of the TMS320F28379D can be used to measure the EPWM output voltages and drain currents.

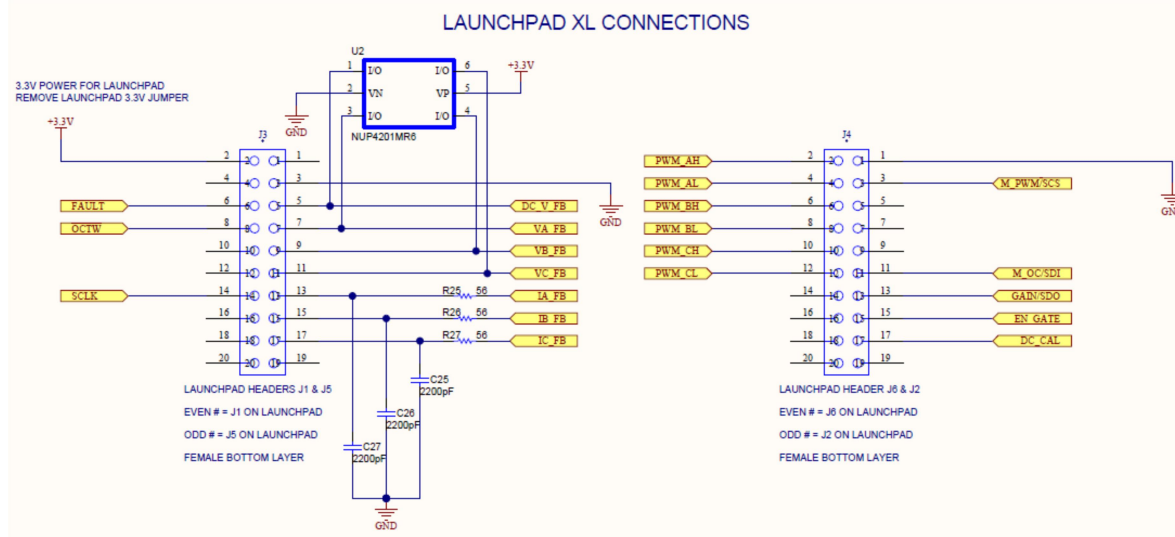


Figure 51 Schematic portion of DRV8301

Table 13 Excerpt from Table 6 for Elevation Axis EPWM-Related Measurements (EPWM1-3)

DRV8301 Signal	F28379D Pins				DSP GPIO	F28379D Signal	Comments
	J1	J3	J4	J2			
SCLK	7				60	GPIO60 / SPICLKA / SPISIMOB	SPI clock
DC-V-FB		23				ADCIN14 / CMPIN4P	
VA-FB		24				ADCINC3 / CMPIN6N	ADCC ADCIN3
VB-FB		25				ADCINB3 / CMPIN3N	ADCB ADCIN3
VC-FB		26				ADCINA3 / CMPIN1N	ADCA ADCIN3
IA-FB		27				ADCINC2 / CMPIN6P	ADCC ADCIN2
IB-FB		28				ADCINB2 / CMPIN3P	ADCB ADCIN2
IC-FB		29				ADCINA2 / CMPIN1P	ADCA ADCIN2
SCS				19	61	GPIO61 / SPISIMOB / SPISTEA*	SPI chip select
SDI				15	58	GPIO58 / SPISIMOA	Slave (DRV-8301) Input, DSP Output
SDO				14	59	GPIO59 / SPISOMIA	Slave (DRV-8301) Output, DSP Input
FAULT	3				19	GPIO19 / SCIRXDB / SPISTEA*	
OCTW	4				18	GPIO18 / SCITXDB / SPICLKA	Over-current, over-temperature, or both

Table 14 Excerpt from **Table 7** for RA Axis EPWM-Related Measurements (EPWM4-6)

DRV8301 Signal	F28379D Pins				DSP GPIO	F28379D Signal	Comments
	J5	J7	J8	J6			
SCLK	47				65	GPIO65 / SPICLKB	SPI clock
DC-V-FB		63				ADCIN15 / CMPIN4N	
VA-FB		64				ADCINC5 / CMPIN5N	ADCC ADCIN5
VB-FB		65				ADCINB5	ADCB ADCIN5
VC-FB		66				ADCINA5 / CMPIN2N	ADCA ADCIN5
IA-FB		67				ADCINC4 / CMPIN5P	ADCC ADCIN4
IB-FB		68				ADCINB4	ADCB ADCIN4
IC-FB		69				ADCINA4 / CMPIN2P	ADCA ADCIN4
SCS				59	66	GPIO66 / SPISTEB*	SPI chip select
SDI				55	63	SPISIMOB / GPIO63	Slave (DRV-8301) Input, DSP Output
SDO				54	64	SPISOMIB / GPIO64	Slave (DRV-8301) Output, DSP Input
FAULT	43				139	GPIO139 / SCIRXDC	
OCTW	44				56	GPIO56 / SCITXDC	Over-current, over-temperature, or both

CPU1 with EPWM4 – EPWM6 handles RA
 CPU2 with EPWM1 – EPWM3 handles Dec

Table 15 EPWM Drain Current Related ADC Setup Information

Current	ADC	CPU1 & RA Axis				CPU2 & Dec Axis			
		EPWM	SOC	ADC In	INT	EPWM	SOC	ADC In	INT
IA-FB	C	4	0	ADCIN4	INT_ADCC1	1	3	ADCIN2	INT_ADCC1
IB-FB	B	5	1	ADCIN4	INT_ADCB1	2	4	ADCIN2	INT_ADCB1
IC-FB	A	6	2	ADCIN4	INT_ADCA1	3	5	ADCIN2	INT_ADCA1

Table 16 PIE Channel Mapping³⁴**Table 3-2. PIE Channel Mapping**

	INTx.1	INTx.2	INTx.3	INTx.4	INTx.5	INTx.6	INTx.7	INTx.8	INTx.9	INTx.10	INTx.11	INTx.12	INTx.13	INTx.14	INTx.15	INTx.16
INT1.y	ADCA1	ADCB1	ADCC1	XINT1	XINT2	ADCD1	TIMER0	WAKE	-	-	-	-	IPC0	IPC1	IPC2	IPC3
INT2.y	EPWM1_TZ	EPWM2_TZ	EPWM3_TZ	EPWM4_TZ	EPWM5_TZ	EPWM6_TZ	EPWM7_TZ	EPWM8_TZ	EPWM9_TZ	EPWM10_TZ	EPWM11_TZ	EPWM12_TZ	-	-	-	-
INT3.y	EPWM1	EPWM2	EPWM3	EPWM4	EPWM5	EPWM6	EPWM7	EPWM8	EPWM9	EPWM10	EPWM11	EPWM12	-	-	-	-
INT4.y	ECAP1	ECAP2	ECAP3	ECAP4	ECAP5	ECAP6	-	-	-	-	-	-	-	-	-	-
INT5.y	EQEP1	EQEP2	EQEP3	-	-	-	-	-	SD1	SD2	-	-	-	-	-	-
INT6.y	SPIA_RX	SPIA_TX	SPIB_RX	SPIB_TX	MCBSPA_RX	MCBSPA_TX	MCBSPB_RX	MCBSPB_TX	SPIC_RX	SPIC_TX	-	-	-	-	-	-
INT7.y	DMA_CH1	DMA_CH2	DMA_CH3	DMA_CH4	DMA_CH5	DMA_CH6	-	-	-	-	-	-	-	-	-	-
INT8.y	I2CA	I2CA_FIFO	I2CB	I2CB_FIFO	SCIC_RX	SCIC_TX	SCID_RX	SCID_TX	-	-	-	-	-	-	UPPA (CPU1 only)	-
INT9.y	SCIA_RX	SCIA_TX	SCIB_RX	SCIB_TX	CANA_0	CANA_1	CANB_0	CANB_1	-	-	-	-	-	-	USBA (CPU1 only)	-
INT10.y	ADCA_EVT	ADCA2	ADCA3	ADCA4	ADCB_EVT	ADCB2	ADCB3	ADCB4	ADCC_EVT	ADCC2	ADCC3	ADCC4	ADCD_EVT	ADCD2	ADCD3	ADCD4
INT11.y	CLA1_1	CLA1_2	CLA1_3	CLA1_4	CLA1_5	CLA1_6	CLA1_7	CLA1_8	-	-	-	-	-	-	-	-
INT12.y	XINT3	XINT4	XINT5	-	-	VCU	FPU_OVER_FLOW	FPU_UNDER_FLOW	EMIF_ERROR	RAM_CORRECTABLE_ERROR	FLASH_CORRECTABLE_ERROR	RAM_ACCESS_VIOLATION	SYS_PLL_SLIP	AUX_PLL_SLIP	CLA OVER_FLOW	CLA UNDER_FLOW
Note: Cells marked "-" are Reserved																

³⁴ §3.4.5 of U26369

12.1 Matching Telescope Optics to CCD/CMOS Camera Sensor³⁵

Two approximations can be used to compute how well a candidate telescope and CCD sensor are matched. The total field of view for a given telescope main objective is given by³⁶

$$FoV_{\text{arc-minutes}} = \frac{3438 L_{\text{ccd-mm}}}{L_{\text{obj-mm}}} \quad (41)$$

where

$$\begin{array}{ll} L_{\text{ccd-mm}} & \text{CCD dimension of interest in mm} \\ L_{\text{obj-mm}} & \text{Focal length of the telescope's objective in mm} \end{array}$$

The *FoV* calculation is an important factor for initially capturing the airplane in view. As a point of reference, the moon extends an angle of about 0.5 degrees of arc (or equivalently 30 arc-minutes).

The second approximation is for the resolving power per CCD pixel given by

$$R_{\text{pixel-arcsec}} = \frac{206265 \times L_{\text{pixel-dim-mm}}}{L_{\text{obj-mm}}} \quad (42)$$

where

$$\begin{array}{ll} L_{\text{pixel-dim-mm}} & \text{Pixel dimension of interest in mm} \\ L_{\text{obj-mm}} & \text{Focal length of the telescope's objective in mm} \end{array}$$

For the high-quality Raspberry Pi camera having the details given in Figure 52,

	High Quality Camera
Sensor	Sony IMX477
Sensor resolution	4056 × 3040 pixels (12.33MP)
Image size	Diagonal 7.857 mm (Type 1/2.3)
Pixel size	1.55 × 1.55 μm
Image modes	4056 × 3040 2028 × 1520 2028 × 1080 1012 × 760

Figure 52 Raspberry Pi high-quality CCD camera³⁷

³⁵ From §2.1 of [1].

³⁶ From U15516, *My Notes*, Vol. 14.

³⁷ From U27676

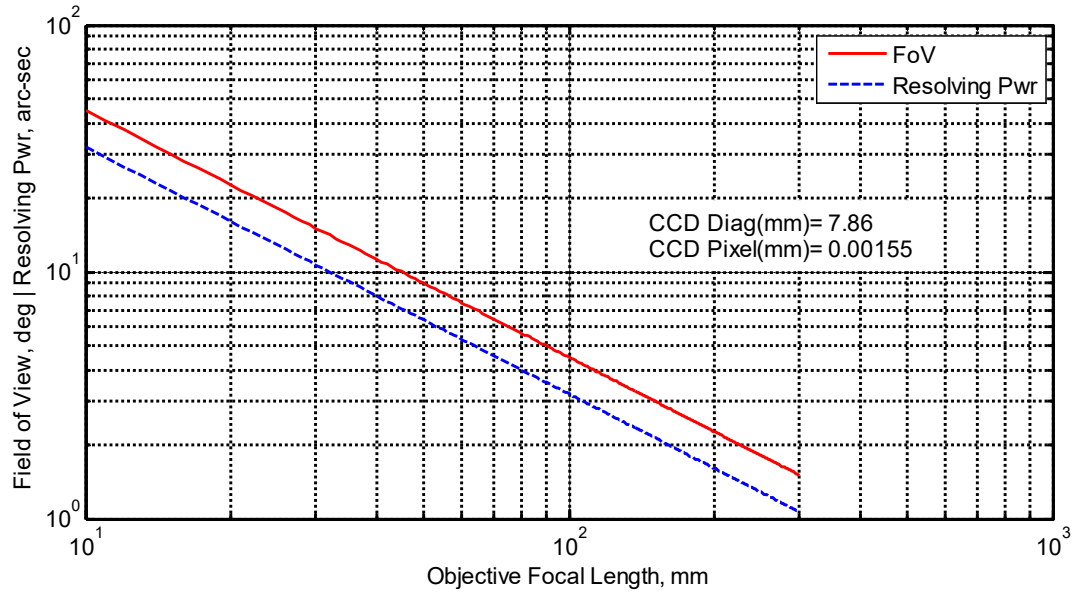


Figure 53 Field of view ($^\circ$) and resolving power (arc-sec) for the Raspberry Pi CCD sensor versus objective lens focal length³⁸. Large-sky viewing requires a short focal length (e.g., 0.5") whereas resolution of 2 arc-seconds requires a focal length on the order of 6.3" minimum. The diffraction limit posed by (51) dictates a minimum objective lens diameter of about 2.25" which further implies an $f\#$ of about $6.3/2.25 = 2.8$.

12.1.1 Back-Up Material

A simple lens arrangement is shown in Figure 54 where the distant object of height h_1 is located along the optical axis at distance d_1 with the image formed with height h_2 at distance d_2 . The f -number for the lens is given by

$$f\# = \frac{f_o}{D} \quad (43)$$

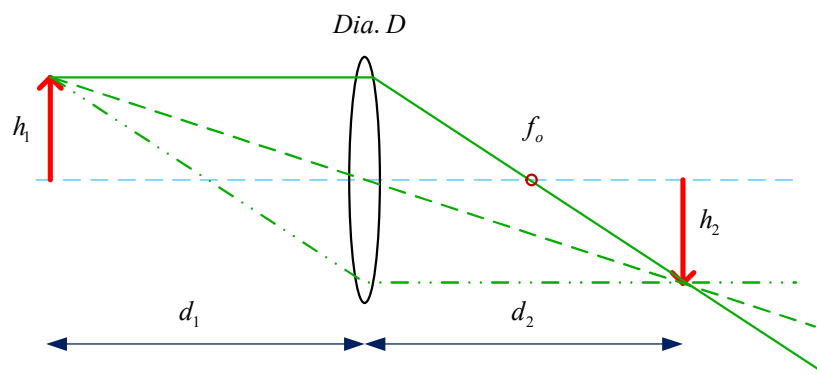


Figure 54 Simple lens behavior³⁹

The simple lens equation is given by

³⁸ u29637_telescope_ccd_match.m

³⁹ From U29655_Optics.vsd

$$\frac{1}{f_o} = \frac{1}{d_1} + \frac{1}{d_2} \quad (44)$$

Solving for d_2 produces

$$d_2 = \frac{f_o d_1}{d_1 - f_o} = \frac{f_o}{1 - \frac{f_o}{d_1}} \cong f_o \left(1 + \frac{f_o}{d_1} \right) \quad (45)$$

The amount of image magnification is given by

$$M = \frac{d_2}{d_1} \quad (46)$$

The similar triangles formed by the dashed ray through the center of the lens in Figure 54 permits the equality $h_2 / h_1 = d_2 / d_1$ to be made from which (46) can be rewritten as

$$M = \frac{d_2}{d_1} = \frac{f_o}{d_1 - f_o} \quad (47)$$

It is straight forward to show that the image height h_2 is given by

$$h_2 = \frac{f_o h_1}{d_1 - f_o} \quad (48)$$

When $f_o \ll d_1$, the image height is given by

$$h_2 = \frac{f_o h_1}{d_1 - f_o} 3.048 \times 10^5 \mu m \quad (49)$$

when f_o, h_1, d_1 are given in feet.

The diffraction limit on angular resolution is given by

$$\delta_{arc-sec} = \frac{206265 \lambda}{D} arc-sec \quad (50)$$

when the diameter of the objective lens D and the light wavelength of interest are given in mm. In the case where D is given in inches and the wavelength corresponds to green light ($\lambda = 5.55 nm$), (50) becomes

$$\delta_{arc-sec} = \frac{4.5}{D_{in}} arc-sec \quad (51)$$