Pre-Loaded demonstration application code example for DVC 7 module

The file "025-00044_DVC7_default_ship_file_06_16_09.DVC" has been loaded into the dvc7 module. More information on the following topics can be found in the 'Programmer's User Guide' located on the HCT website. This file will give the programmer some examples of the type of programming syntax that is used with the DVC7.

The program allows the output group PWM to be controlled by the analog input with the assistance of an I/O function curve, i.e. analog 1 will control output group1 coil1HS1 through Input/ Output Function curve valve_shaper_1. Factory default gives a 1:1 ratio of input to output. This can be modified by using the Intella Program Loader Monitor (PLM). Start the PLM, select the DVC 7 Master indicator to open the window that displays all the DVC7 I/O. Near the center of the screen contains a button labeled I/O functions. Selecting this button will display the I/O function curves. They are selectable by a yellow pull down menu located in the upper left hand corner. Using the output section, the user can adjust the 1:1 ratio, i.e. the user may determine, that having 20% input should deliver 40% PWM output. This is adjustable using this function. Remember to select 'Send Changes' to make the change active. These changes are automatically stored so they are present on the next power up.

All analog inputs can be 'tuned' online. Select the Analog Inputs button. Variables that can be adjusted are Min, Max and Center Volts, Min and Max limits, deadband volts, and Lower and Upper Ramp Up and Ramp Down values. These variables are saved to EEmem variable locations. EEmem variables will be discussed later in this document.

All PWM outputs can be 'tuned' online. Select the Output Groups button. Variables that can be adjusted are 'A' coil min / max current, 'B' coil min / max current, Current P & I, 'A' and 'B' coil ramp up and down, and dither amp and Hz. These variables are saved to EEmem variable locations. EEmem variables will be discussed later in this document.

Virtual memory allows the programmer to monitor programmed variable status while program is executing online. An example of how to program virtual display variables is located in the 'Always' code section and the virtual display icon located in the Intella programming tool.

The programmer initially sets values to variables in the analog inputs as well as the Output groups, but these variables can be saved while performing 'online tuning'. All variables created in the program are volatile memory, except variables that are designated as EEmem variables. This means that when power is cycled all volatile memory variables are reset to 0. When the program is initially started, values from the EEmem locations are loaded into memory and executed as instructed by the program. An example of how to move volatile memory into EEmem locations is demonstrated in the 'dim_vars_here' bubble. 'dim_code' is where the variables of the program are defined. It should be noted to organize the variables according to their section or definition. This allows for a 'cleaner' program. '2' equates the data located in the PLM and EEmem variables to a variable located in the program. Note: give the variable a meaningful name, this will aid in troubleshooting. '3' sets a timer EEmem_update_timer for 5 seconds. At the end of the 5 seconds, the program pointer transfers into logic bubble '10'.

'10', a sample of the logic is seen below.

```
'**********Analog Input 1 logic**********
'min volts Analog 1
if (AI1_minVolts_cal_PLM_IV != Analog_In1.MinVolts) then
   AI1_minVolts_cal_PLM_IV = Analog_In1.MinVolts
   AI1_minVolts_cal_eem = Analog_In1.MinVolts
   state = 100
end if

if (AI1_minVolts_cal_EEM_IV != AI1_minVolts_cal_eem) then
   AI1_minVolts_cal_EEM_IV = AI1_minVolts_cal_eem
   Analog_In1.MinVolts = AI1_minVolts_cal_eem
   state = 101
end if
```

The first two lines are comments as they are began with a '. Comments should be added to programs to aid the programmer in troubleshooting and modification. The first 'if' statement compares the internal variable (IV), AI1_minVolts_cal_PLM_IV, with the PLM value Analog_In1.MinVolts. If these two variables are not equal, then the program goes to the next line where the value of the second variable, Analog_In1.MinVolts is copied to the first variable, AI1_minVolts_cal_PLM_IV. The program then goes to the next line and moves the value of the second variable, Analog_In1.MinVolts, into the first variable, AI1_minVolts_cal_eem. The program then goes to the next line and places the integer '100' into the variable 'state'. The contents of the variable 'state' could be monitored by the virtual display to tell the operator if the if-then statement was executed. If the variables in the 'if' statement were equal, the lines below it would not have been executed.

High Country Tek, Inc. 208 Gold Flat Court Nevada City, CA, 95959 Tel: (1) 530 265 3236 Fax: (1) 530 265 3275 www.hctcontrols.com

The two 'if' statements will look at the value of the variables 'AI1_minVolts_cal_PLM_IV != Analog_In1.MinVolts' and 'AI1_minVolts_cal_EEM_IV != AI1_minVolts_cal_eem' to make sure they are equal. If they are not equal, this piece of logic will assure that the latest values for the PLM are saved in EEmem variables, so they can be retained on the next power sequence.

The last command of bubble '10' is EEcommand = EEwrite. This command compares the value of the program memory EEmem and the actual EEmem. If these two values are not equal, the program copies the values from program memory EEmem to the actual EEmem.

Bubble '5' will initialize all values in the EEmem location to preset values. Variable ee_memory_status is defined in bubble '2'. It is a number that can be set to the date of modification, in this instance 08208, august 20, 2008. The value of this variable is checked in the transition to bubble '5'. If the variables don't match, a transition to bubble '5' is allowed. In bubble '5', 08158 is moved into ee_memory_status and the EEmem variables are initialized. Take care to assure ee_memory_status is updated in both areas of the code.

Enable_valve_1 contains the machine logic. Bubble '2' contains logic to enable the Output groups. It also contains the following logic.

```
'allows analog inputs to control the appropriate valve coils
'set input of graph to Analog input
valve_shaper_1.in = Analog_In1
valve_shaper_2.in = Analog_In2
valve_shaper_3.in = Analog_In2

' set valve to output of graph
Valvecoil1 = valve_shaper_1.out
Valvecoil2 = valve_shaper_2.out
Valvecoil3 = valve_shaper_3.out
```

The ' denotes a comment. Use as many comments as necessary to describe the programming function. There are three lines of code that take the value of the analog input and equate them to the I/O function curve input. Then the output of the I/O function curve is equated to the valve coil. Remember, the variable on the right hand side of the '=' is copied into the variable on the left hand side of the '='.

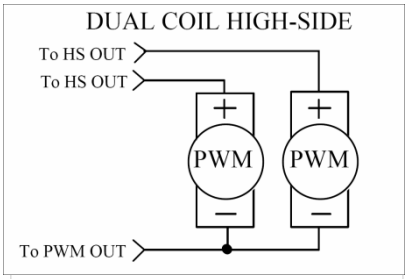
The outputs will not be driven until a eemem variable 'Module_Enable' is set to a '1'. This can be accomplished by using the PLM. Start the PLM, select the EE-MEMORY indicator to open the window that displays all the DVC7 eememory variables. The third variable is 'Module_Enable'. When the controller is initially loaded, the enable variable will have a '0'. Modify the value by placing a '1' in the window. Select the button 'Save Changes' in the lower left hand corner. These changes are automatically stored so they are present on the next power up. The outputs will now follow the corresponding analog input and the corresponding I/O curve.

An EEmem variable, 'ee_system_startup_delay_time', comes preset in the template code at 30. This value of 30 will cause a 30 second startup wait time from when the controller is first started, until the output of the valves are enabled. This time can be diminished by modifying the value in the EEmem section. This timer was added in bubble 2 of 'dim_vars_here' logic sequence to demonstrate how a timer can be constructed and executed.

NOTE:
The demonstration application code is pre-loaded into the DVC 7 unit and allows the user to apply power and ground, connect a simple potentiometer or one axis of a joystick to input 1, and connect a dual coil valve to output 1 to quickly control a bi-directional flow project giving speed and direction of an actuator or motor. The user may expand the project by connecting another potentiometer or second joystick axis to input 2 and a second dual valve to output 2 to realize fully functional open loop control of direction and speed for dual actuators or motors.

User Connection details for DVC7:

+Power In:-	**K1, K2 or K3**
Power GND:-	**B2, B3**
Input 1:-	**Analog Input #1 (D2), +5v Reference output (E3), Ground (B3)**
Input 2:-	**Analog Input #2 (E2), +5v Reference output (E3), Ground (B3)**
Output 1:-	**PWM output#1 (H1), forward direction coil (G1), reverse direction coil (J1)**
Output 2:-	**PWM output#2 (H2), forward direction coil (G2), reverse direction coil (J2)**



Connection detail for proportional outputs using H1/H2 and G1/G2 and J1/J2

High Country Tek, Inc. 208 Gold Flat Court Nevada City, CA, 95959 Tel: (1) 530 265 3236 Fax: (1) 530 265 3275 www.hctcontrols.com