Developments on Control Board and DSP software

1. DC bus voltage measurement from each module
2. Phase current measurement from each module
3. Position and speed measurement from encoder
4. Motor speed control
5. Phase current control for each module
6. Balancing of DC bus voltages
7. Waveform synthesizing with PWM for low harmonic content
8. DC bus overvoltage protection
9. Phase overcurrent protection
10. Short circuit protection (?)
11. Advanced techniques (fault detection, operation under fault etc.)

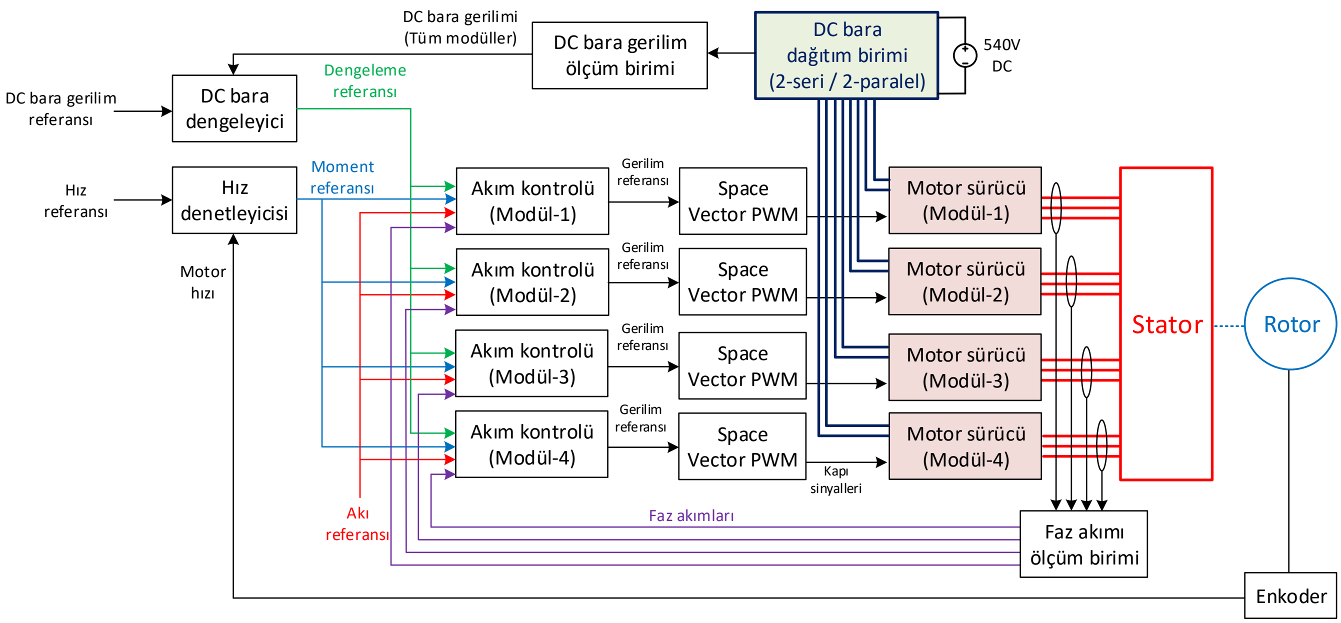


Fig. 1. General control block diagram

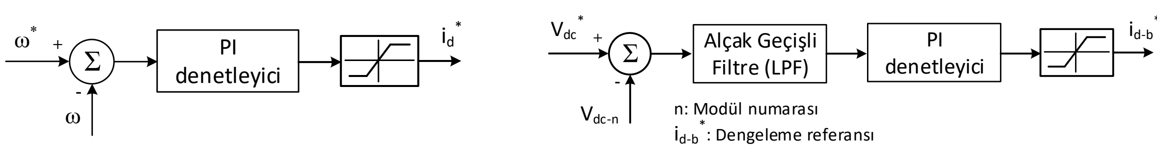


Fig. 2. Speed and voltage controllers

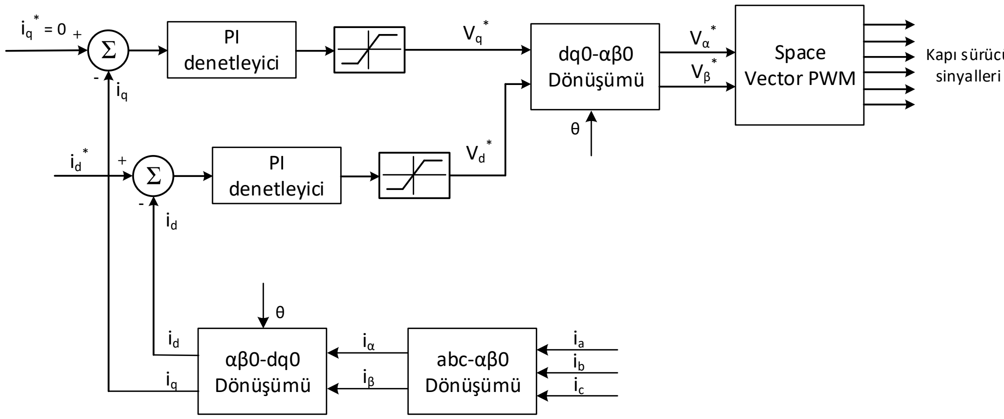


Fig. 3. Current controller

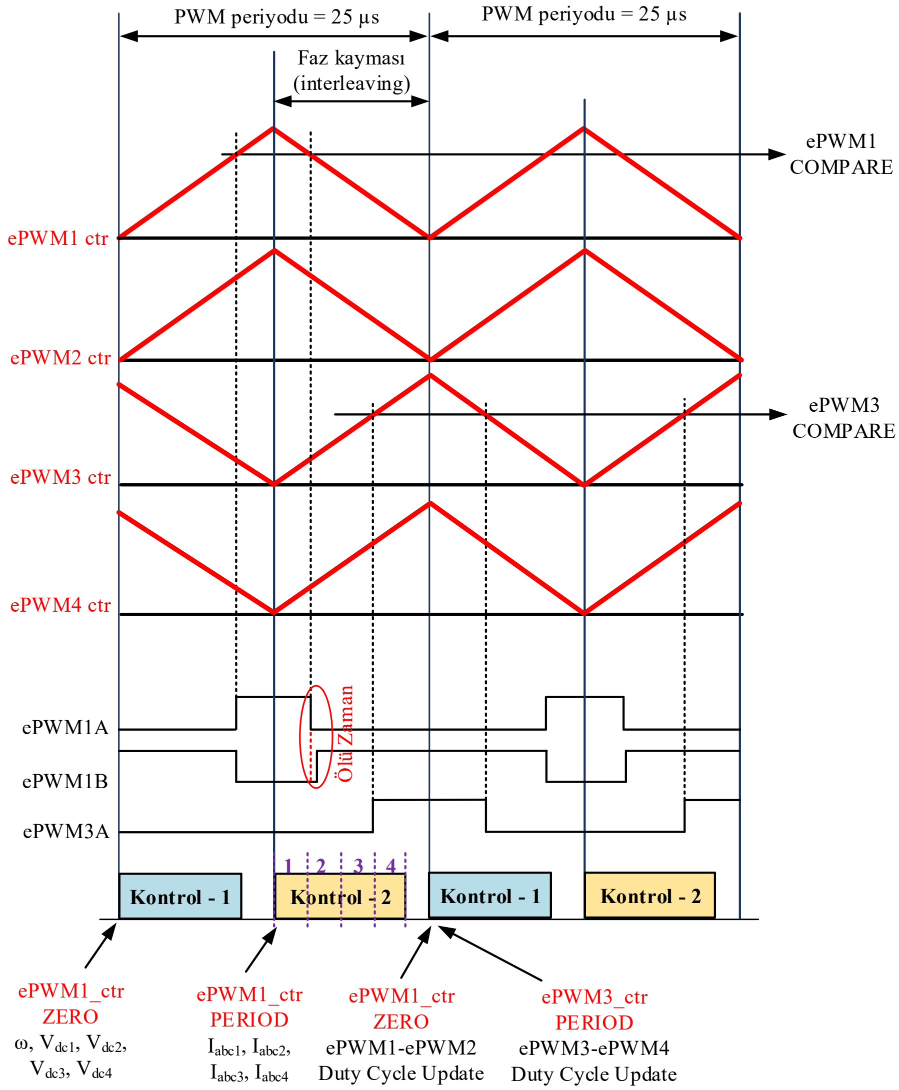
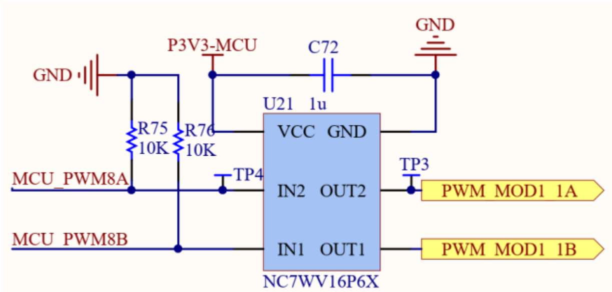
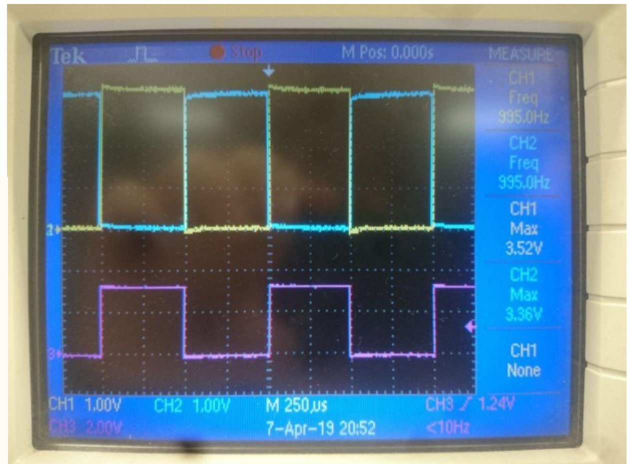


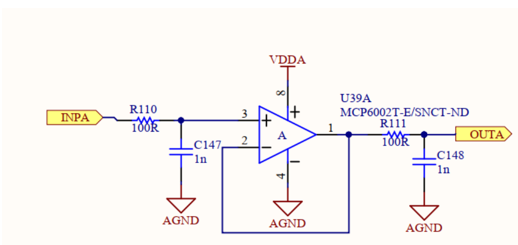
Fig. 4. Control system timing

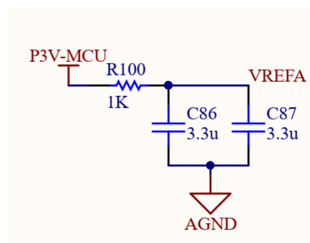
PWM generation:

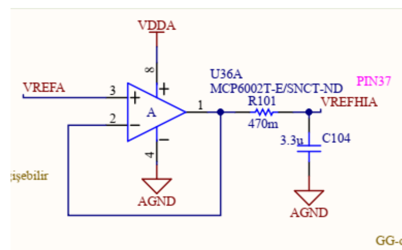




Measurement with ADC:







Control system timing:

There are two settings to configure for each PLL – a multiplier and a divider. They obey the formulas:

$f_{PLLSYSCLK} = f_{OSCCLK}$ * (SYSPLLMULT.IMULT + SYSPLLMULT.FMULT) / SYSCLKDIVSEL.PLLSYSCLKDIV

Source	Frequency	Description
INTOSC2	10 MHz	Default clock source
INTOSC1	10 MHz	Set as clock source if missing clock is detected at power up or right after device reset

The base ADC clock is provided directly by the system clock (SYSCLK). This clock is used to generate the ADC acquisition window. The register ADCCTL2 has a PRESCALE field which determines the ADCCLK. The ADCCLK is used to clock the converter.

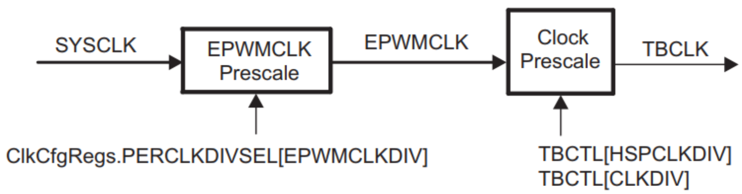
Table 5-44. ADC Operating Conditions (12-Bit Single-Ended Mode)

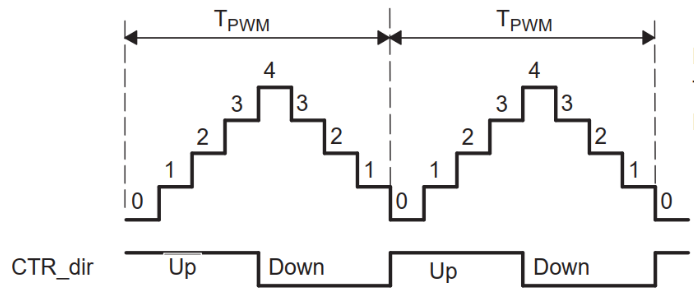
over recommended operating conditions (unless otherwise noted)

	MIN	TYP	MAX	UNIT
ADCCLK (derived from PERx.SYSCLK)	5		50	MHz
Sample window duration (set by ACQPS and PERx.SYSCLK)(1)	75			ns
V_{REFHI}	2.4	2.5 or 3.0	V_{DDA}	V

Assuming a 100ns sample window is desired with a SYSCLK frequency of 200MHz, then the acquisition window duration should be 100ns/5ns = 20 SYSCLK cycles. The ACQPS field should therefore be set to 20 - 1 = 19.

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;      //SOC5 will convert ADCINA1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 19;     //SOC5 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;   //SOC5 will begin conversion on ePWM3 SOCB
```



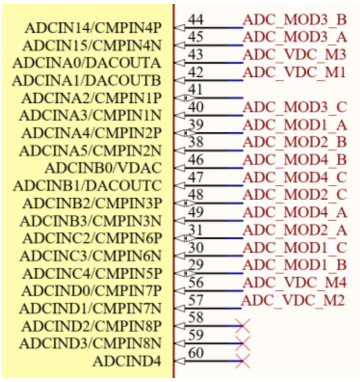


For Up and Down Count

$T_{PWM} = 2 \times TBPRD \times T_{TBCLK}$

$F_{PWM} = 1 / (T_{PWM})$

ADC results

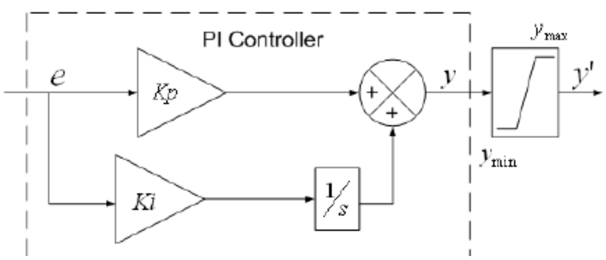



```
float Voltage_TransferFunction = 114.406;
float Current_TransferFunction = 13.333;
float Current_Offset = 1.5;
```

```
// First digital readings from ADCs
Vdc_M3_adc     = AdcaResultRegs.ADCRESULT0;
Vdc_M1_adc     = AdcaResultRegs.ADCRESULT1;
Is_M3_PhC_adc = AdcaResultRegs.ADCRESULT3;
Is_M1_PhA_adc = AdcaResultRegs.ADCRESULT4;
Is_M2_PhB_adc = AdcaResultRegs.ADCRESULT5;
Is_M3_PhB_adc = AdcaResultRegs.ADCRESULT14;
Is_M3_PhA_adc = AdcaResultRegs.ADCRESULT15;
Is_M4_PhB_adc = AdcbResultRegs.ADCRESULT0;
Is_M4_PhC_adc = AdcbResultRegs.ADCRESULT1;
Is_M2_PhC_adc = AdcbResultRegs.ADCRESULT2;
Is_M4_PhA_adc = AdcbResultRegs.ADCRESULT3;
Is_M2_PhA_adc = AdccResultRegs.ADCRESULT2;
Is_M1_PhC_adc = AdccResultRegs.ADCRESULT3;
Is_M1_PhB_adc = AdccResultRegs.ADCRESULT4;
Vdc_M4_adc     = AdcdResultRegs.ADCRESULT0;
Vdc_M2_adc     = AdcdResultRegs.ADCRESULT1;
DSP_Temp_Sensor_adc = AdcaResultRegs.ADCRESULT13;
DSP_Temp_Sensor = GetTemperatureC(DSP_Temp_Sensor_adc);
```

```
// Calculate actual measurements
Vdc_M1 = (Vdc_M1_adc * max_adc_analog / max_adc_digital) * Voltage_TransferFunction;
Vdc_M2 = (Vdc_M2_adc * max_adc_analog / max_adc_digital) * Voltage_TransferFunction;
Vdc_M3 = (Vdc_M3_adc * max_adc_analog / max_adc_digital) * Voltage_TransferFunction;
Vdc_M4 = (Vdc_M4_adc * max_adc_analog / max_adc_digital) * Voltage_TransferFunction;
Is_M1_PhA = ((Is_M1_PhA_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M1_PhB = ((Is_M1_PhB_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M1_PhC = ((Is_M1_PhC_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M2_PhA = ((Is_M2_PhA_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M2_PhB = ((Is_M2_PhB_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M2_PhC = ((Is_M2_PhC_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M3_PhA = ((Is_M3_PhA_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M3_PhB = ((Is_M3_PhB_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M3_PhC = ((Is_M3_PhC_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M4_PhA = ((Is_M4_PhA_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M4_PhB = ((Is_M4_PhB_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
Is_M4_PhC = ((Is_M4_PhC_adc * max_adc_analog / max_adc_digital) - Current_Offset) * Current_TransferFunction;
```

PI controller:



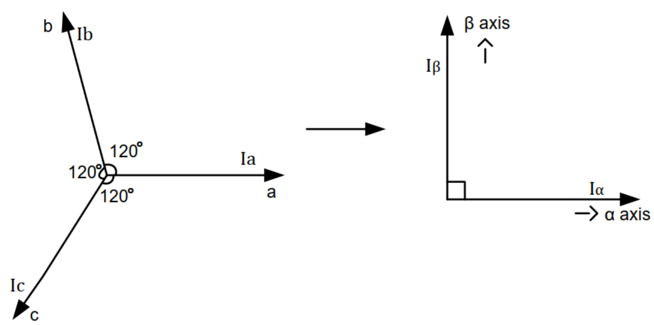
$$G(z) = \frac{Y(z)}{E(z)} = K_p + \frac{K_i}{1 - z^{-1}}$$

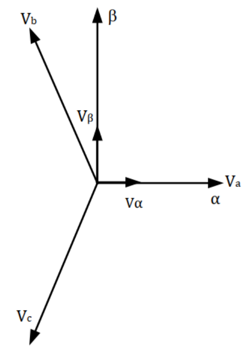
$$Y(z) = Y(z)z^{-1} + E(z)(K_p + K_i) - E(z)z^{-1}K_p$$

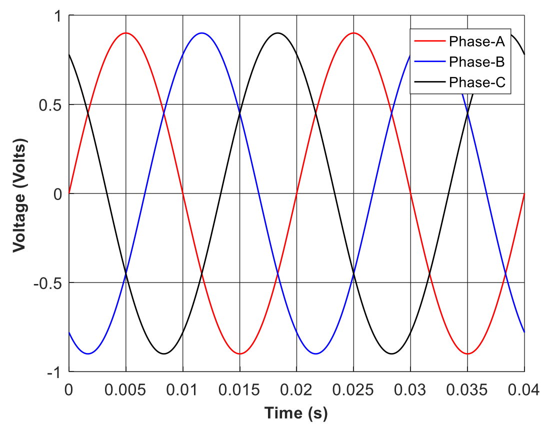
$$y[n] = y[n-1] + e[n]\big(K_p + K_i\big) - e[n-1]K_p$$

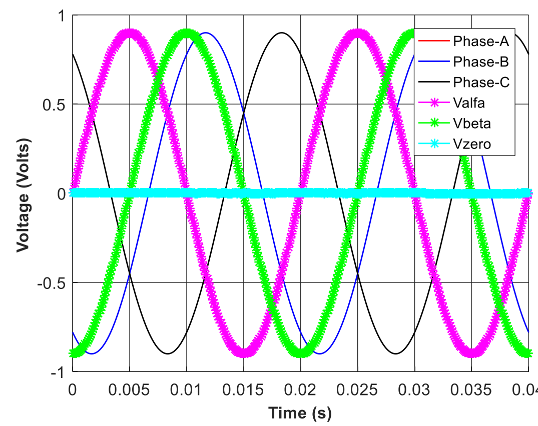
Transformations:

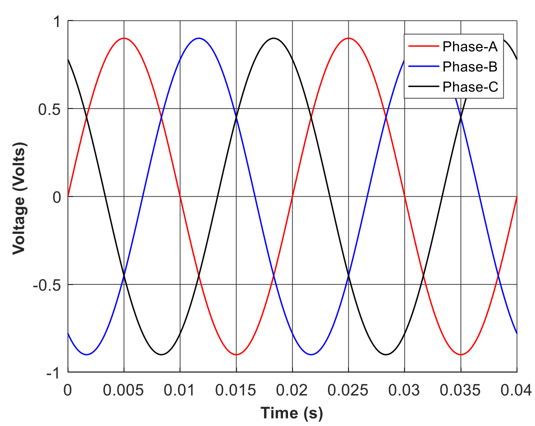
Clarke and Inverse Clark

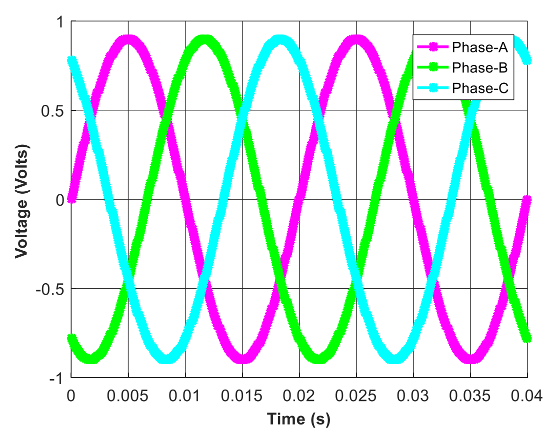












Park and Inverse Park