


*THREE PHASE  
SPACE  
VECTOR PWM  
GENERATOR*

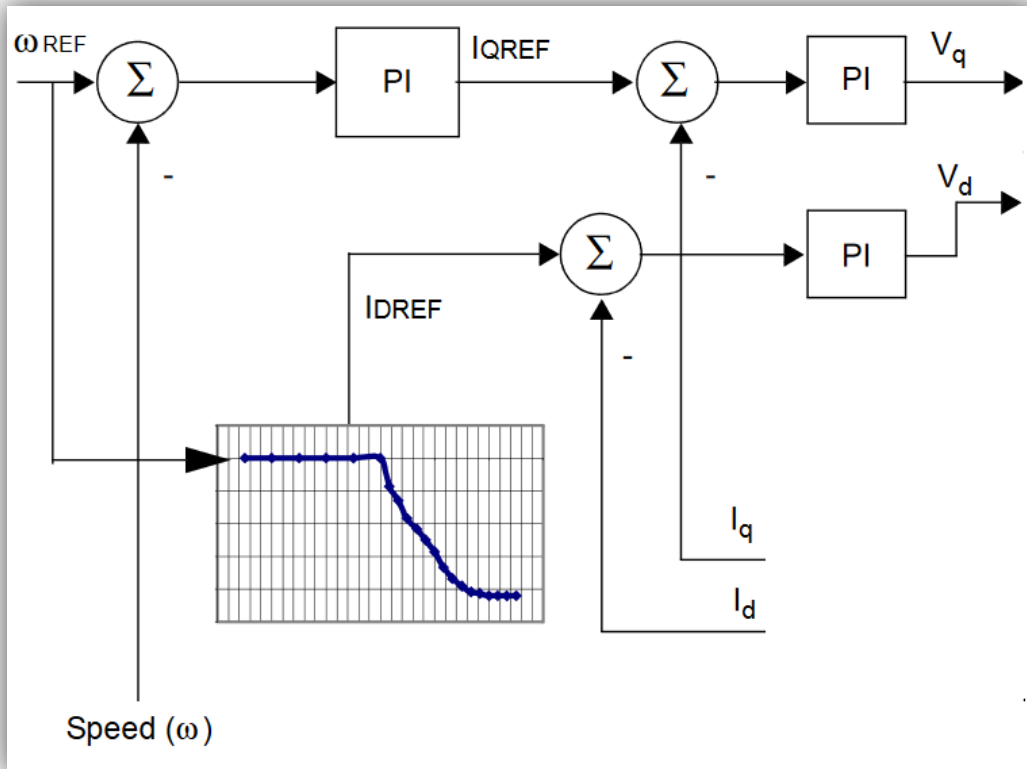
*Team King of the Hills: Val Wold, Armando Solis, Hai Huang*

# *Introduction*

---

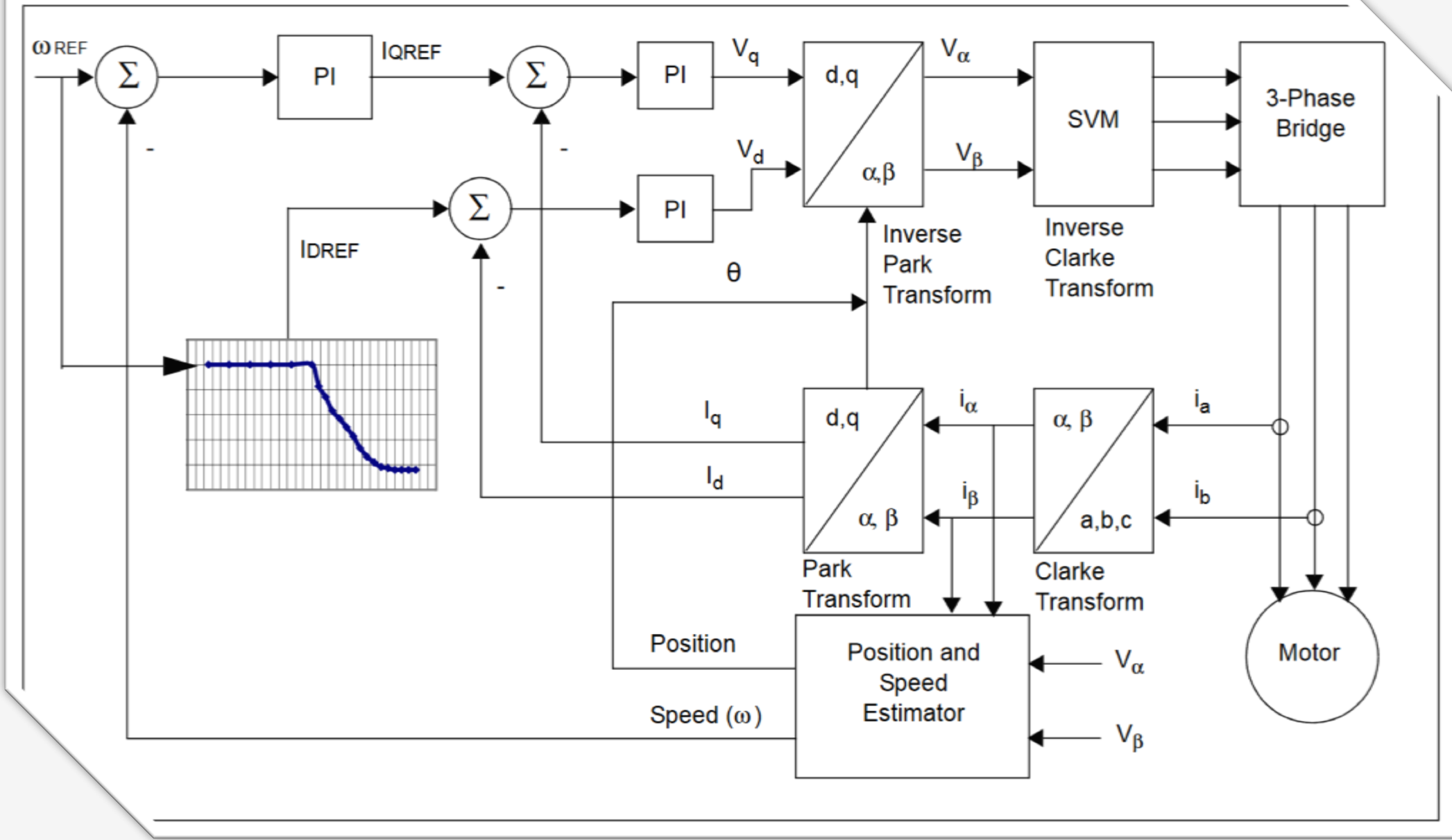
- No more alternators, no more EVs
  - Still King of the Hills
- 

# *FOC in two minutes*



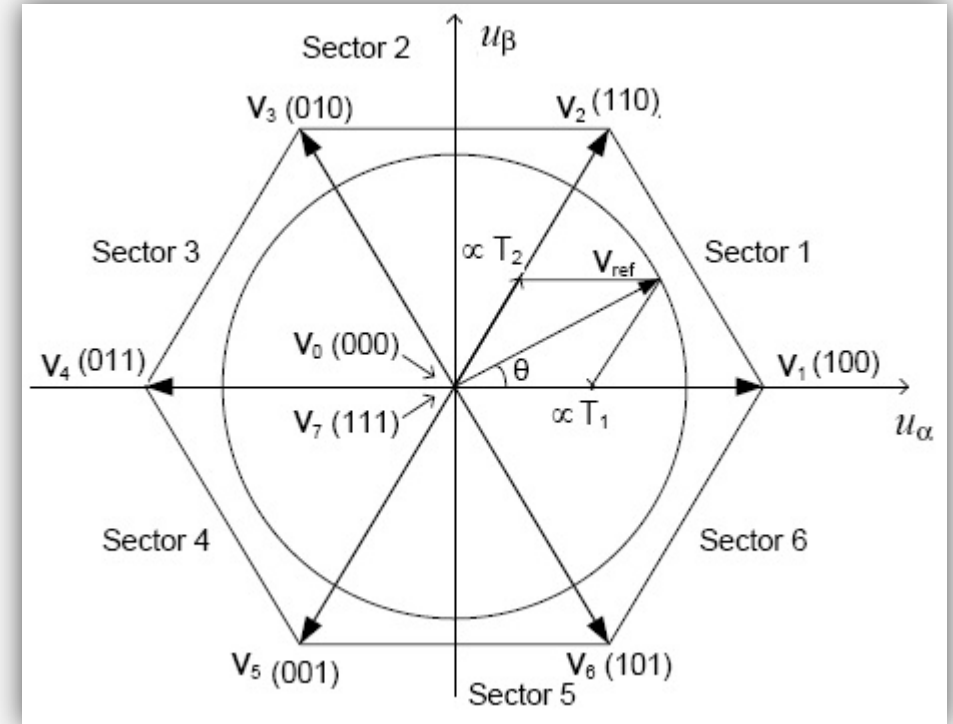
- Control is actually quite simple
- Objective: make rotor field orthogonal to stator field
- AC machines are quite nonlinear if controlled in a stationary, three phase, reference frame
- Some reference-frame shifting can yield not only steady state DC values but the dynamics are actually decoupled.
  - Linear algebra and trigonometry is used in the loop to calculate the decoupled and linearized dynamics of a machine.
  - Controlled parameters: torque, flux

**FIGURE 6: VECTOR CONTROL BLOCK DIAGRAM**



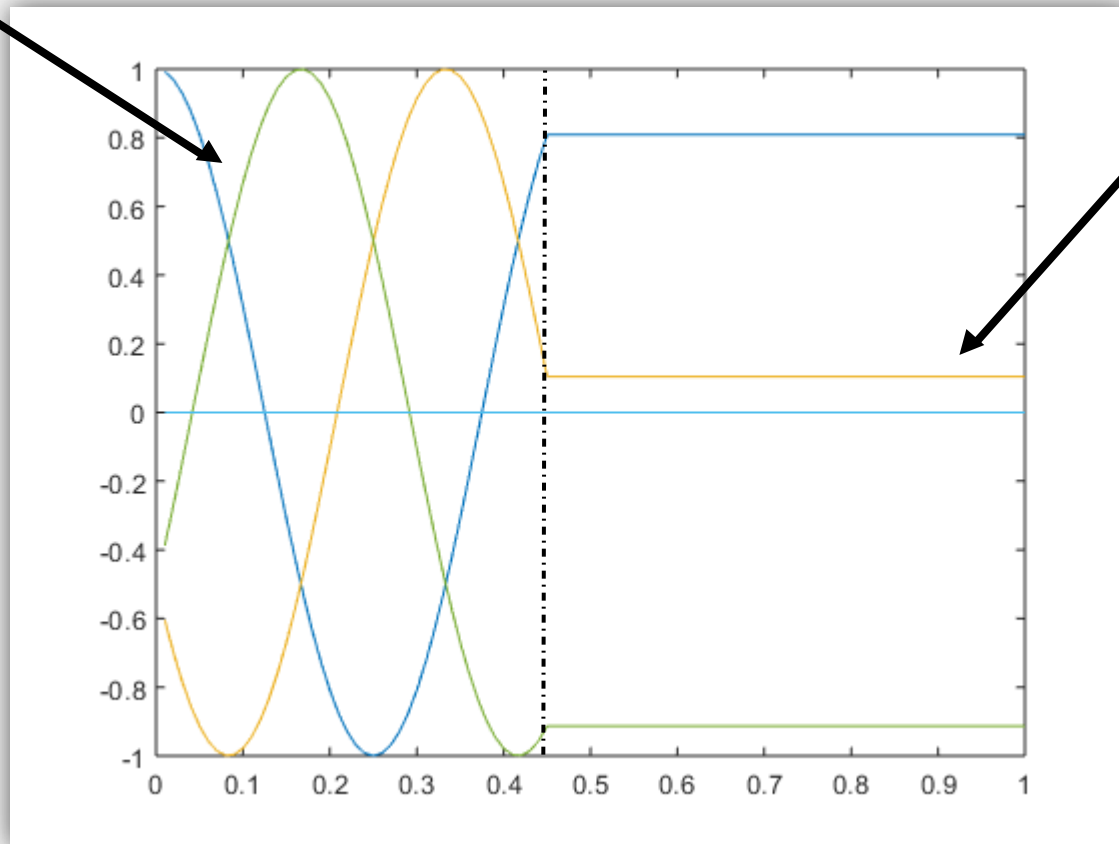
# Space Vector Theory

- Not phasors
  - Phasors: Describe a single sinusoidal output, with the same frequency as the reference, but with variable magnitude and phase
  - Space Vectors: Describes three sinusoidal output, all with the same phase relation, but with variable magnitude and position
- Switching states
  - Each possible combination of the switching states is a unit vector in voltage space (for a voltage regulator) or current space (for a current regulator)
  - Using switching times to weight these states allows for precise control over the position of the three-phase signal

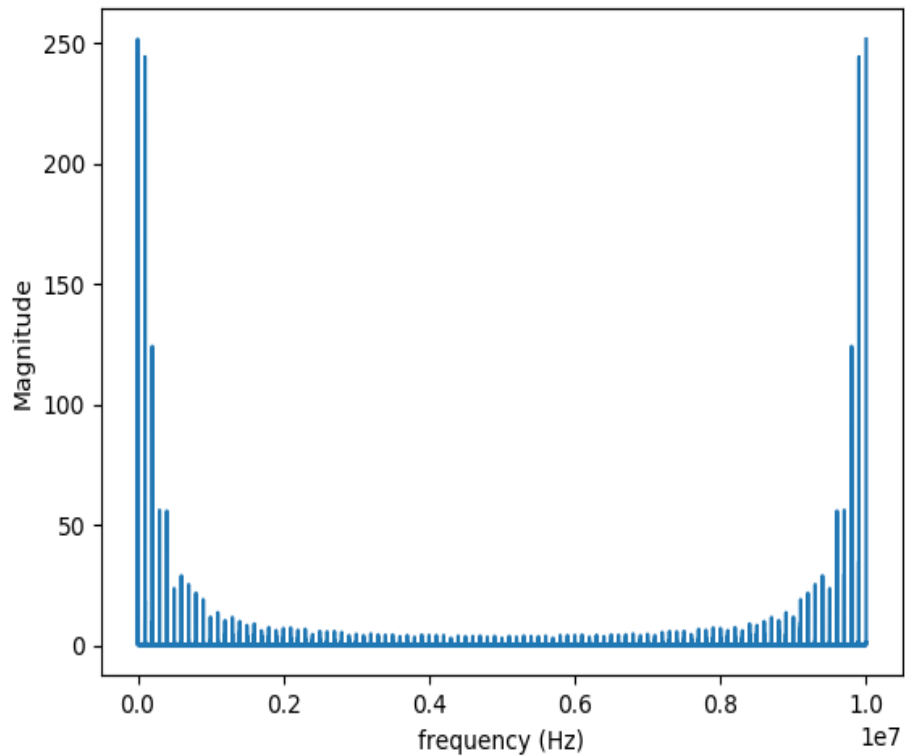


Rotating Space Vector

Stationary Space Vector



# *Simulation*



- Phases B and C should be shifted by -120 degrees and 120 degrees, respectively
- PWM is a form of frequency modulation, thus the fft of the PWM signal should resemble FM spectral characteristics
- If these yield the correct results, we can conclude that the SVM algorithm generates correct switching times

# Implementation in c

- Scheduling, fastest to slowest:
    - PWM counter ( $freq_{sw} = \frac{80MHz}{counts\ per\ duty\ cycle}$ )
    - Space Vector PWM updater ( $\frac{fundamental}{resolution}$ )
      - $resolution = \frac{smallest\ step\ in\ degrees}{360}$
      - Example: 50Hz fundamental (1500 RPM), 3 degree smallest step, 6 kHz updates
    - Frequency updater
      - Change the fundamental frequency
      - Example: 50 Hz to 60 Hz using a push-button
-





# *Implementation in c*

- Fixed point multiplication
  - Example: divide 50 by pi
  - First, express  $\pi^{-1}$  as a multiple of  $2^{-11}$ 
    - $\frac{1}{\pi} \cong 652 * 2^{-11}$
    - $\frac{50}{\pi} \cong 50 * 652 * 2^{-11} = (50 * 652) \gg 11 = 15.918$
    - Actual answer:  $\frac{50}{\pi} = 15.9155$
    - Dividing by powers of 2 is simply a right shift
  - Maximizing resolution
    - 16-bit architecture
    - Q-number format (Q integer bits.fractional bits)
      - Moving the point to where as many fractional bits are assigned as possible but no more

# Testing

- Values will be based off the results from the Python SVM Simulation

## Steps:

- Input the desired angle
- Python carries out the calculations
- Output is a 3x3 array which tells us what switches are on, and the switching times

```
Space Vectors: [[1, 2, 3], [2, 3, 4], [4, 6, 2]]  
Switching Times: [461, 70, 468]
```

67 degrees

```
Space Vectors: [[2, 3, 4], [3, 4, 5], [1, 3, 5]]  
Switching Times: [331, 243, 424]
```

145 degrees



# Verification

6  
7  
D  
e  
g  
r  
e  
e  
s

Name	Type	Address	Value
<input checked="" type="checkbox"/> PDC1	SFR	0xC26	0x0000
<input checked="" type="checkbox"/> PDC2	SFR	0xC46	0x0000
<input checked="" type="checkbox"/> PDC3	SFR	0xC66	0x0000
<input checked="" type="checkbox"/> theta_var	int	0x1060	0x0000
<input checked="" type="checkbox"/> integrator_flag	int	0x1062	0x0000
<Enter new watch>			
<input checked="" type="checkbox"/> PERIOD	int (PSV)	0x36C	0x03E8
<input checked="" type="checkbox"/> states	unsigned int[3][3]	0x1794	
<input checked="" type="checkbox"/> t1	int	0x178A	0x01CD
<input checked="" type="checkbox"/> t2	int	0x1790	0x0046
<input checked="" type="checkbox"/> tz	int	0x1792	0x01D5

1  
4  
5  
D  
e  
g  
r  
e  
e  
s

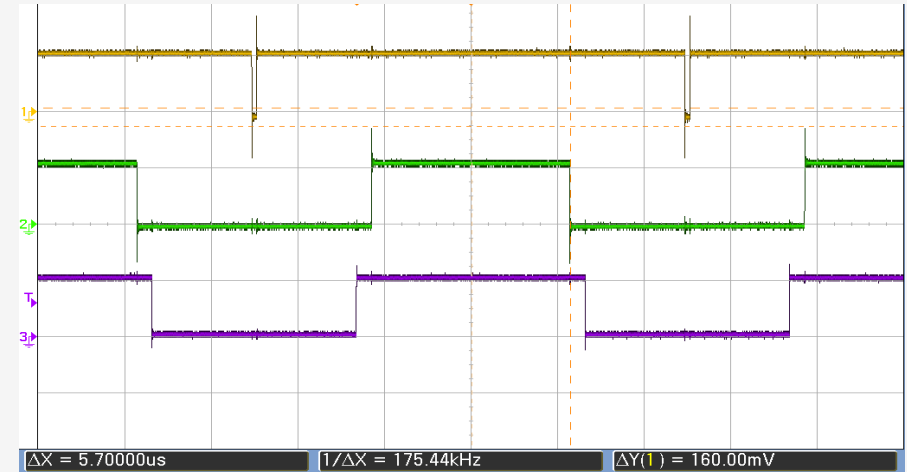
Name	Type	Address	Value
<input checked="" type="checkbox"/> PDC1	SFR	0xC26	0x0000
<input checked="" type="checkbox"/> PDC2	SFR	0xC46	0x0000
<input checked="" type="checkbox"/> PDC3	SFR	0xC66	0x0000
<input checked="" type="checkbox"/> theta_var	int	0x1060	0x0000
<input checked="" type="checkbox"/> integrator_flag	int	0x1062	0x0000
<Enter new watch>			
<input checked="" type="checkbox"/> PERIOD	int (PSV)	0x36C	0x03E8
<input checked="" type="checkbox"/> states	unsigned int[3][3]	0x1794	
<input checked="" type="checkbox"/> t1	int	0x178A	0x014B
<input checked="" type="checkbox"/> t2	int	0x1790	0x00F4
<input checked="" type="checkbox"/> tz	int	0x1792	0x01A9

# Verification

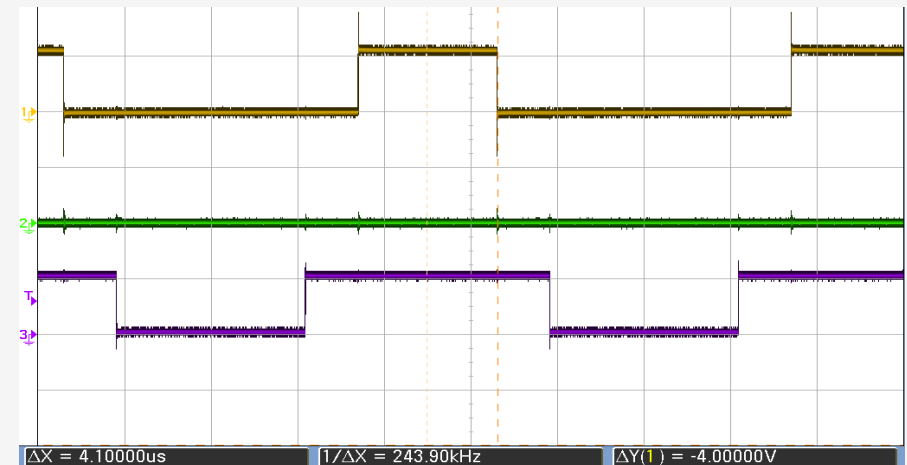
## Steps:

- Insert desired angle, run code
- Measure switching times
- Convert to counts

$$\text{Counts} = T_{us} * \left(\frac{1}{40 * 10^3}\right)(1000)$$



67 degrees



145 degrees

# Verification

67 Degrees	T1	T2	TZ
Python	461	70	468
MPLAB	461	70	469
Oscilloscope	456	72	472

145 Degrees	T1	T2	TZ
Python	331	243	424
MPLAB	331	244	425
Oscilloscope	328	244	428

# *Conclusion*

---

- Big picture: Field oriented control
  - Little picture: Space vector implementation on 16-bit mcu-dsp hybrid
  - Simulation of algorithm in python yields correct frequency-domain characteristics
  - C implementation: scheduling and computation
  - Verification
- 