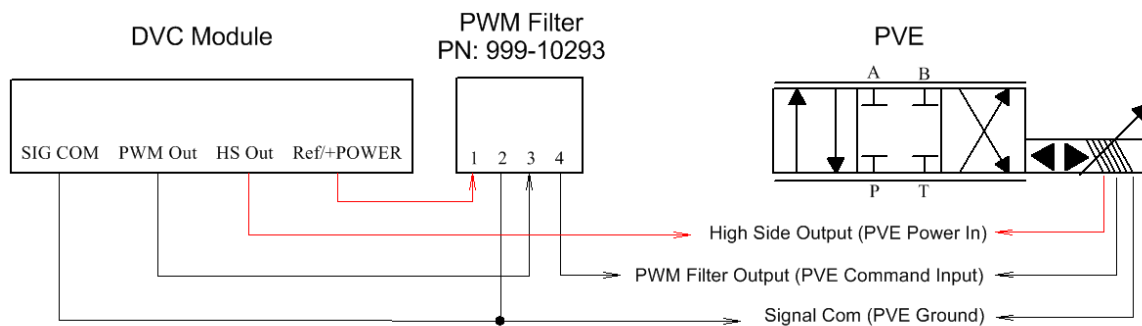


Introduction

The 999-10293, PWM Filter may be used with a DVC710, DVC707, DVC750 or similar DVC proportional module to convert a sinking PWM output to a sourcing DC voltage output that has a range of 0 Volts to its input supply voltage. It is typically used to drive PVE type valves but may be used as a 0V to 5V output by simply using the DVC's Reference output voltage as the power input to the PWM Filter. The PWM Filter will supply up to 4mA on a 24 volt system and 2ma on a 12 volt system.

Simplified Wiring for PVE Valve



Programming the DVC to Drive the PVE Valve

The following points should be considered when programming the DVC to drive the filter output.

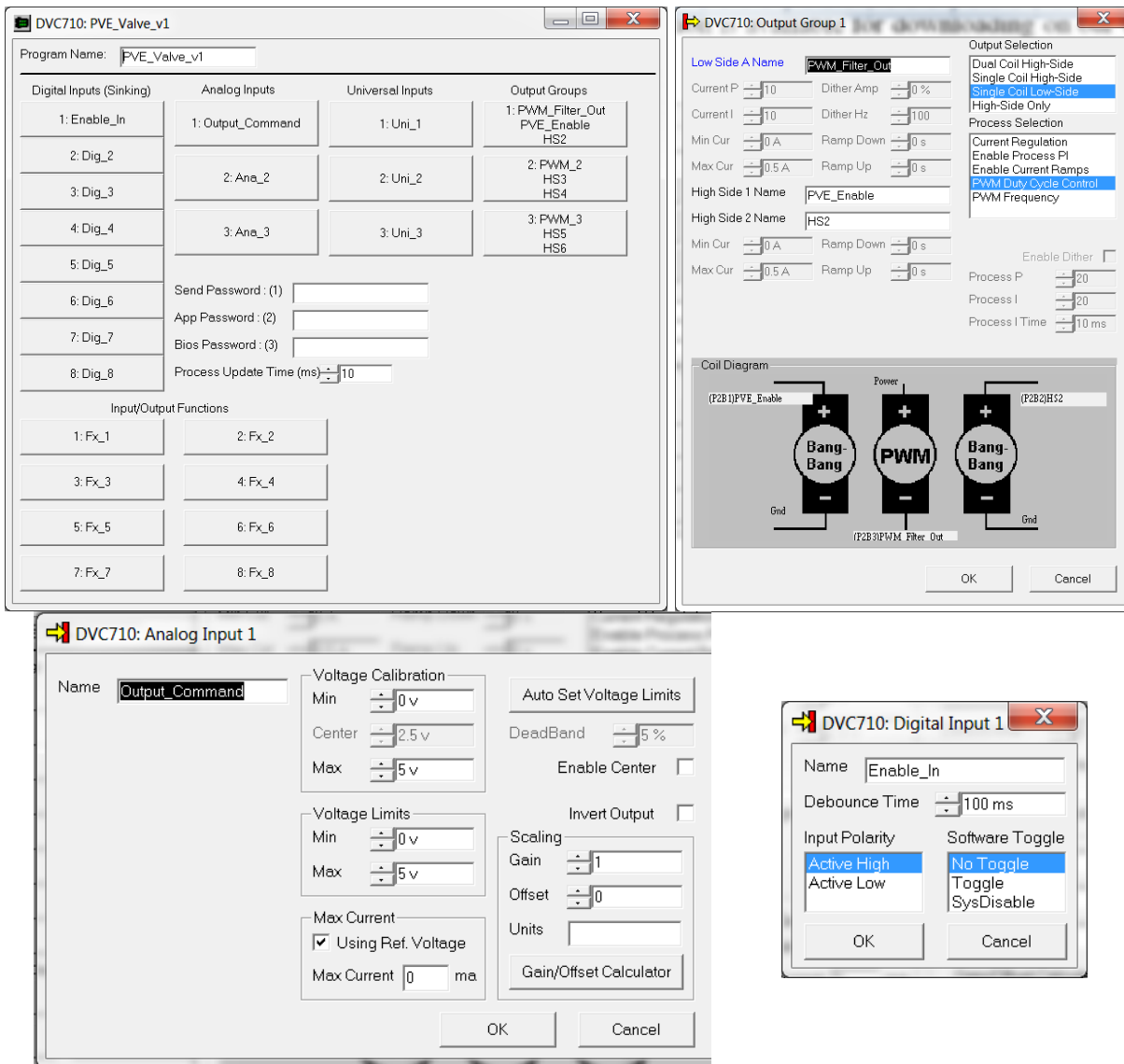
1. When commanding a device such as a PVE type valve using the PWM Filter, the output command to the device will be inversely proportional to the command to the PWM output. Therefore you must invert commands to the output in order to obtain standard directional outputs from the cylinder, motor, etc.
2. To prevent unexpected mechanical operation when initializing a system, enable and set the PWM output to a neutral setting (typically 50%) before enabling (applying power to) the PVE Valve with the HS output.
3. Run the PWM output group in Single Coil High Side, PWM Duty Cycle Mode.
4. When using a High Side Output to provide power to a PVE valve, set the variable *HSname.opendisable* to true to prevent false open detection on the High Side Output.

Sample DVC Code

This code example includes all considerations listed above as well as a Ramp feature that may be adjusted through EEMEM. Valid settings for the EEMEM variable, Ramp_Scaler are, 0 – 100. The program will automatically clamp this at 100. This corresponds to about 5 seconds per side or 10 seconds end to end.

The sample application is attached to this PDF file.

Module I/O Settings



The screenshots show the following configurations:

- DVC710: PVE_Valve_v1**: Program Name: PVE_Valve_v1. Digital Inputs (Sinking): 1: Enable_In, 2: Dig_2, 3: Dig_3, 4: Dig_4, 5: Dig_5, 6: Dig_6, 7: Dig_7, 8: Dig_8. Analog Inputs: 1: Output_Command, 2: Ana_2, 3: Ana_3. Universal Inputs: 1: Uni_1, 2: Uni_2, 3: Uni_3. Output Groups: 1: PWM_Filter_Out, PVE_Enable, HS2; 2: PWM_2, HS3, HS4; 3: PWM_3, HS5, HS6. Input/Output Functions: 1: Fx_1, 2: Fx_2, 3: Fx_3, 4: Fx_4, 5: Fx_5, 6: Fx_6, 7: Fx_7, 8: Fx_8. Send Password: (1), App Password: (2), Bios Password: (3). Process Update Time (ms): 10.
- DVC710: Output Group 1**: Low Side A Name: PWM_Filter_Out. Current P: 10, Dither Amp: 0%. Current I: 10, Dither Hz: 100. Min Cur: 0 A, Ramp Down: 0 s. Max Cur: 0.5 A, Ramp Up: 0 s. High Side 1 Name: PVE_Enable, High Side 2 Name: HS2. Min Cur: 0 A, Ramp Down: 0 s. Max Cur: 0.5 A, Ramp Up: 0 s. Enable Dither: . Process P: 20, Process I: 20, Process I Time: 10 ms. Coil Diagram: (P2B1)PVE_Enable (Bang-Bang), Power (P2B2)HS2 (Bang-Bang), (P2B3)PWM_Filter_Out (Bang-Bang).
- DVC710: Analog Input 1**: Name: Output_Command. Voltage Calibration: Min: 0 v, Center: 2.5 v, Max: 5 v. Auto Set Voltage Limits: . DeadBand: 5%. Enable Center: . Voltage Limits: Min: 0 v, Max: 5 v. Invert Output: . Scaling: Gain: 1, Offset: 0. Units: [blank]. Max Current: Using Ref. Voltage, Max Current: 0 ma. Gain/Offset Calculator:
- DVC710: Digital Input 1**: Name: Enable_In. Debounce Time: 100 ms. Input Polarity: Active High (selected), Active Low. Software Toggle: No Toggle (selected), Toggle, SysDisable.



Using the PWM Filter, 999-10293

Always Code

```
Output_Demand = 1023 - Output_Command
PWM_Filter_Out.enable = Enable_In
PVE_Enable.opendisable = 1

if (Enable_In = True) then
  if ((Output_Command < 460) OR (Output_Command > 563)) AND (Input_Ready = 0) then
    PWM_Filter_Out = 512
  else
    if (Output_Demand > PWM_Filter_Out) then
      if (Output_Demand > (PWM_Filter_Out + Ramp_Scaler)) then
        PWM_Filter_Out = PWM_Filter_Out + Ramp_Scaler
      else
        PWM_Filter_Out = Output_Demand
      end if

      if (PWM_Filter_Out > 1023) then
        PWM_Filter_Out = 1023
      end if
    else
      if (Output_Demand < (PWM_Filter_Out - Ramp_Scaler)) then
        if (PWM_Filter_Out > Ramp_Scaler) then
          PWM_Filter_Out = PWM_Filter_Out - Ramp_Scaler
        else
          if (PWM_Filter_Out > 0) then
            PWM_Filter_Out = PWM_Filter_Out - 1
          end if
        end if
      else
        PWM_Filter_Out = Output_Demand
      end if
    end if

    Input_Ready = 1

  end if
else
  Output_Demand = 512
  Input_Ready = 0
end if

PVE_Enable = Input_Ready

if (Ramp_Scaler > 100) then
  Ramp_Scaler = 100
  eecommand = eewrite
else
  eecommand = 0
end if

***** Program Variables *****
dim Ramp_Scaler as eemem

dim Input_Ready as uint
dim Output_Demand as uint
```