Fluid*Mechatronix*TM

Process Automation System



Curriculum Sample

Unit 1 - General Overview	. 13
Chapter 1 - Overview of Fluid <i>Mechatronix</i> ™	
Section 1: Description of System	
Focus 1: Electrical Schematics	
Focus 2: System Schematic	. 23
Focus 3: System Parts	. 24
Focus 4: System Capabilities	
Section 2: Basic Pumping	. 27
Focus 1: Pumping	. 27
Focus 2: Cavitation	. 28
Focus 3: Pump Performance Curves	. 29
Section 3: Tank Controls	
Focus 1 - Primary Tank Controls	. 31
Focus 2 - Secondary Tank Controls	. 33
Focus 3 - Temperature Control	
Focus 4 - Fluid Level Control with Pumps and Valves	. 35
Focus 5 - Solenoid Valve Control	. 36
Focus 6 - Pump Control	. 37
Section 4: PIDE Controls	. 38
Focus 1 - Pressure PIDE Control	. 38
Focus 2 - Suction PIDE Control	. 39
Focus 3 - Flow PIDE Control	. 40
Chapter 2 - Introduction to Automation	. 41
Section 1: The History of Automation	. 42
Focus 1: Manual Control	. 42
Focus 2: Relays and Timers	. 43
Focus 3: Programmable Relays	. 44
Section 2: Current Automation Technology	. 45
Focus 1: Programmable Automation Controller	. 45
Focus 2: Communication Networks	
Chapter 3 - Introduction to PAC	. 47
Section 1: Central Processing Unit	. 48
Focus 1: Microprocessor	. 48
Focus 2: Advantages	. 48
Focus 3: Execute Instruction (Ladder Logic Basics)	
Section 2: Scan-Cycle in PAC Operations	. 50
Focus 1: Parts of the Scan-Cycle	. 50
Focus 2: Input Scan	. 50
Focus 3: Execute Programs	. 51
Focus 4: Output Scan	. 51
Chapter 4 - Introduction to HMI	. 53
Section 1 - History of the HMI	
Focus 1 - What is an HMI?	
Focus 2: How does an HMI Work?	. 55

Section 2: User Interface Software	
Focus 1: HMI Interface	56
Focus 2: HMI Applications	57
Focus 3: Software Advantages	58
Section 3: User Interface Controls	
Focus 1: Programmers	59
Focus 2: Operators	59
Chapter 5 - Introduction to VFD	
Section 1: VFD Basic Operation	61
Focus 1: Frequency	61
Focus 2: Voltage	61
Focus 3: Pulse Width Modulation (PWM)	61
Section 2: Variable Frequency Drive Benefits	63
Focus 1: Speed Control	63
Focus 2: Optimized Motor Starts	63
Focus 3: Constant Torque	
Chapter 6 - Introduction to Communications	65
Section 1: Serial Communication	
Focus 1: The History of RS-232	
Focus 2: RS-232 Uses in Automation	67
Section 2: Ethernet Communication	68
Focus 1: History of Ethernet	68
Focus 2: Types of Ethernet Topology	
Focus 3: How Ethernet is Useful	70
Chapter 7 - Introduction to Power	71
Section 1: Basics of Power	72
Focus 1: What is Power?	72
Focus 2: What is Voltage	73
Focus 3: What is Current?	
Focus 4: What is Resistance?	75
Focus 5: Ohm's Law	
Focus 4: Electrical Power	77
Section 2: Types of Power Supplies	78
Focus 1: Switched Mode Supplies	78
Focus 2: Uninterruptible Power Supplies	79
Section 3: Power Supply Requirements	80
Focus 1: Calculating Load	80
Focus 2: Calculating Inrush	
Focus 3: Choosing a Power Supply	81
Unit 2 - Centrifugal Pumping	
Chapter 1 - Introduction to Centrifugal Pumping	
Section 1: Define Centrifugal Pumping	
Focus 1: Let's Fling Stones!	
Section 2: Specific Gravity of a Fluid	

Section 4: Friction and Friction Head 93 Section 5: Introduction to Suction 96 Focus 1: Suction Lift. 96 Focus 2: Capacity and Suction Lift. 100 Section 6: Net Positive Suction Head (NPSH) & Cavitation 102 Focus 1: Net Positive Suction Head Available (NPSHa) 103 Focus 1: Net Positive Suction Head Required (NPSHr) 105 Section 7: Cavitation - Another Word for Trouble. 108 Chapter 2: Introduction to Process Control 110 Section 1: Define Process Control 110 Section 2: Electric Motor Operation; a Pump's Driving Force. 112 Focus 1: Watering the Lawn. 110 Section 3: Electric Frequency. 113 Focus 1: Components of an Electric Motor 112 Focus 2: What is Electric Frequency in a Single Phase Motor 115 Focus 3: Inverter 118 Focus 1: Auctifier 119 Focus 2: Direct Current Bus (DC Bus) 120 Focus 4: Three Phase Motors 120 Focus 4: Constant Pressure 121 Focus 5: Effective Voltage 122 Section 4 - Variable Speed Applications in Pumping 125 Foc	Section 3: Viscosity of a Fluid	91
Section 5: Introduction to Suction 96 Focus 1: Suction Lift 96 Focus 2: Capacity and Suction Lift 100 Section 6: Net Positive Suction Head (NPSH) & Cavitation 102 Focus 1: Net Positive Suction Head Required (NPSHr) 103 Focus 2: Net Positive Suction Head Required (NPSHr) 105 Section 7: Cavitation - Another Word for Trouble 108 Chapter 2 - Introduction to Process Control 110 Section 1: Define Process Control 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Components of an Electric Motor 112 Focus 2: What is Electric Frequency in a Single Phase Motor 115 Focus 3: Three Phase Motors 116 Section 3 - Variable Frequency Drives 118 Focus 4: Output of the Inverter 120 Focus 3: Inverter 120 Focus 4: Output of the Inverter 121 Focus 3: Nerter 122 Focus 4: Output of the Inverter 121 Focus 3: Nerter 122 Focus 4: Output of the Inverter 121 Focus 3: Variable Speed Applications in Pumping 125 Focus 4: Soft Star	Section 4: Friction and Friction Head	93
Focus 2 :Capacity and Suction Lift. 100 Section 6: Net Positive Suction Head (NPSH) & Cavitation 102 Focus 1: Net Positive Suction Head Available (NPSHa) 103 Focus 2: Net Positive Suction Head Required (NPSHr) 105 Section 7: Cavitation - Another Word for Trouble 108 Chapter 2 - Introduction to Process Control 110 Section 1: Define Process Control 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Watering the Lawn 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Components of an Electric Motor 112 Focus 2: What is Electric Frequency in a Single Phase Motor 115 Focus 3: RPM as a Function of Frequency in a Single Phase Motor 116 Section 3 - Variable Frequency Drives 118 Focus 3: Inverter 120 Focus 3: Inverter 120 Focus 3: Inverter 120 Focus 3: Constant Flow 121 Focus 3: Constant Pressure 125 Focus 3: Constant Pressure 125 Focus 4: Soft Start 128 Focus 4: Soft Start 128 Foc		
Section 6: Net Positive Suction Head (NPSH) & Cavitation 102 Focus 1: Net Positive Suction Head Required (NPSHa) 103 Focus 2: Net Positive Suction Head Required (NPSHr) 105 Section 7: Cavitation - Another Word for Trouble 108 Chapter 2 - Introduction to Process Control 110 Section 1: Define Process Control 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Components of an Electric Motor 112 Focus 2: What is Electric Frequency 113 Focus 3: RPM as a Function of Frequency in a Single Phase Motor 115 Focus 4: Three Phase Motors 118 Focus 2: Direct Current Bus (DC Bus) 120 Focus 2: Direct Current Bus (DC Bus) 120 Focus 5: Effective Voltage 121 Focus 5: Constant Flow 127 Focus 1: Constant Pressure 125 Focus 1: Constant Pressure 125 Focus 1: Open Loop Control: A Return to Watering the Lawn 130 Focus 1: Open Loop Control: A Return to Watering the Lawn 130 Focus 1: Open Loop Control: A Return to Watering the Lawn 130 Focus 1: Open Loop Control: A Return to Watering the Lawn 130	Focus 1: Suction Lift	96
Focus 1: Net Positive Suction Head Available (NPSHa) 103 Focus 2: Net Positive Suction Head Required (NPSHr) 105 Section 7: Cavitation - Another Word for Trouble 108 Chapter 2 - Introduction to Process Control 110 Section 1: Define Process Control 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Watering the Lawn 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 2: What is Electric Frequency 113 Focus 2: What is Electric Frequency in a Single Phase Motor 115 Focus 3: Three Phase Motors 116 Section 3 - Variable Frequency Drives 118 Focus 1: Reclifier 119 Focus 2: Direct Current Bus (DC Bus) 120 Focus 4: Output of the Inverter 121 Focus 5: Effective Voltage 122 Section 4 - Variable Speed Applications in Pumping 125 Focus 1: Constant Pressure 125 Focus 2: Oreatant Flow 128 Focus 4: Soft Start 128 Section 5 - Process Control Logic 130 Focus 4: Soft Start 128 Section 5 - Proces	Focus 2 :Capacity and Suction Lift	. 100
Focus 1: Net Positive Suction Head Available (NPSHa) 103 Focus 2: Net Positive Suction Head Required (NPSHr) 105 Section 7: Cavitation - Another Word for Trouble 108 Chapter 2 - Introduction to Process Control 110 Section 1: Define Process Control 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Watering the Lawn 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 2: What is Electric Frequency 113 Focus 2: What is Electric Frequency in a Single Phase Motor 115 Focus 3: Three Phase Motors 116 Section 3 - Variable Frequency Drives 118 Focus 1: Reclifier 119 Focus 2: Direct Current Bus (DC Bus) 120 Focus 4: Output of the Inverter 121 Focus 5: Effective Voltage 122 Section 4 - Variable Speed Applications in Pumping 125 Focus 1: Constant Pressure 125 Focus 2: Oreatant Flow 128 Focus 4: Soft Start 128 Section 5 - Process Control Logic 130 Focus 4: Soft Start 128 Section 5 - Proces	Section 6: Net Positive Suction Head (NPSH) & Cavitation	. 102
Focus 2: Net Positive Suction Head Required (NPSHr) 105 Section 7: Cavitation - Another Word for Trouble 108 Chapter 2 - Introduction to Process Control 110 Section 1: Define Process Control 110 Focus 1: Watering the Lawn 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Components of an Electric Motor 112 Focus 2: What is Electric Frequency 113 Focus 3: RPM as a Function of Frequency in a Single Phase Motor 116 Section 3 - Variable Frequency Drives 118 Focus 1: Three Phase Motors 120 Focus 2: Direct Current Bus (DC Bus) 120 Focus 2: Direct Current Bus (DC Bus) 120 Focus 3: Inverter 121 Focus 5: Effective Voltage 122 Section 4 - Variable Speed Applications in Pumping 125 Focus 2: Constant Pressure 125 Focus 2: Constant Flow 128 Section 5 - Process Control Acture to Watering the Lawn 130 Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process) 131 Section 5 - Process Control Acture to Watering the Lawn 130 Focus 3: Depor		
Section 7: Cavitation - Another Word for Trouble 108 Chapter 2 - Introduction to Process Control 110 Section 1: Define Process Control 110 Focus 1: Watering the Lawn 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 2: What is Electric Frequency 113 Focus 3: RPM as a Function of Frequency in a Single Phase Motor 116 Section 3 - Variable Frequency Drives 118 Focus 4: Three Phase Motors 110 Focus 2: Direct Current Bus (DC Bus) 120 Focus 3: Inverter 120 Focus 4: Output of the Inverter 121 Focus 4: Output of the Inverter 122 Section 4 - Variable Speed Applications in Pumping 125 Focus 2: Constant Pressure 125 Focus 2: Constant Pressure 128 Focus 4: Soft Start 128 Section 5 - Process Control Logic 130 Focus 3: Open Loop Control: A Return to Watering the Lawn 130 Focus 4: Soft Start 128 Section 5 - Process Control Logic 133 Focus 4: Soft Start 128 Section 6: The PID Control Precent Preel for Controller Log		
Section 1: Define Process Control 110 Focus 1: Watering the Lawn 110 Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Components of an Electric Motor 112 Focus 2: What is Electric Frequency 113 Focus 3: RPM as a Function of Frequency in a Single Phase Motor 115 Focus 4: Three Phase Motors 116 Section 3 - Variable Frequency Drives 118 Focus 2: Direct Current Bus (DC Bus) 120 Focus 2: Direct Current Bus (DC Bus) 120 Focus 3: Inverter 121 Focus 4: Output of the Inverter 121 Focus 5: Effective Voltage 122 Section 4 - Variable Speed Applications in Pumping 125 Focus 1: Constant Pressure 125 Focus 3: Variable Flow 127 Focus 4: Soft Start 128 Section 5 - Process Control Logic 130 Focus 1: Open Loop Control: A Return to Watering the Lawn 130 Focus 2: Proportional Control (The Pin PID) 133 Focus 2: Proportional Control (The Pin PID) 133 Focus 2: Proportional Control (The Din PID) 131 Section 7: Tuning Methodo		
Focus 1: Watering the Lawn	Chapter 2 - Introduction to Process Control	110
Section 2: Electric Motor Operation; a Pump's Driving Force 112 Focus 1: Components of an Electric Motor 112 Focus 2: What is Electric Frequency 113 Focus 3: RPM as a Function of Frequency in a Single Phase Motor 115 Focus 4: Three Phase Motors 116 Section 3 - Variable Frequency Drives 118 Focus 2: Direct Current Bus (DC Bus) 120 Focus 3: Inverter 120 Focus 4: Output of the Inverter 121 Focus 5: Effective Voltage 122 Section 4 - Variable Speed Applications in Pumping 125 Focus 2: Constant Pressure 125 Focus 3: Variable Flow 128 Focus 4: Soft Start 128 Section 5 - Process Control Logic 130 Focus 2: Closed Loop Control: A Return to Watering the Lawn 130 Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process) 131 Section 6: The PID Control Overview 133 Focus 2: Proportional Control (The P in PID) 136 Focus 3: Integral Control (The I in PID) 139 Focus 4: Derivative Control (The D in PID) 141 Focus 3: Integral Control (The I in PID) 13	Section 1: Define Process Control	. 110
Focus 1: Components of an Electric Motor112Focus 2: What is Electric Frequency113Focus 3: RPM as a Function of Frequency in a Single Phase Motor115Focus 4: Three Phase Motors116Section 3 - Variable Frequency Drives118Focus 2: Direct Current Bus (DC Bus)120Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure126Focus 3: Variable Flow127Focus 4: Soft Start128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 2: Closed Loop Control: A Return to Watering the Lawn130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Controller133Focus 3: Integral Control (The P in PID)139Focus 4: Derivative Control (The P in PID)139Focus 5: Proting It All Together: PID Control141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 1: Introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Focus 1: Watering the Lawn	110
Focus 1: Components of an Electric Motor112Focus 2: What is Electric Frequency113Focus 3: RPM as a Function of Frequency in a Single Phase Motor115Focus 4: Three Phase Motors116Section 3 - Variable Frequency Drives118Focus 2: Direct Current Bus (DC Bus)120Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure126Focus 3: Variable Flow127Focus 4: Soft Start128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 2: Closed Loop Control: A Return to Watering the Lawn130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Controller133Focus 3: Integral Control (The P in PID)139Focus 4: Derivative Control (The P in PID)139Focus 5: Proting It All Together: PID Control141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 1: Introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Section 2: Electric Motor Operation; a Pump's Driving Force	. 112
Focus 2: What is Electric Frequency113Focus 3: RPM as a Function of Frequency in a Single Phase Motor115Focus 4: Three Phase Motors116Section 3 - Variable Frequency Drives118Focus 1: Rectifier119Focus 2: Direct Current Bus (DC Bus)120Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Output of the Inverter130Focus 2: Constant Pressure130Focus 3: Variable Flow128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The P in PID)136Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 1: Introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 3: RPM as a Function of Frequency in a Single Phase Motor.115Focus 4: Three Phase Motors116Section 3 - Variable Frequency Drives118Focus 1: Rectifier119Focus 2: Direct Current Bus (DC Bus)120Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 2: Closed Loop Control: A Return to Watering the Lawn130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Controller133Focus 3: Integral Control (The P in PID)136Focus 3: Integral Control (The P in PID)136Focus 3: Integral Control (The D in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 4: Three Phase Motors116Section 3 - Variable Frequency Drives118Focus 1: Rectifier119Focus 2: Direct Current Bus (DC Bus)120Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Pressure126Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System133Section 6: The PID Controller133Section 6: The PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The P in PID)139Focus 4: Derivative Control (The D in PID)141Focus 4: Derivative Control (The D in PID)141Focus 2: Proportional Control (The D in PID)141Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 1: Rectifier119Focus 2: Direct Current Bus (DC Bus)120Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow127Focus 4: Soft Start128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 2: Closed Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System133Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 2: Direct Current Bus (DC Bus)120Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Controllor133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The I in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Section 3 - Variable Frequency Drives	. 118
Focus 2: Direct Current Bus (DC Bus)120Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Control Overview133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The I in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Focus 1: Rectifier	119
Focus 3: Inverter120Focus 4: Output of the Inverter121Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Control Overview133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error146Focus 3: The Ziegler-Nichols Method146		
Focus 5: Effective Voltage122Section 4 - Variable Speed Applications in Pumping125Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The D in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error146Focus 3: The Ziegler-Nichols Method146	Focus 4: Output of the Inverter	. 121
Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Control143Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 1: Constant Pressure125Focus 2: Constant Flow127Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Control143Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	•	
Focus 3: Variable Flow128Focus 4: Soft Start128Section 5 - Process Control Logic130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process)131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 4: Soft Start128Section 5 - Process Control Logic.130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process).131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Focus 2: Constant Flow	. 127
Section 5 - Process Control Logic.130Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process).131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Focus 3: Variable Flow	. 128
Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process).131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Focus 4: Soft Start	. 128
Focus 1: Open Loop Control: A Return to Watering the Lawn130Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process).131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Section 5 - Process Control Logic	. 130
Focus 2: Closed Loop Control: The Heating System130Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process).131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 1: Introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 3: Getting a Better Feel for Controller Logic (Hand-Controlling the Process).131Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Section 6: The PID Controller133Focus 1: PID Control Overview133Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 2: Proportional Control (The P in PID)136Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 1: Introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 1: Introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Focus 1: PID Control Overview	. 133
Focus 3: Integral Control (The I in PID)139Focus 4: Derivative Control (The D in PID)141Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 1: Introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146		
Focus 5: Putting It All Together: PID Control142Section 7: Tuning Methodologies-An introduction to Gain145Focus 1: Introduction to Gain145Focus 2: Manual Tuning-Trial and Error145Focus 3: The Ziegler-Nichols Method146	Focus 3: Integral Control (The I in PID)	. 139
Section 7: Tuning Methodologies-An introduction to Gain	Focus 4: Derivative Control (The D in PID)	. 141
Focus 1: Introduction to Gain	Focus 5: Putting It All Together: PID Control	. 142
Focus 1: Introduction to Gain	Section 7: Tuning Methodologies-An introduction to Gain	. 145
Focus 3: The Ziegler-Nichols Method 146		
Focus 3: The Ziegler-Nichols Method 146	Focus 2: Manual Tuning-Trial and Error	. 145
	•	
	Focus 4: PID Tuning Software	
Unit 3 - Ethernet Communication	-	

Chapter 1 - Introduction to Industrial Ethernet	. 148
Section 1: What is Ethernet?	. 149
Focus 1: Types of Ethernet	. 149
Focus 2: Types of LAN Technology	. 150
Focus 3: IP (Internet Protocol)	. 151
Focus 4: TCP (Transmission Control Protocol)	. 151
Focus 5: UDP (User Datagram Protocol)	. 152
Focus 6: IP Addressing	. 152
Section 2: Types of Switched Ethernet	
Focus 1: Basic Switches	. 153
Focus 2: Intelligent Switches	. 154
Focus 3: Managed Switches	
Chapter 2 - Network Setup	. 157
Section 1 - IP Address Classification	. 157
Focus 1: Binary Code	. 157
Focus 2: Addressing	
Focus 3: Classes	. 159
Section 2: Subnets	
Focus 1: What are Subnets?	. 162
Focus 2: Subnet Benefits	. 162
Section 3: IP Routing	
Focus 1: What is IP Routing?	
Focus 2: Routers	
Chapter 3 - Communication Setup	
Section 1: PAC Ethernet Setup	
Section 2: HMI Ethernet Setup	
Section 3: VFD Ethernet Setup	
Chapter 4 - Ethernet with Wireless Technology	
Section 1 - Introduction to WLAN	
Focus 1 - What is WLAN?	
Focus 2 - Access Points	
Focus 3 - Security	
Focus 4 - Interference	
Unit 4 - Programmable Automation Controller	176
Chapter 1 - Introduction to CompactLogix 5370 L1 Controller	. 177
Section 1: 1769-L18ER Controller	. 178
Focus 1: Memory	. 178
Focus 2: Embedded Inputs / Outputs	. 179
Focus 3: 1734 Series Expansion Cards	. 183
Section 2: Communication	
Chapter 2 - Embedded Input / Output	. 186
Section 1: Inputs (Sinking)	
Focus 1: Drain Pump Float Switch	
Focus 2: Solenoid Valve Float Switch	. 191

Section 2: Outputs (Sourcing)	195
Focus 1: Prime Pump	195
Focus 2 - Tank Heater	198
Focus 3 - Secondary Tank Drain Pump	203
Focus 4 - Secondary Tank Solenoid Fill Valve	205
Section 3: Sports Drink Scenario	207
Focus 1 - Start the Process	207
Focus 2 - Fluid Process	208
Focus 3 - Back to the Process Tank	209
Chapter 3 - I / O Expansion Modules	212
Section 1: 1734 Module Installation	
Focus 1 - 1734 Point I/O System	213
Focus 2 - 1734 Point I/O Features	
Focus 3 - 1734 Module Installation	215
Focus 4 - Wiring the Modules	218
Section 2: 1734-IE4C Analog Input Cards	
Focus 1: Pressure: 4-20mA Transducer	
Section 2: 1734-IE4C Analog Input Cards	224
Focus 2: Suction: 4-20mA Transducer	
Focus 3: Tank Levels: 4-20mA FlowLine Ultrasonic Sensor	227
Section 3: 1734-IT2I Temperature Input Card	230
Focus 1: Primary Tank Flow Temperature	230
Focus 2: Secondary Tank Temperature	
Section 4: 1734-VHSC24 Flow Input Card	236
Focus 1: Flow Meter: Pulse Frequency Counters	236
Section 5: Pharmaceudical Scenario	241
Focus 1 - Process Plant	241
Focus 2 - Buffering and Bottling a Batch of Eyedrops	243
Focus 3 - Buffering and Bottling a Batch of Contact Lens Solution	246
Chapter 4 - Tag Database	251
Section 1: Project Structure	252
Focus 1 - Project	252
Focus 2 - Task	
Focus 3 - Program	254
Focus 4 - Routines	255
Focus 1 - Tag Memory	256
Focus 2 - Tag Naming Rules	257
Focus 3 - I/O Tag Format	258
Focus 4 - Tag Locations	259
Section 2: Tag Basics	260
Focus 5 - FLUID <i>Mechatronix</i> ™ Tag Values	260
Section 3: Types of Tags	263
Focus 1: BOOL Tags	263
Focus 2: DINT Tags	264
Focus 3: REAL Tags	<u> 265</u>

Focus 4: STRINGS	. 266
Unit 5 - Human Machine Interface	268
Chapter 1 - Introduction to Dynics SW15	
Section 1: Dynics Computer	
Focus 1: Operating System	
Focus 2: RAM	
Focus 3: Storage	
Focus 4: Central Processing Unit	
Section 2: Communication	
Focus 1: Dual LAN Inputs	
Focus 2: PAC and HMI Data Flow	
Chapter 2 - HMI Software	
Section 1: Studio 5000 Logix Designer Programming Environment	
Focus 1: Creating a New Project	
Focus 2: Ladder Logic	
Focus 3: Adding Expansion Modules	
Focus 4: Function Block Diagram	
Focus 5: Structured Text	
Focus 6: I/O Tag Data	
Section 2: FactoryTalk View ME Operator Environment	
Focus 1: Communication	
Focus 2: Tags	
Focus 3: Displays	
Focus 4: Global Objects	
Focus 5: Alarms	
Chapter 3 - Input / Output	
Section 1: HMI Operator Inputs	
Focus 1: Toggles	
Focus 2: Multistate Buttons	
Focus 3: Set Points	
Section 2: GracePort I/O	
Focus 1: USB	
Focus 2: HDMI	
Focus 3: VGA	
Focus 4: Ethernet	
Chapter 4 - VNC Control	
Section 1: VNC Server	
Focus 1: VNC Network Communications	
Focus 2: HMI VNC Server	
Section 2: VNC Client	
Focus 1: Network Communication	
Focus 2: Connecting to a VNC Server	. 329
Unit 6 - Variable Frequency Drive	330
Chapter 1 - Introduction to PowerFlex 525	

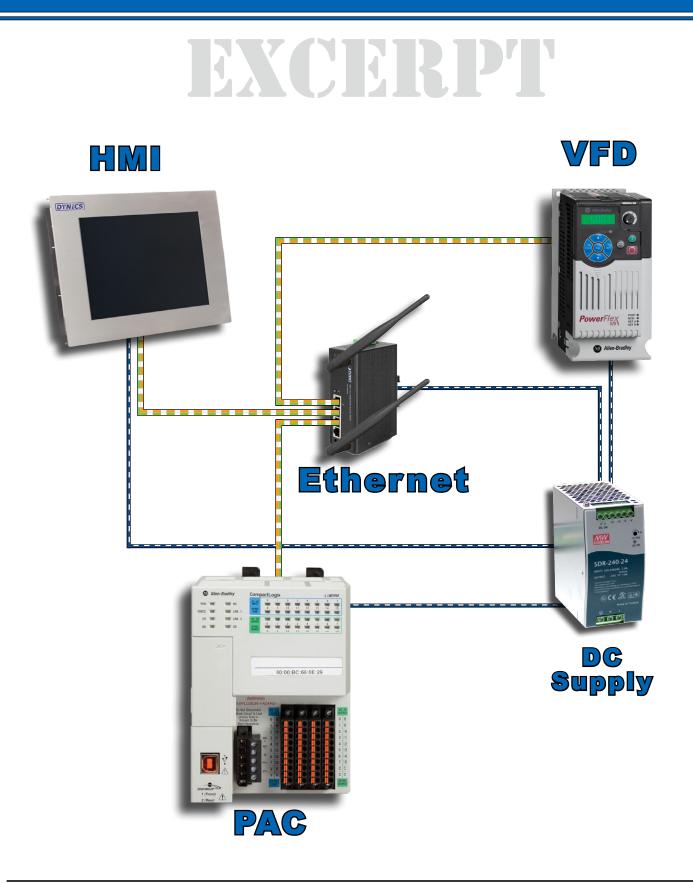
Section 1: Drive Functions	
Focus 1: I/O	
Focus 2: Basic Control	333
Focus 3: Feedback Control	334
Section 2: Communication	
Focus 1: VFD Data Flow	335
Focus 2: Embedded	
Focus 3: Versatile Programming and Network Solutions	337
Chapter 2 - Keypad Control	338
Section 1: Faceplate Layout	
Focus 1: Navigation	
Focus 2: Manual Controls	340
Focus 3: Feedback Control	341
Section 2: Motor Nameplate Data	
Chapter 3 - Safety Input	343
Section 1: Safe-Torque-Off Function	
Focus 1: Overview	344
Focus 2: Embedded Safety	345
Unit 7 - Safety	346
Chapter 1 - Automation Safety	
Section 1: Automation Safety	
Focus 1: Failsafe Programming	
Focus 2: Product Integrity	
Section 2: Operator Safety	
Focus 1: Proximity Sensors	350
Focus 2: Power Disconnects	351
Focus 3: E Stops	352
Focus 4: Safety Relays	353
Chapter 2 - Alarms / Notifications	354
Section 1: Physical Alarms / Notifications	
Focus 1: VFD Status Lights	355
Focus 2: Safety Relay MSR127TP Status Lights	
Focus 3: SensaGuard Pump Housing Status Light	357
Focus 4: Power Buttons and Switches	
Section 2: Programmed Alarms / Notifications	
Focus 1: Main Tank Level Alarms / Notifications	359
Focus 2: Secondary Tank Level Float Switches	
Focus 3: Temperature Alarms	361
Chapter 3 - System Safety Circuits	362
Section 1: Tank Heating Circuit	
Focus 1: Over Temperature Alarm	
Focus 2: Heater Time Limit Notification	
Focus 2: Heater Time Limit Notification	
Section 2: Fluid Level Circuit	

Focus 1: Main Tank Levels	365
Focus 2: Secondary Tank Levels	365
Focus 2: Secondary Tank Levels	366
Section 3: VFD Circuit	
Focus 1: VFD Dual Safety Inputs	367
Focus 2: MSR127TP Safety Relay	367
Focus 3: SensaGuard Switch	367
Focus 4: E Stop	
Focus 2: MSR127TP Safety Relay	
Focus 3: SensaGuard Switch	
Focus 4: E Stop	
Chapter 4 - Lockout Tagout	
Section 1: Lockouts	
Focus 1: Power Disconnects	
Focus 2: Circuit Breakers	
Focus 3: Padlocks and Cable Locks	
Section 2: Tagouts	375
Unit 8 - Studio 5000 Logix Designer	
Chapter 1 - Introduction Studio 5000 Logix Designer	
Section 1: Creating a Project	
Section 2: Connecting Your Computer to the Controller	
Section 3: Downloading Project from the Computer to the Controller	
Section 4: Configuring I/O	
Section 5: Testing Your Logic Program	
Section 6: Adding Logic and Tags Online	
Section 7: Creating and Running a Trend	
Section 8: (Optional) Creating and Using User Defined Types (UDT)	
Section 9: (Optional) Using Studio 5000 Help	
Chapter 2 - I/O and Tag Data	
Section 1: Communicate with I/O Modules	
Section 2: Organize Tags	
Section 3: Force I/O	
Section 4: Data Access Control	
Chapter 3 - Ladder Logic Programming	379
Section 1: Introduction	
Section 2: Write Ladder Logic	
Section 3: Enter Ladder Logic	
Section 4: Assign Instruction Operands	
Section 5: Enter A Rung Comment	379
Section 6: Verify the Routine	
Chapter 4 - Structured Text Programming	380
Section 1: Introduction	
Section 2: Assignments	
Section 3: Expressions	

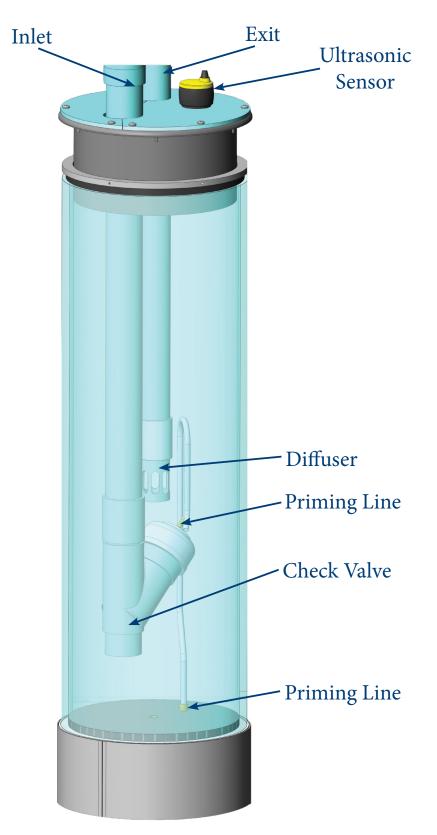
Section 4: Instructions	
Section 5: Constructs	
Section 6: If Then	
Section 7: Case Of	
Section 8: For Do	
Section 9: While Do	
Section 10: Repeat Until	
Chapter 5 - Function Block Diagram Programming	
Section 1: Introduction	
Section 2: Choose the function block	
Section 3: Choose a tag name for an element	
Section 4: Define the order of execution	
Section 5: Identify any connectors	
Section 6: Define program/operator control	
Section 7: Add a sheet	
Section 8: Add a function block element	
Section 9: Create a text box	
Section 10: Connect elements	
Section 11: Assign a tag	
Section 12: Assign an immediate value (constant)	
Section 13: Connect blocks with an OCON and ICON	
Section 14: Verify the routine	
Chapter 6 - Plant PAX	
Section 1 - PlantPAX Library of Process Objects	
Focus 1: Overview	
Focus 2: How to Install the Library	
Focus 3: Common Configuration Considerations	
Focus 4: Use the Library	
Section 2: P_RunTime	
Section 3: P_VSD	
Section 4: P_Perm	
Section 5: P_Intlk	
Section 6: P_Alarm	
Section 7: P_AIn	
Section 8: P_Mode	
Section 9: P_PIDE	
Unit 9 - FactoryTalk View ME	
Chapter 1 - FactoryTalk View ME User's Guide	
Section 1: Getting Started	
Section 2: Explore FactoryTalk View Studio	
Section 3: Plan applications	
Section 4: Work with applications	
Section 5: Set up communications	
Section 6: Work with tags	
v	

Section 7: Use HMI tags	392
Section 8: Set up global connections	392
Section 9: Set up alarms	392
Section 10: Set up FactoryTalk Diagnostics	392
Section 11: Set up security	392
Section 12: Set up language switching	392
Section 13: Set up display navigation	392
Section 14: Create run-time applications	392
Section 15: Run applications on a personal computer	392
Section 16: Transfer applications to a PanelView Plus terminal	392
Section 17: Use your application	393
Section 18: Work with components	393
Section 19: Use graphic displays	393
Section 20: Use graphic objects	393
Section 21: Set up graphic objects	393
Section 22: Animate graphic objects	393
Section 23: Use expressions	393
Section 24: Use embedded variables	393
Section 25: Use parameters and global objects	393
Section 26: Set up data logging	393
Section 27: Use information messages	393
Section 28: Set up trends	393
Section 29: Set up RecipePlus	393
Section 30: Use macros	393

Unit 1 - General Overview



Primary Tank Components



30

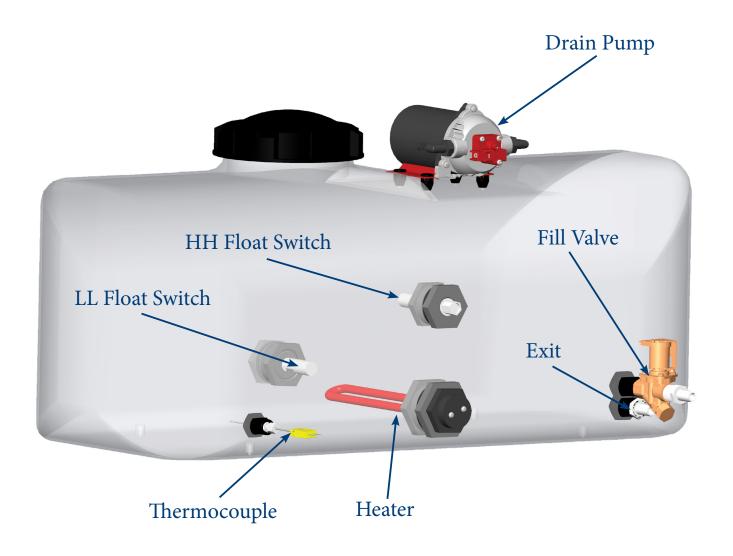
Focus 1 - Primary Tank Controls

Fluid*Mechatronix*[™] is equipped with two different fluid storage tanks. The main tank is the visible tank on the corner of the unit. The secondary tank is inside the unit. The main tank represents the main fluid and system monitoring. The main tank has a check valve near the bottom of the tank fluid inlet. This helps hold the prime of the fluid circuit on the top of the unit. The tank exit has a diffuser installed on the end. This changes the direction of fluid flow when re-entering the tank and will reduce tank agitation and air bubbles. The small line coming in above the check valve is for priming the flow system. This forces water into the top flow circuit so the main pump can be operated.

On top of the tank there is an ultrasonic sensor to indicate main tank fluid levels. This sensor monitors the level of the fluid in the tank with high accuracy. This is discussed further in Section 3 > Focus 4.

Primary Tank				
		Tank Temperature		
	Fill Pump OFF	Analog Input		
	Tank Level	18.42 C		
		Tank Drain Valve		
		S Closed		
		Valve Control		
		Valve Control		
		Circulation Pump		
		Motor Control		
		Stopped		
	-18.67	(0)		
	cm	Motor Speed Control		
		0.00 Hz		

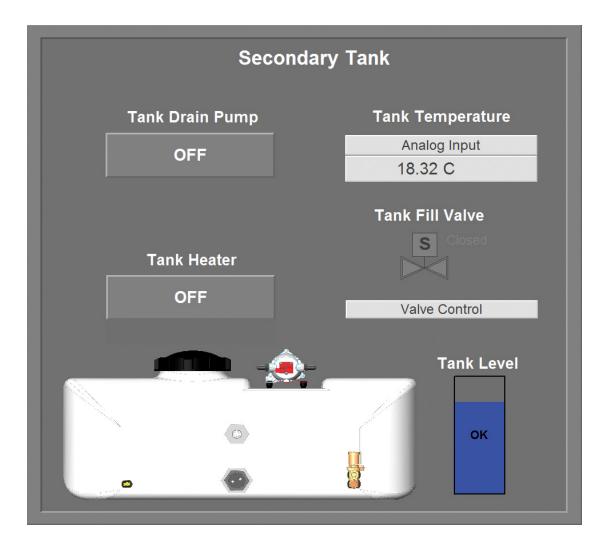
Secondary Tank Components



Focus 2 - Secondary Tank Controls

The second tank provides more flexibility of the system. The second tank contains a heater, a valve for level control, and a pump for moving fluid from the secondary tank to the main tank. There are also high and low float switches for over and under filling.

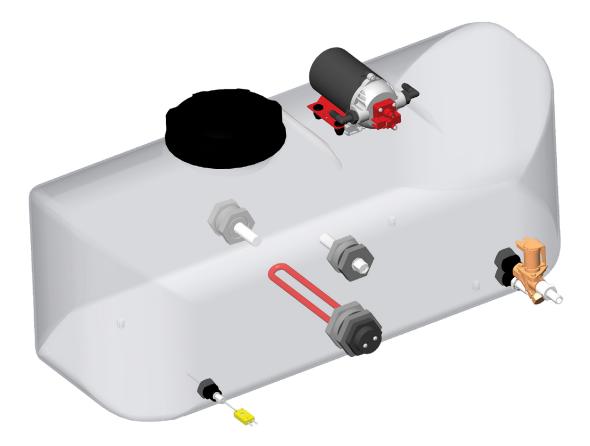
The tank has an inlet from the bottom of the main tank, and has an exit that is pumped back into the main tank. At the inlet there is a solenoid valve that allows controlled entrance of water into the tank. The valve is small, which increases the drain time of the main tank. This helps simulate a large main tank with slow drain times.



Focus 3 - Temperature Control

The second tank has a heating circuit that allows the user to heat the water for 10 minutes. This tank gives Fluid*Mechatronix*TM the ablility to mix fluids with different temperatures in one system. There is a thermocouple mounted in the tank for temperature monitoring. The heater is programmed to operate between 10° and 50° C. If the thermocouple detects a temperature that is out of range, the heating circuit relay will not engage. The heating circuit raises the temperature in the tank about 5 degrees. There is a timing circuit on the heater, so the heater automatically shuts off after 10 minutes. This is a safety measure to prevent heating element damage if the tank runs low on water.

The tank exit pump is wired with the LL float switch. The float switch is located above the heating element in the back of the tank. This prevents the pump from removing too much water, and exposing the heating element.

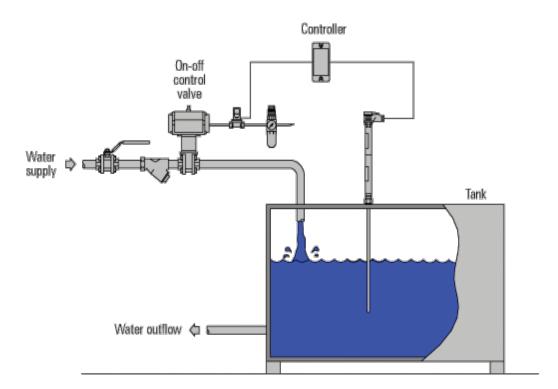


Focus 4 - Fluid Level Control with Pumps and Valves

Pumps and solenoids perform 2 functions. First, they can removes water from a tank to decrease the fluid levels. This would be useful when studying the setpoints and alarms of a tank system. Second, they can increase the fluid levels in a tank. For a solenoid, water flows from a tank to the secondary tank when this solenoid is open. Head pressure or water pressure is the only thing moving the fluid, so the process can be a little slow. When using head pressure alone, the increase or decrease in the tank can be difficult to see visually.

Fluid*Mechatronix*[™] has fluid level control in 4 variations:

- Drain a tank with a solenoid valve
- Fill a tank with a solenoid valve
- Drain a tank with a pump
- Fill a tank witha pump

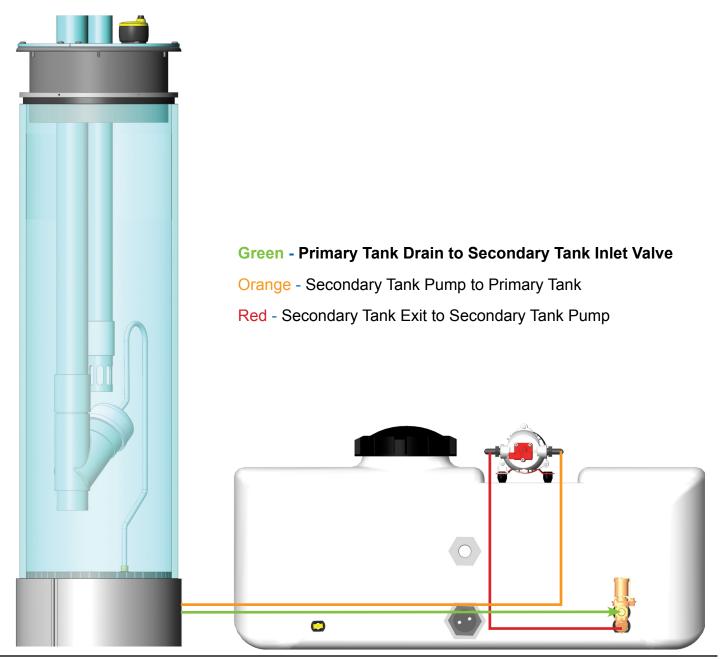


Basic Tank Level Control System

Focus 5 - Solenoid Valve Control

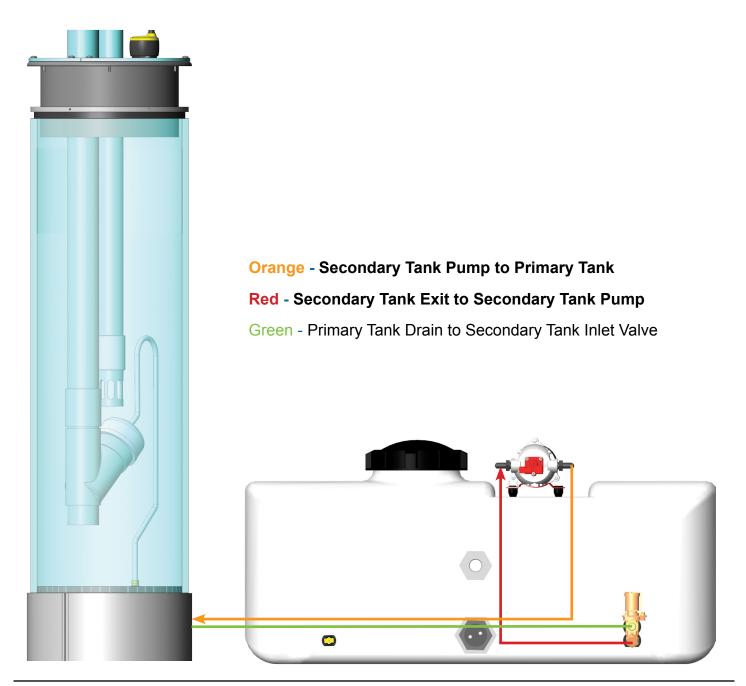
Opening the secondary tank solenoid affects the levels of the main tank. When the solenoid is open, fluid drains from the primary tank and fills the secondary tank. This would be very similar to a system with a large tank, and a relatively small inlet and exit. The decreasing fluid level of the main tank is difficult to see, but the software can graphically track the process and give the operator a visual indication of the tanks level and progress.

Fluid will fill the secondary tank until the HH float limit switch is reached. When fluid reaches the float and lifts it up, the solenoids electrical circuit is broken and it returns to the default state of normally closed. See diagram below for the plumbing of the Primary and Secondary Tank:



Focus 6 - Pump Control

The secondary tank pump also affects the levels of the main tank. When the secondary tank is full of fluid, the pump can be enabled to pump fluid back to the primary tank. The pump runs continuously until it is shut off, or until it reaches the LL float limit switch. If the LL float limit switch falls to its open position due to lack of water, the prime pump will turn off. In reality, the float switch stops passing electricity to the pump relay. This causes the relay to turn off, and the relays output drives the pump, therefore the pump will turn off. The pump will not be able to turn back on until the fluid levels in the secondary tank increase.



Skill Section 1, Focus 1-6

- 1. What are some of the Primary Tank Control features?
- 2. Describe a similar situation where fluid level control is used.
- 3. What does a PLR typically show on the LCD display?

Section 3: Power Supply Requirements

Focus 3: Choosing a Power Supply

Most power supplies are designed to handle higher inrush currents for a brief period of time (like 1 second). After that, the electrical load of the system needs to be below the power supplies ratings. It is a good practice to avoid constant operation of a power supply near its limits. Typically the power supply needs to be about 20% larger than the maximum load conditions of the electrical devices connected to it. What would the proper power supply size be for the 5A motor?

To be safe, we want a supply 20% larger than normal load so we need to multiply by 120%

$$P = IV = 5A \cdot 24V = 120W \cdot 120\% = 144W$$

In this case, it would be best to purchase a supply that can handle 144 watts of power continuously and can handle an inrush of 216 watts.

Section 3, Focus 1

1. A 12V motor has a maximum current draw of 7A, with a normal operating current of 5A. Calculate the proper size for the power supply.

2. If a 2W PAC and a 100W HMI are connected to the motor above, what size power supply would be required to operate the system?

Unit 2 - Centrifugal Pumping





Chapter 1 - Introduction to Centrifugal Pumping

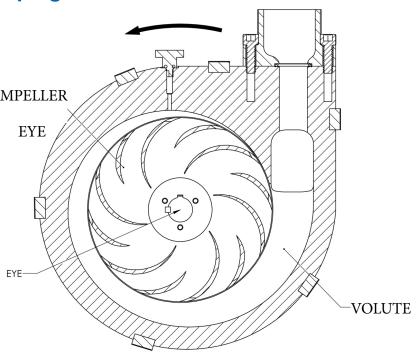
Student will understand what a centrifugal pump is, what the major components of a centrifugal pump are, what pumping head is, and how to size a pump impeller.

Section 1: Define Centrifugal Pumping

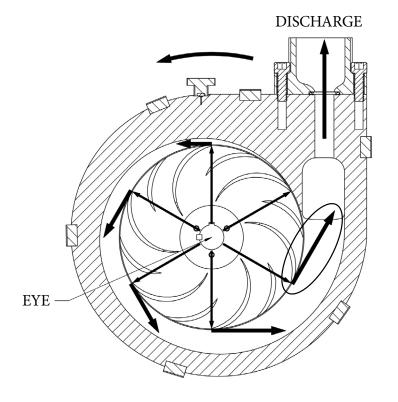
Centrifugal pumps are used in many applications including water, sewage, petroleum and petrochemical pumping. They have become a primary component of many industries. A centrifugal pump is a machine that IMPELLER gives energy to a fluid. This energy infusion can cause a liquid to flow, rise to a higher level, or both.

The centrifugal pump is a very simple rotary machine which consists of two major parts: 1) impeller (rotating component) and 2) volute (the stationary structure or casing). The figure shows these two parts (along with the eye, which is the fluid inlet into the volute).

The centrifugal pump's function is as simple as its design. It is filled with liquid and the impeller is rotated (usually with an electric motor). The impeller vanes grab the liquid and "fling" it, giving it energy, which causes it to exit the impeller's vanes at a greater velocity than it possessed when it entered.







This outward flow (flinging) reduces the pressure at the impeller eye (the center where the inlet piping allows the fluid to enter the casing), allowing more liquid to enter. The liquid exits the impeller at its edge (peripheral velocity) and is collected in the casing (volute) where its velocity is converted to pressure before it leaves the pump's discharge.

Figure 1.2: Impeller Outward Flow

What does a real centrifugal pump look like?



Figure 1.3: Commercial Centrifugal Pump

What does a real centrifugal impeller look like?



Figure 1.4: Centrifugal Pump Impeller

Section 1: Define Centrifugal Pumping

Focus 1: Let's Fling Stones!

(How gravity and velocity relate to centrifugal pumping)

Gravity is one of the more important forces that a centrifugal pump must overcome. You will find that the relationship between final velocity, due to gravity, and initial velocity, due to impeller speed, is a very useful one.

Let's use an example that's easy to visualize to better understand what we're talking about.

If a stone is dropped from the top of a building, its speed (velocity) will increase at a known rate of 32.2 feet per second for each second that it falls. This increase in velocity is known as acceleration due to gravity.

If we ignore the effect of air resistance on the falling stone, we can predict the velocity it will hit the ground based upon its starting height off the ground and the effect of acceleration due to gravity (that's 32.2 feet per second PER second).

Now, with help of mathematics and even Sir Isaac Newton himself (you remember that he discovered this thing called gravity), a convenient equation was developed which describes the relationship of velocity, height, and gravity as it applies to a falling body, like that stone as:

$$v^2 = 2gh$$

Where:

 \mathbf{v} = the velocity of the body in ft/sec

 \mathbf{g} = the acceleration due to gravity @ 32.2 ft/sec/sec (or ft/sec²)

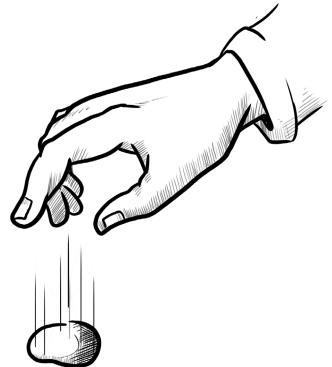
 \mathbf{h} = the distance through which the body falls

For example, if a stone is dropped from a building 100 feet high:

$$v^2 = 2 \times 32.2 \frac{ft}{s^2} \times 100 ft$$
, which gives us..
 $v^2 = 6440 \frac{ft^2}{s}$

$$v = \sqrt{6440 \frac{ft^2}{s^2}} = 80.3 \frac{ft}{s}$$

The stone, therefore, will strike the ground at a velocity of 80.3 feet per second (about 55 miles per hour).



Previous

Section 1: Define Centrifugal Pumping

Now, let's reverse this scenario. That same equation allows us to figure out how fast we would have to throw that stone (initial velocity) upward from ground level for it to reach the top of the building (100 feet). This is true because the **final velocity** of a falling body **happens to be equal** to the **initial velocity** required to launch it to get it to that same height from which it fell (got that?). Using the example above, the initial velocity required to throw the stone to a height of 100 feet is 80.3 feet per second, the same as its final velocity.

So what does dropping and throwing stones have to do with pumping? Here's the cool part: the same equation we used for the stone applies when pumping water with a centrifugal pump. The velocity of the water as it leaves the impeller determines the head developed. In other words the water is "thrown" to a certain height. To reach this height it must start with the same velocity it would attain if it fell from that height.

So, if we rearrange the falling body equation we get:

$$h = \frac{v^2}{2g}$$

Now we can determine the height to which a body (or water) will rise given a particular initial velocity. For example, at 10 ft per sec:

$$h = \frac{\left(10\frac{ft}{s}\right)^2}{2 \times 32.2\frac{ft}{s^2}}$$

h = 1.55 ft

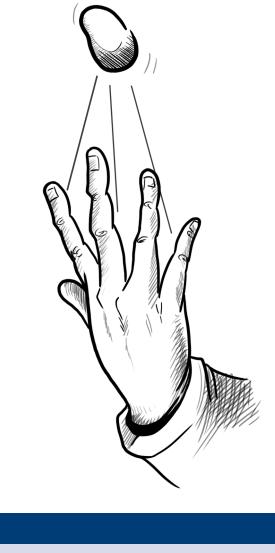


Section 1, Focus 1

1. Calculate the velocity of a marble at impact if it is dropped from a 6 story building (each story is 12 ft. high). Disregard any air resistance.

2. At what speed would we need to launch the marble from the ground level to reach the top of the building?

3. If you were to try this with several different initial velocities, you would find out that there is an interesting relationship between the height achieved by a body and its initial velocity. **This relationship is one of the Affinity Laws of centrifugal pumps which can be studied independently.**



Section 1: Define Centrifugal Pumping

As a finale to this section, let's apply what we have learned to a practical application. Follow each step carefully and you will gain a good understanding of how this works (you'll feel so smart).

Problem: We have a centrifugal pump that is spinning at 1800 RPM. We need to figure out what impeller diameter would be needed to fling the water up 200 feet (a.k.a. Develop a head of 200 feet).

<u>First</u> we must calculate the initial velocity required to develop a head of 200 feet. Using our falling body equation from before: $v^2 = 2gh$

$$v^{2} = 2 \times 32.2 \frac{ft}{s^{2}} \times 200 \, ft$$
$$v^{2} = 12,880 \frac{ft^{2}}{s^{2}}$$
$$v = \sqrt{12,880 \frac{ft^{2}}{s^{2}}} = 113 \frac{ft}{s}$$

We also need to know the number of rotations the impeller undergoes each second:

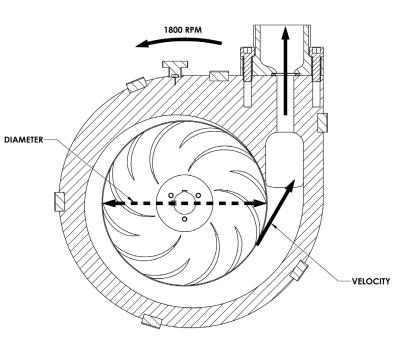
1800 RPM / 60 sec = 30 RPS (Revolutions per Second).

Now we can compute the number of feet a point on the impellers rim travels in a single rotation: 113 ft/sec divided by 30 rotations/sec = 3.77 ft/rotation

Since distance traveled per rotation is **the same** as the circumference of the impeller we can compute the diameter using the circular relation formula as follows:

$$C = \pi D \Longrightarrow D = \frac{C}{\pi}$$
$$D = \frac{3.77 \text{ ft}}{\pi}$$
$$D = 1.2 \text{ ft} = 14.4 \text{ in}$$

Therefore an impeller with a diameter of approximately 14.4" turning at 1800 RPM will produce a head of (fling water up) 200 Feet.



Section 1, Focus 2

We have a centrifugal pump that is spinning at 1800 RPM. What impeller diameter would be needed to fling the water up 122 feet (a.k.a. Develop a head of 122 feet).

Focus 3: The Ziegler-Nichols Method

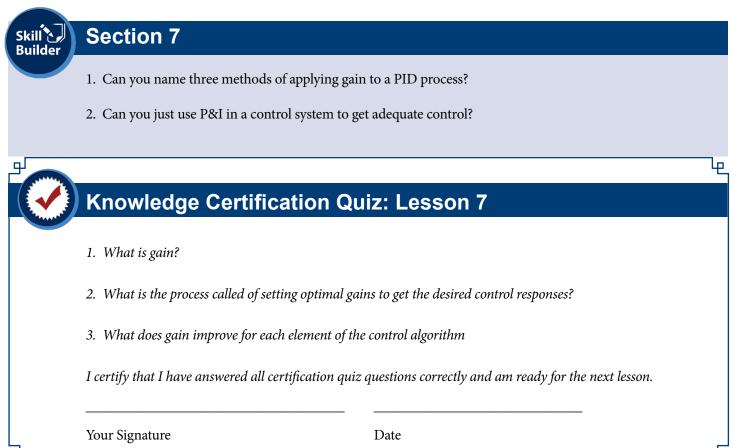
The Ziegler-Nichols method is another popular method of tuning a PID controller. It is very similar to the trial and error method wherein I and D are set to zero and P is increased until the loop starts to oscillate. Once oscillation starts, the ultimate gain K_u and the period of oscillations T_u are noted. The P, I and D are then adjusted as per the tabular column shown below.

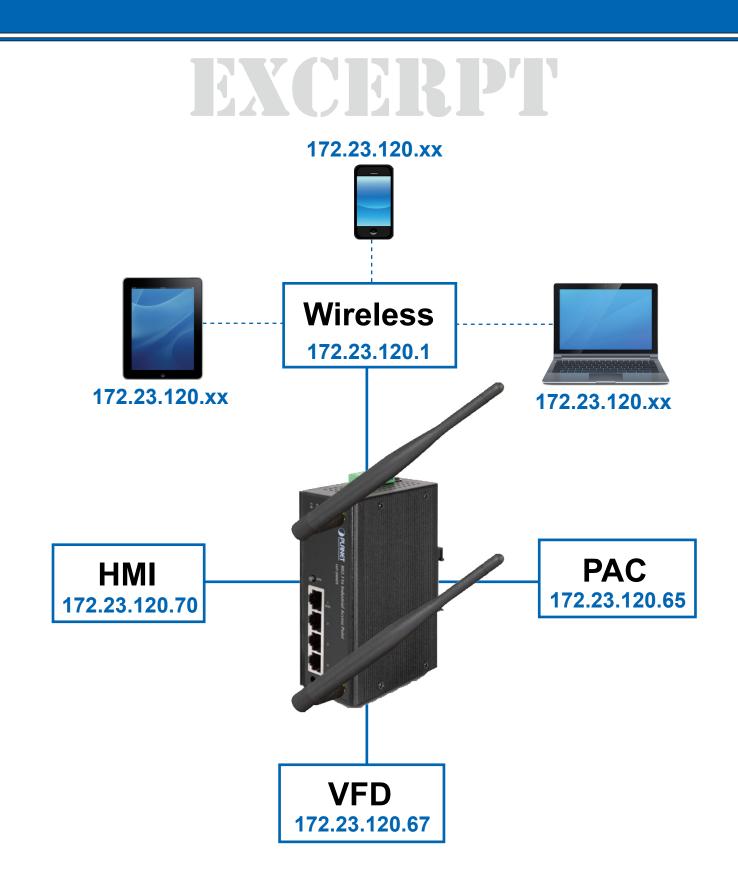
Zie	egler-Nichols	Method	
Control Type	Kp	K _i	K _d
PID	0.6 K _u	2/T _u	0.125 T _u

Table 7.1- Ziegler-Nichols tuning, using the oscillation method

Focus 4: PID Tuning Software

Modern process facilities, especially with complex processes, can also utilize PID tuning and loop optimization software to ensure consistent results. These software packages will gather the data, develop process models of the system and suggest optimal tuning. There are a variety of programs available, including custom application developments.





Next

Chapter 1 - Introduction to Industrial Ethernet

Section 1 – What is Ethernet?

- IP
- TCP
- UDP
- IP Addressing

Section 2 – Switched Ethernet

- Basic Switches
- Managed Switches
- Intelligent Switches



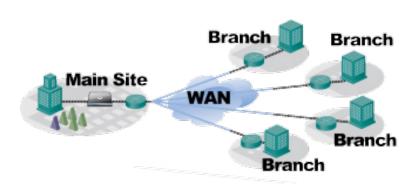
Manufacturing companies are increasingly expanding their global operations to address new opportunities and reduce operating costs. They are also seeking to continuously improve efficiency and drive down costs for existing facilities and processes. Achieving the goals of globalization and operations excellence requires improving connectivity between plant and business systems for real-time visibility to information and effective collaboration. This helps to assure consistent quality and performance across global operations, and to reduce the cost of design, deployment, and support of distributed manufacturing and IT systems. Manufacturers must be able to balance production with demand to optimize material usage and asset utilization, while continuing to meet increasingly exacting customer requirements and metrics for on-time delivery. Manufacturers also need to improve response to events that occur on the plant floor, regardless of location, while implementing more flexible and agile operations in order to react to rapidly changing market conditions.

As manufacturers seek to improve processes many are turning to Ethernet technology on the factory floor. This migration is rapidly gaining momentum. To deploy this technology, engineers on the manufacturing floor should be familiar with some of the important concepts behind Industrial Ethernet.

Focus 1: Types of Ethernet

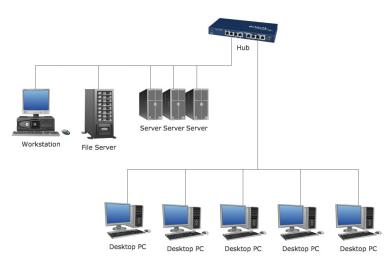
Local Area Networks (LANs)

A network is any collection of independent computers that exchange information with each other over a shared communication medium. Local Area Networks or LANs are usually confined to a limited geographic area, such as a single building or a college campus. LANs can be small, linking as few as three computers, but can often link hundreds of computers used by thousands of people. The development of standard networking protocols and media has resulted in worldwide proliferation of LANs throughout business and educational organizations.



Wireless Local Area Networks (WLANs)

Wireless LANs, or WLANs, use radio frequency (RF) technology to transmit and receive data over the air. This minimizes the need for wired connections. WLANs give users mobility as they allow connection to a local area network without having to be physically connected by a cable. This freedom means users can access shared resources without looking for a place to plug in cables, provided that their terminals are mobile and within the designated network coverage area. With mobility, WLANs give flexibility and increased productivity, appealing to both entrepreneurs and to home users. WLANs may also enable network administrators to connect devices that may be physically difficult to reach with a cable.



Wide Area Networks (WANs)

Often elements of a network are widely separated physically. Wide area networking combines multiple LANs that are geographically separate. This is accomplished by connecting the several LANs with dedicated leased lines such as a T1 or a T3, by dial-up phone lines (both synchronous and asynchronous), by satellite links and by data packet carrier services. WANs can be as simple as a modem and a remote access server for employees to dial into, or it can be as complex as hundreds of branch offices globally linked. Special routing protocols and filters minimize the expense of sending data over vast distances.



Focus 2: Types of LAN Technology

Ethernet is by far the most widely used LAN technology today. It is estimated that more than 85 percent of the world's LAN connected PCs and workstations use Ethernet. Ethernet refers to the family of computer networking technologies covered by the IEEE 802.3 standard, and can run over both optical fiber and twisted-pair cables. Over the years, Ethernet has steadily evolved to provide additional performance and network intelligence. This continual improvement has made Ethernet an excellent solution for industrial applications. Today, the technology can provide four data rates.

• 10BASE-T Ethernet delivers performance of up to 10 Mbps over twisted-pair copper cable.

• Fast Ethernet delivers a speed increase of 10 times the 10BASE-T Ethernet specification (100 Mbps) while retaining many of Ethernet's technical specifications. These similarities enable organizations to use 10BASE-T applications and network management tools on Fast Ethernet networks.

• Gigabit Ethernet extends the Ethernet protocol even further, increasing speed tenfold over Fast Ethernet to 1000 Mbps, or 1 Gbps. Because it is based upon the current Ethernet standard and compatible with the installed base of Ethernet and Fast Ethernet switches and routers, network managers can support Gigabit Ethernet without needing to retrain or learn a new technology.

• 10 Gigabit Ethernet, ratified as a standard in June 2002, is an even faster version of Ethernet. Because 10 Gigabit Ethernet is a type of Ethernet, it can support intelligent Ethernet-based network services, inter-operate with existing architectures, and minimize users' learning curves.

To date, more than 300 million switched Ethernet ports have been installed worldwide. Ethernet technology has a wide acceptance because it is easy to understand, deploy, manage, and maintain. Ethernet is low-cost and flexible, and supports a variety of network topologies. Compared to traditional, non-Ethernet-based industrial solutions that have a data rate of between 500 Kbps to 12 Mbps, Ethernet technology can deliver substantially higher performance. It is based on industry standards, and it can connect over any Ethernet-compliant device from any vendor.

Focus 3: IP (Internet Protocol)

IP is often understood as the Internet Protocol suite: A suite of protocols and standards on which the Internet and most enterprise networks are based. The IP suite includes not only lower-level specifications, such as TCP and IP, but specifications for such common applications as e-mail, terminal emulation, and file transfer. The most relevant of these to the factory floor though are IP itself, TCP, and UDP. These three protocols communicate in collaboration to form what is known as the Internet. This technology is moving automation forward, giving operators the ability to access and control plants remotely.

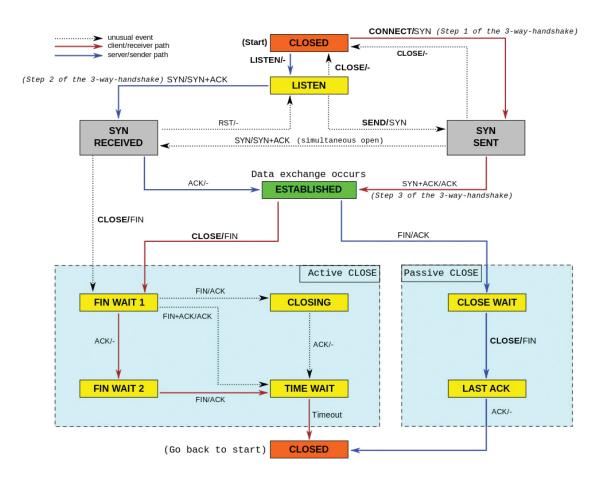


IP is the common and primary network (Layer 3) protocol in the Internet suite. IP represents the core of the Internet Protocol suite. It defines an address by which the network can transmit the packet from source to destination—even across LANs. This is opposed to the MAC addresses, which are used to network locally, within a LAN. IP has two primary responsibilities: providing connectionless, best-effort delivery of datagrams or packets through a network; and providing fragmentation and reassembly of datagrams to support data links with different maximum-transmission unit (MTU) sizes. More on IP address allocation and administration is found in the network management section.

Focus 4: TCP (Transmission Control Protocol)

TCP provides reliable, in-order delivery of packets between two devices. It relies upon IP. TCP applications establish connections between one another over which they send packets. TCP is a stateful protocol; it maintains the state after the packet is sent. TCP checks whether all packets have arrived and can request re-transmission if a packet is dropped, lost, or corrupted during transmission. Due to this overhead though, TCP is not always ideal for realtime applications. TCP has emerged as the dominant protocol used for the bulk of Internet connectivity. Its services break large data sets into individual packets, checking for and resending lost packets and reassembling packets into the correct sequence. But these additional services come at a cost in terms of additional data overhead, and delays called latency.

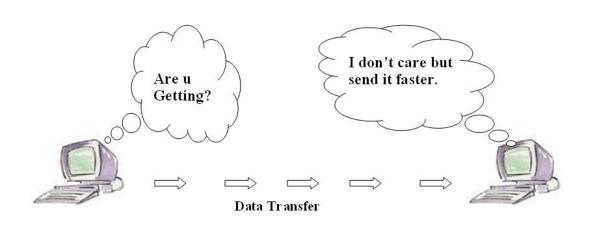
TCP is a connection-oriented protocol, which means a connection is established and maintained until the application programs at each end have finished exchanging messages. It determines how to break application data into packets that networks can deliver, sends packets to and accepts packets from the network layer, and manages flow control. It is meant to provide error-free data transmission. If any data packets get dropped or garbled, they will re-transmit until acknowledgment that all packets arrived successfully. This error-free feature requires a complicated routine. Below is a diagram showing the TCP routine.



Focus 5: UDP (User Datagram Protocol)

UDP is often used for real-time communications such as voice and I/O traffic. UDP also relies on IP. UDP does not guarantee delivery or the order of the packets, thus simplifying the protocol. It has much lower bandwidth overhead and latency. The applications would rather drop a packet than receive it late. UDP is considered a "stateless" protocol. It is compatible with packet broadcast (sending to all on local network) and multicasting (sending to all subscribers). UDP is an ideal protocol for network applications in which perceived latency is critical such as gaming, voice and video communications, which can suffer some data loss without adversely affecting perceived quality. In some cases, forward error correction techniques are used to improve audio and video quality in spite of some loss.

UDP can also be used in applications that require lossless data transmission when the application is configured to manage the process of retransmitting lost packets and correctly arranging received packets. This approach can help to improve the data transfer rate of large files compared with TCP. The main goal for UDP transmission is send it as fast as possible. The diagram below summarizes UDP transmission:



UDP

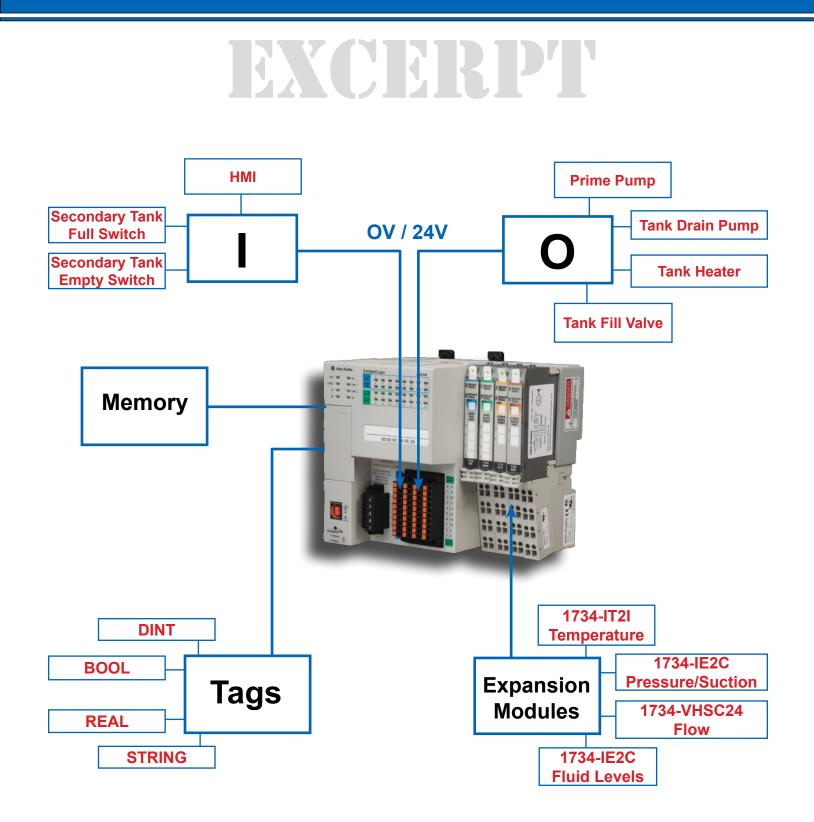


CHAPTER 1 - Introduction to Industrial Ethernet

154

Previous

Unit 4 - Programmable Automation Controller



Next

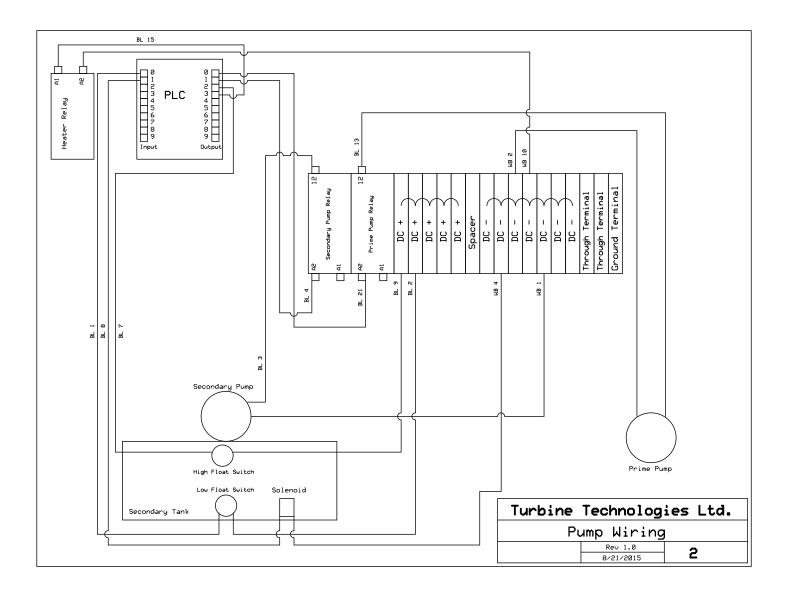
Chapter 2 - Embedded Input / Output

The PAC has 16 embedded inputs. Fluid*Mechatronix*[™] uses 2 inputs for the following devices:

- Drain Pump Float Switch
- Solenoid Valve Float Switch

The PAC has 16 embedded outputs. Fluid*Mechatronix*[™] uses 4 outputs for the following devices:

- Prime Pump
- Tank Heater
- Secondary Tank Drain Pump
- Secondary Tank Solenoid

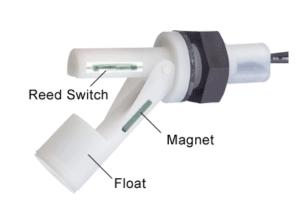


Focus 1: Drain Pump Float Switch

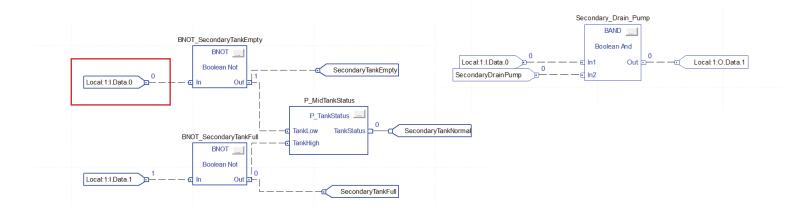
In order to fully understand the FluidMechatronix[™] tank levels, we must first learn how a float switch operates. The purpose of a float switch is to open or close a circuit as the level of a liquid rises or falls. Most float switches are "normally closed," meaning the two wires coming from the top of the switch complete a circuit when the float is at its low point, resting on its bottom clip (for example, when a tank is dry).

Horizontal Type:

To complete a circuit, float switches utilize a magnetic reed switch, which consists of two contacts sealed in a glass tube. When a magnet comes close to the two contacts, they become attracted to each other and touch, allowing current to pass through. When the magnet moves away, the contacts demagnetize and separate (breaking the circuit), as shown below.



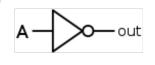
FluidMechatronix[™] monitors the status of the drain pump float switch on **Local:1:I.Data.0**. The float switch serves two purposes. First, to monitor the low level of the fluid in the tank. Second, to deenergize the drain pump coil in the event that the water levels cause the float to drop. The data is stored as a tag and is monitored through software. The data is stored as a boolean value (0 or 1) and is represented in the tag library by BOOL. Recall binary numbers from Chapter 3, the byte values for 0 and 1 would be 00000000 and 0000001 respectively. To fully understand the circuits below, we need to investigate the BNOT and BAND further.



BNOT and BAND are special types of logic used in Studio 5000. This logic can be represented with software (in our case), or integrated circuit (IC) chips. Traditionally, these gates were represented as NOT and AND gates.

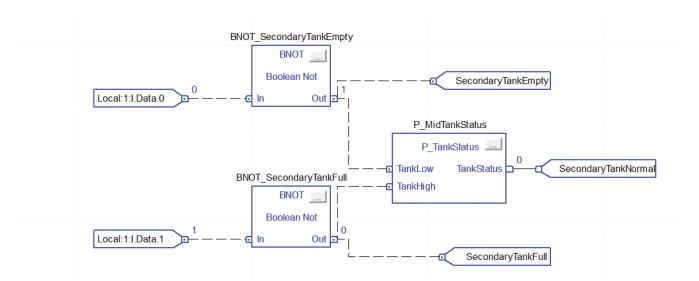
A NOT gate is the most basic type of gate. There are only 2 states for a NOT gate. When the input is 0 to a NOT gate, the output is NOT 0 so it has to be 1. Similalry, when the input to a NOT gate is 1, the output is NOT 1 so it is a 0. The table on the right shows the typical diagram and logic table for a NOT gate. The BNOT gate that is used in the Studio 5000 software is nothing more than a NOT gate that has the ability to use any data with a BOOL tag.

INPUT	OUTPUT
Α	NOT A
0	1
1	0



An AND gate is another fairly simple gate that has 4 states. This type of gate is slightly different because it has 2 channels of input. When A and B are both zero, the output of the AND gate is 0. When A is 0 and B is 1, the output of the AND gate is 0. When A is 1 and B is 0, the output of the AND gate is 0. Finally, when A is 1 and B is 1, the output of the AND gate is 1. Both A and B have to be 1 in order for the AND gate to output a 1. Any combinations that have 0 at the input, the output will always be 0 on an AND gate. The BAND gate that is used in the Studio 5000 software is nothing more than a AND gate that has the ability to use any data with a BOOL tag.

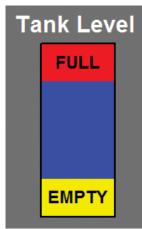
INF	TU	OUTPUT
Α	В	A AND B
0	0	0
0	1	0
1 0		0
1	1	1
A		_ _



Now we have enough information to fully understand the Drain Pump Float schematic used by Studio 5000. There are two states to the float switch:

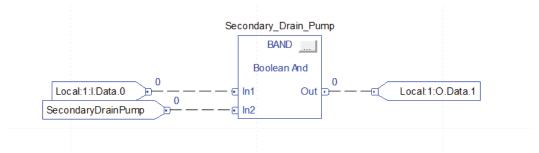
1. When the water level in the secondary tank is above the switch, the float switch is closed and the PAC will see the tag Local:1:I:Data.0 = 1. In this case the tank has enough water.

2. When the water level in the secondary tank is below the switch, the float switch is opened and the PAC will see the tag Local:1:I:Data.0 = 0. In this case the tank is empty.



We need a tank alarm to tell us when the secondary tank is empty. Looking at state number 2 above, the value is zero when the tank is empty. Since we want the alarm to be represented by a 1 (alarm on), we need to send the data through a BNOT to reverse the signals. The data is taken into the BNOT as a 0, so the output from the BNOT must be a 1. When SecondaryTankEmpty = 1 we can use this status to setup an alarm on the HMI screen.

When Local:1:I:Data.0 = 1, there is water in the tank and the output of the BNOT will read 0. So the tank empty alarm status will be 0 and the alarm is off. Alarm configurations with FactoryTalk will be covered in Unit 5. For now, we are only concerned about the data in Studio 5000.



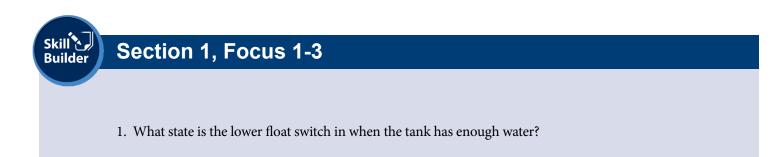
Now we can analyze the Drain Pump circuit that controls the pump. We want the operator to have pump control when there is enough water in the secondary tank. If there is isn't enough water in the tank, we want the PAC to control the pump and shut it off. The BAND gate is perfect for this situation because we require two elements to turn the pump on. We have water level indicated by the float switch tag on Local:1:I.Data.0. We have an operator button controlling the status of the tag SecondaryDrainPump. There are four states for this dual input BAND gate:

1. The float switch is open (0) and the operator switch is off (0). The output of the BAND is 0 and the pump stays off.

2. The float switch is open (0) and the operator switch is on (1). The operator wants to turn the pump on, but there is not water. The output of the BAND remains 0 and the pump stays off.

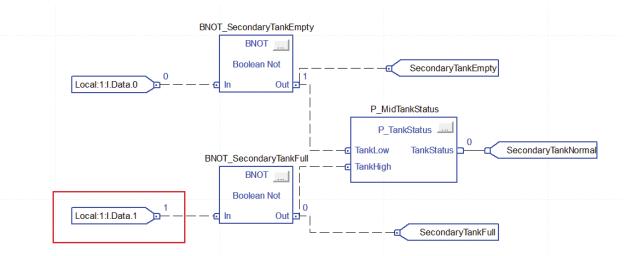
3. The float switch is closed (1) and the operator switch is off (0). There is enough for the pump to run, but the operator doesn't want the pump on. The output of the BAND is still 0 and the pump stays off.

4. The float switch is closed (1) and the operator switch is on (1). There is enough water in the tank and the operator wants to turn the pump on. The output of the BAND is 1 and the pump will turn on.



- 2. What type of logic can invert binary data?
- 3. Describe a situation where a AND gate could be used?

Focus 2: Solenoid Valve Float Switch



FLUID*Mechatronix*[™] monitors the status of the drain pump float switch on **Local:1:I.Data.1**. The float switch serves two purposes. First, to monitor the high level of the fluid in the tank. Second, to deenergize the solenoid valve coil in the event that the water level gets too high. The data is stored as a tag and is monitored through software as a BOOL (boolean). This float switch is oriented opposite the other switch, so when the float is up the switch is open. There are two states to the float switch:

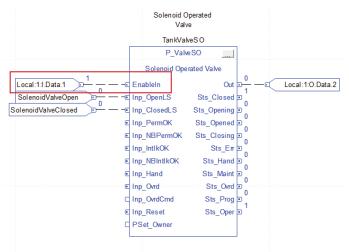
1. When the water level in the secondary tank is above the switch, the float switch is opened and the PAC will see the tag **Local:1:I:Data.1 = 0**. In this case the tank has too much water.

2. When the water level in the secondary tank is below the switch, the float switch is closed and the PAC will see the tag **Local:1:I:Data.0 = 1**. In this case the water in the tank is below the switch.

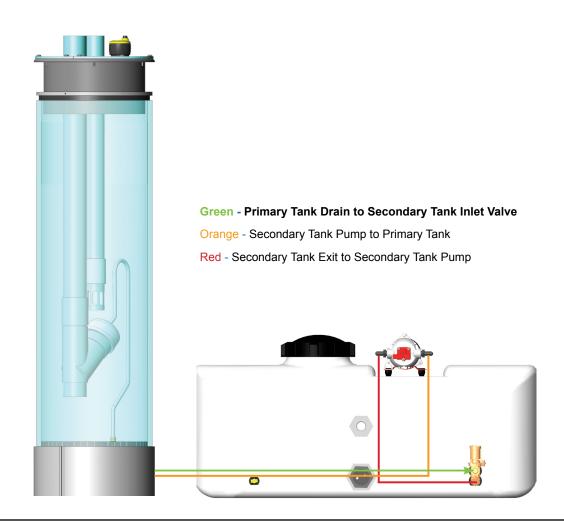
We need a tank alarm to tell us when the secondary tank has too much water. Looking at state number 1 above, the value is zero when the tank has too much water. Since we want the alarm to be represented by a 1 (alarm on), we need to send the data through a BNOT to reverse the signals. The data is taken into the BNOT as a 0, so the output from the BNOT must be a 1. When SecondaryTankFull = 1 we can use this status to setup an alarm on the HMI screen.

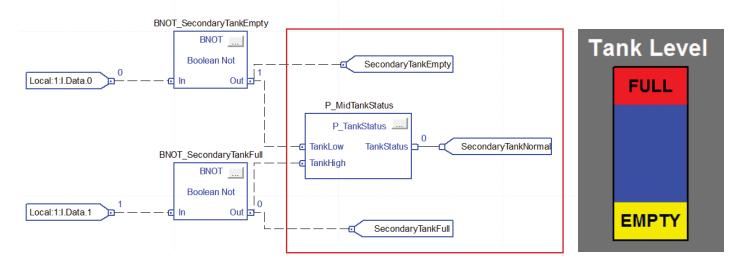
When Local:1:I:Data.0 = 1, there is water in the tank and the output of the BNOT will read 0. So the tank full alarm status will be 0 and the alarm is off. Alarm configurations with FactoryTalk will be covered in Unit 5. For now, we are only concerned about the data in Studio 5000.

Now we can analyze the Solenoid Valve circuit that controls the solenoid. In this case, we don't need to use a BAND gate because the valve is controller with an Add On Instruction (AOI). The AOI operates like a logic gate, but also offers many extra features like EnableIn. Recall the state where **Local:1:I:Data.0 = 1.** The water level in the tank is below the full switch, so the solenoid valve can operate. We can send this tag value directly into the EnableIn on the AOI. When the value is 1, it allows the AOI solenoid control. The operator will request control of the valve from the HMI and that control will be permitted because the feature is enabled. When there is too much water and the tank is full, the value



will be 0. When the EnableIn status of the AOI is 0, the AOI is disabled and the instruction will not execute. The operator can request valve control with the HMI, but the control is not allowed because the AOI is disabled. The AOI EnableIn simplifies the programming to enable and disable the solenoid.





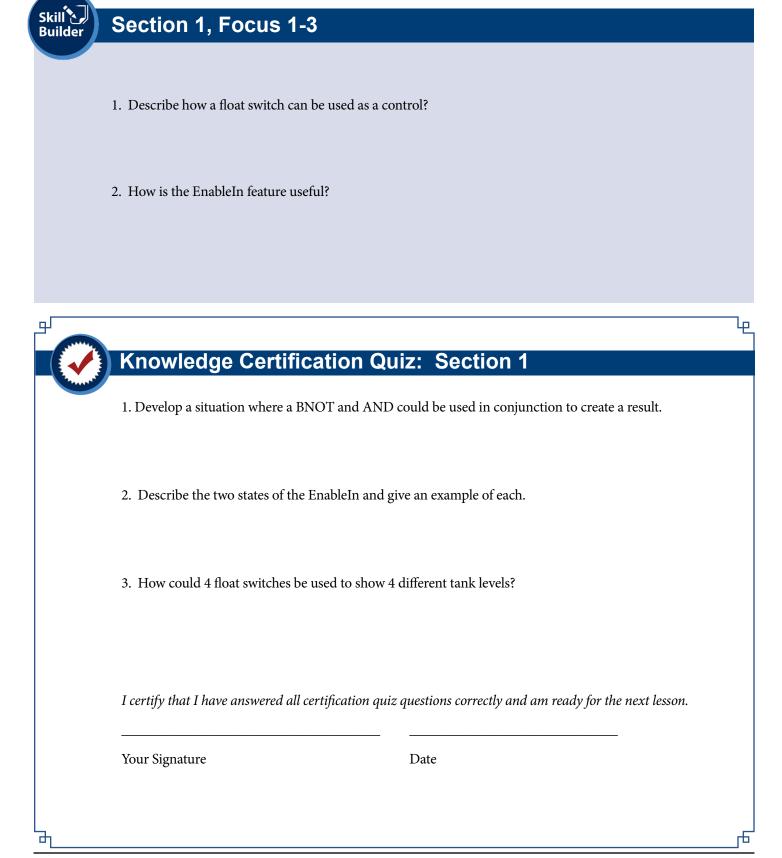
Now we can analyze the P_TankStatus circuit that controls the status of the blue section on the tank level above. There is a custom AOI, built to control the fluid visibility on the HMI. This custom add-on was programmed with Structured Text. We created 2 boolean tags for low and high, and one INT tag for the tank status. The text below represents the actual programming that runs the mid tank status. The blue indicator for mid tank level has 3 different output states. The structured text code below shows the states of the TankStatus:

If TankLow & NOT TankHigh Then TankStatus := 0; EIsIf NOT TankLow & TankHigh Then TankStatus := 2; EIsIf NOT TankLow & NOT TankHigh Then TankStatus := 1; End_IF;

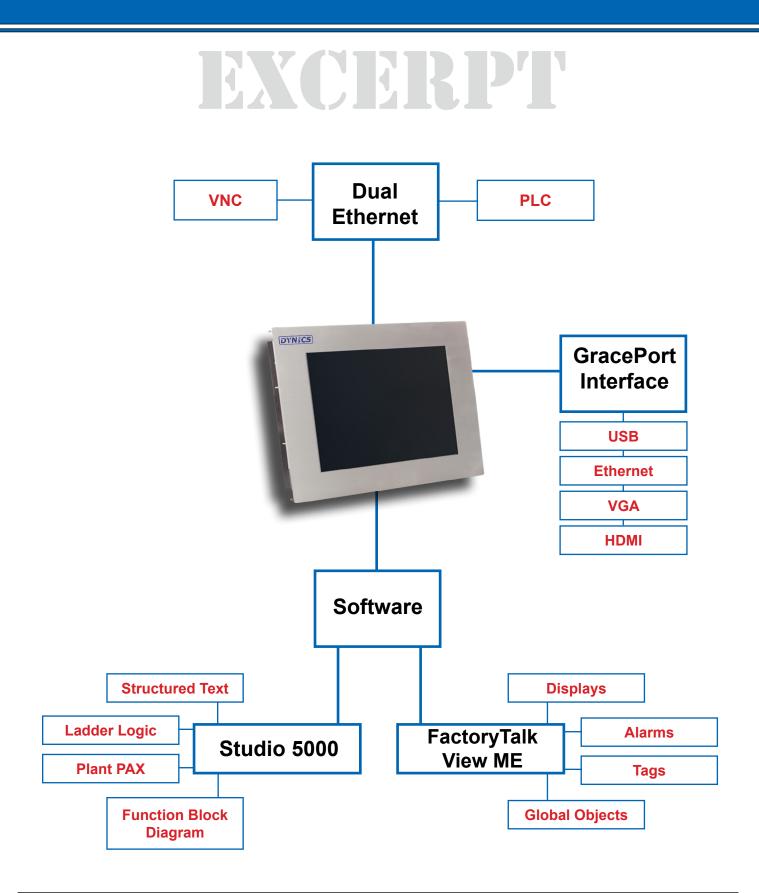
For the first line of code, TankLow = 1 and TankHigh = 0. We want the output for this case to be 0. This result is used to remove the blue and red area of the tank level on the HMI. We don't want any blue or red to show because the tank is empty. We will cover the HMI alarm connections in Unit 5.

For the second line of code, TankLow = 0 and TankHigh = 1. We want the output for this case to be 2. This result will display the blue section on the HMI, but not show the text "OK" in that section. This is because we will also be displaying the full alarm at the same time.

For the third line of code, TankLow = 0 and Tank High = 0. We want the output for this case to show the blue section with the text "OK" on the screen. This is very similar to the second line of code, but we will not have the full alarm on the screen.



Unit 5 - Human Machine Interface



Focus 1: Creating a New Project

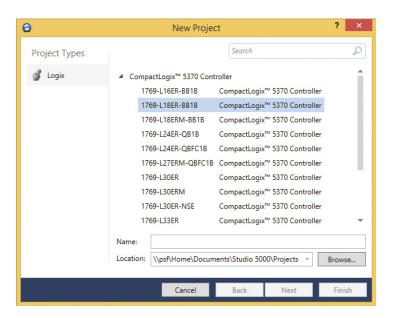
Double click the Studio 5000 icon in the dock



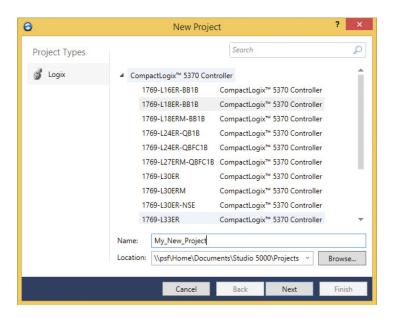
The Studio 5000 splash screen with open. You will see FLUID*Mechatronix*TM is a recent project. If you wish to open that project for viewing, click on it. Otherwise, click on **New Project**.



A new window will open. Click on Logix and navigate to the 1769-L18ER-BB1B controller.



Choose a name for the new project and click Next.



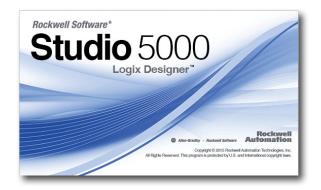
A new window will open with the New Project details. We can see the current software revision is v26.

0	New Project	?	×
1769-L18ER-BB1 My_New_Project	B CompactLogix™ 5370 Controller		
Revision: Expansion I/O:	26		
Security Authority:	No Protection v Use only the selected Security Authority for authentication and authorization		
Description:			
	Cancel Back Next	Fini	ish

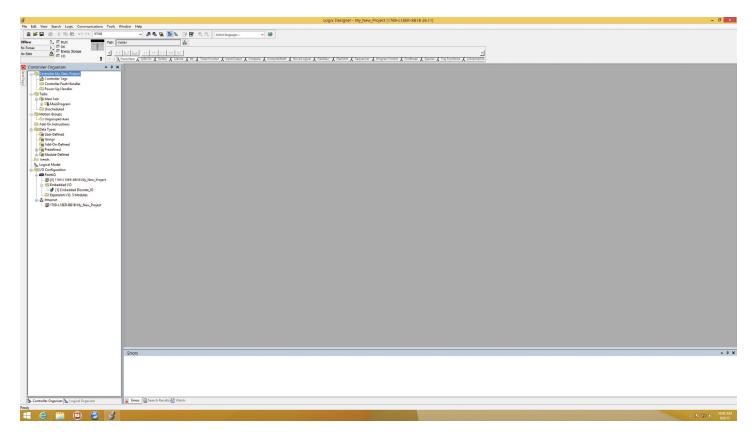
Select the proper number of modules for the system. In this case, we have 3 expansion modules installed. Select **3 Modules**. If security is desired, select the type of security and click **Finish**. Note: the software already knows the total number of modules allowed for expansion is 8 modules.

0		New Proje	ct		?	×
1769-L18ER-BB1 My_New_Project	3 CompactLogi	ix™ 5370 Controlle	r			
Revision:	26 ~					
Expansion I/O: Security Authority: Description:	1 Module 2 Modules 3 Modules	en online, if the mod d in the project, unex tting must match the selected Security Aut	pected control r actual number	may occur. The of modules.		
		Cancel	Back	Next	Fini	sh

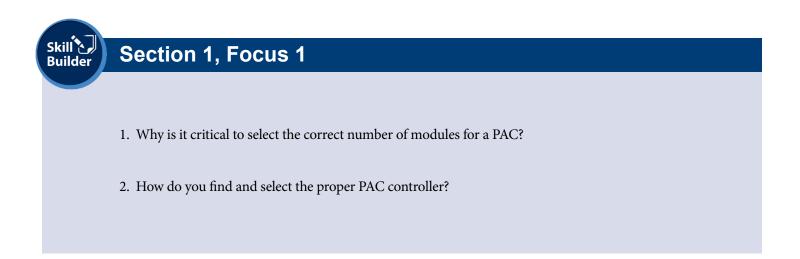
The new project will begin loading and show the following screen.



The new project will open and display the main window.

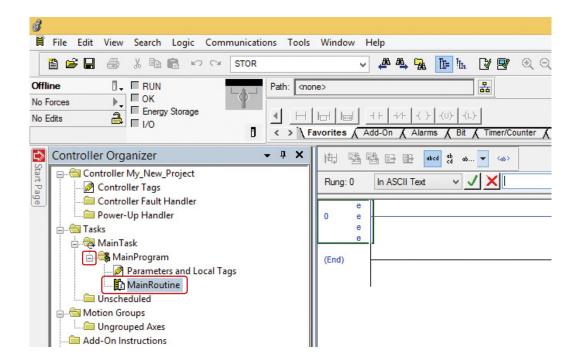


The **Organizer Window** appears on the left side of the Studio 5000 window. There is a folder highlighted for Controller My_New_Project. There is no I/O, tag database, or logic associated with the controller.

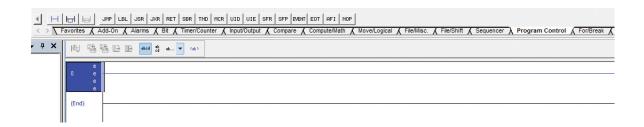


Focus 2: Ladder Logic

The next step in creating a new project is setting up the main routine. The main routine executes the ladder logic for the main ladder. Click the (+) sign to open the **MainProgram** and show the **MainRoutine**.



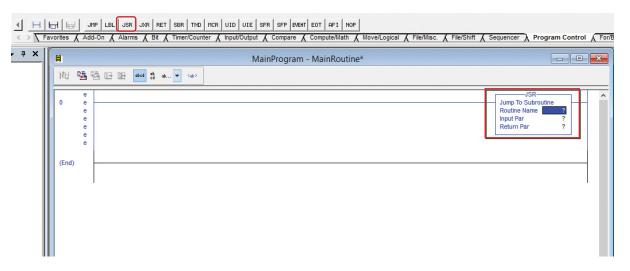
Find the tab near the top that says **Program Control**. Click on the tab and a list of program controls will show above the tabs.



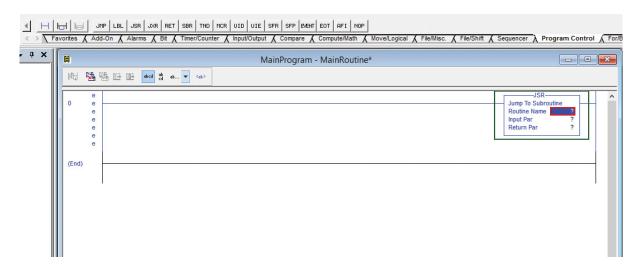
Find the tab near the top that says **Program Control**. Click on the tab, and a list of program controls will show above the tabs. We are

		JXR RET SBR TND		AFI NOP pute/Math 🔏 Move/Logi	ical 🔏 File/Misc. 🔏 F	ile/Shift 🔏 Sequencer	Program Control	For/Break
- 4 ×	e 0 e e (End)	i di ok V (a)						

Click on the JSR (Jump To Subroutine) to create a sub routine in the main routine.



Right-click on the question mark for the Routine Name. A window will appear for creating a new sub routine.



Name the new sub routine and select the drop-down for type. Select Function Block Diagram for the sub routine. Click OK twice to accept the settings.

Name:	SubRoutine		OK
Description:		^	Cancel
		\sim	
Туре:	Function Block Diagram	~	Help
n Program:	Ladder Diagram		
	Function Block Diagram		
	Structured Text		

Click on the JSR (Jump To Subroutine) to create a sub routine in the main routine.

Ħ	MainProgram - MainRoutine*	
問		
0	e e e e e e	JSR Autoroutine Routine Name: SubRoutine Input Par ? Return Par ?
(End		

Right-click on the question mark for the Parameter. A window will appear for editing the parameter. Click remove instruction parameter. Do this for both parameters.

Par	2	
	New Tag	
*	Cut Instruction	Ctrl+X
	Copy Instruction	Ctrl+C
8	<u>P</u> aste	Ctrl+V
	Delete Instruction	Del
	Add Ladder Element	Alt+Ins
	Edit Main Operand Description	Ctrl+D
	Add Input Parameter	
	Add Return Parameter	
	Remove Instruction Parameter	
	Save Instruction Defaults	
	Clear Instruction Defaults	
	R <u>e</u> move Force	
	<u>G</u> o To	Ctrl+G

Before removing the unused parameters, the routine has rung errors. Errors are denoted by the letter "e" next to the rung. See below:

Ħ	MainProgram - MainRoutine*	- • •
問	S	
0 (End)		JSR- Jump To Subroutine Routine Name SubRoutine Input Par ? Return Par ?

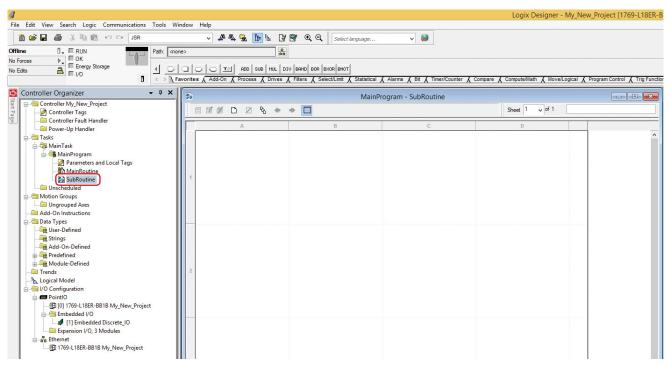
After removing the parameters the routine is error free.

Image:	×	MainProgram - MainRoutine*
0 JSR Jump To Subroutine Routine Name SubRoutine		
(End)	^	0 Jump To Subroutine Routine Name SubRoutine

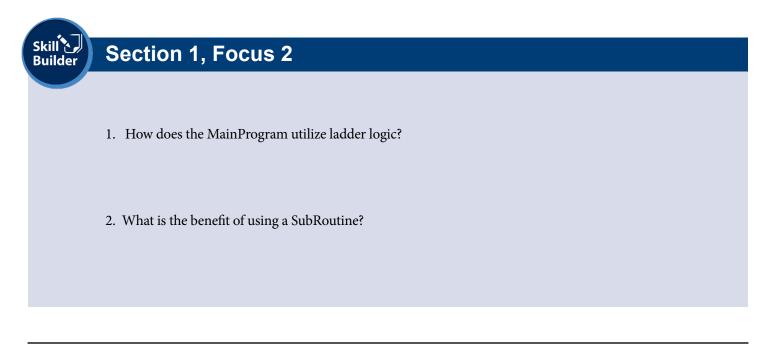
To verify the programming the for errors, click the verify controller button as shown below. Monitor the errors in the **Errors** windows.

3	Logix I	Designer - My_New_Project [1769-L18ER-B
File Edit View Search Logic Communications Tools V	indow Help	
🗎 🗃 🛃 🏯 🐇 🖻 💼 🕫 🖙 🕞 JSR	🗸 🏘 强 📴 🌆 诸 🕼 🧟 🔍 Select language	
Offline I + I RUN No Forces OK	ne>	
No Edits	H J+P LBL JSR JSR ASH SBR SBR THO HCR UID UIE SFR SFP BAHT BOT AFI HOP Favories & Add-On & Alarms & BR & TmeriCounter & Input/Output & Compare & Compute/Nath & Movel.cgical & FielMisc. & FielShift	🗴 Sequencer 👌 Program Control 🖌 For/Break 📈
Controller Organizer • # ×	MainProgram - MainRoutine	- • 💌
Controller Tags		
Generation Service Control Co	0	Jump To Subroutine Routine Name SubRoutine
Parameters and Local Tags MainRoutine SubRoutine Unscheduled	(End)	
Motion Groups Motion	Errors Verifying Controller Verifying routine 'MainRoutine' of program 'MainProgram' Verifying program connections Complete - 0 error(s), 0 warning(s)	* ^ *

The new program is now set to run correctly under the new sub routine. This allows the use of the customized Function Block Programming for using Add-On Instructions (AOIs).



For detailed instruction on Ladder Logic, see Unit 8.

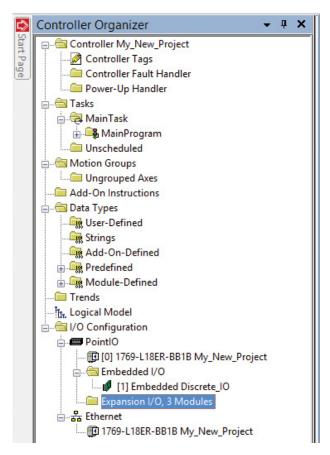


Focus 3: Adding Expansion Modules

When creating a new project, the number of expansion modules was pre-determined. If that number is different than what the project shows, edit the properties of the controller. Right click on the controller and select properties. The controller properties window will show up. Update the correct number of expansion modules

Start Page	Controller Organizer 🔹 🖣 🗙	3	Contro	ller Properties	- My_New_f	Project			
art	Controller My_New_Project	Newslatte Mer		Internet Protoc	al Deat Coord	Second Second Second		Constant	Alexa Las
Pag	Controller Tags	Nonvolatile Mem General				-		Security	Alarm Log
	Controller Fault Handler	General	Major Faults	Minor Faults	Date/Time	Advanced	SFC E	xecution	Project
	Power-Up Handler	Vendor:	Allen-Bradley						
	🖶 🖓 Tasks						_		
	🖨 🝓 MainTask	Type:	1769-L18ER-BB	1B CompactLogix"	5370 Controller			Change Co	ontroller
	🗄 🛱 MainProgram	Revision:	26.11						
	Unscheduled	TIC VISION.	20.11						
	🖕 🔚 Motion Groups	Name:	My New Project	x .					
	Ungrouped Axes								
	Add-On Instructions	Description:					^		
	🚊 🖓 🔂 Data Types								
							\sim		
		Expansion I/O:	3 Modules		R: When online.	if the modules pr	resent do		
		Expansion i/o.	3 Wodules	not mate	ch the modules sp	pecified in the pro	oject,		
	🕀 🚋 Predefined	Slot:	0		cted control may				
	🖶 🚋 Module-Defined			setting i	nusi matori trie at	ciual number of fi	nouules.		
	Trends								
	🗄 🔚 I/O Configuration								
	PointlO								
	[0] 1769-L18ER-BB1B My_New_Project								
	🖃								
	[1] Embedded Discrete_IO								
	Expansion I/O, 3 Modules								
	1769-L18ER-BB1B My_New_Project								
					OK	Cancel	l A	Apply	Help

Next, right-click on the Expansion I/O folder and select



A new window will be displayed for **Select Module Type**

	Jearch Text Tor	Module Type Cle	ar Filters		Hide Filters 🛠
•	M	lodule Type Category Filters		Module Type Ver	ndor Filters
>>>>	Analog Digital Other Specialty		radley ced Micro Controls Inc. (um Controls, Inc.	(AMCI)	
Cat	alog Number	Description	Vendor	Category	^
88	1734-232ASC	RS232 ASCII Interface Module	Allen-Bradley	Specialty	
	1734-485ASC	RS485 ASCII Module	Allen-Bradley	Specialty	
	1734-8CFG	8 Point 10V-28V DC Self-Configuring	Allen-Bradley	Digital	
	1734-8CFGDLX	8 Point 10V-28V DC Self-Configuring	Allen-Bradley	Digital	
		2 Point 120V AC Input	Allen-Bradley	Digital	
	1734-IA2 1734-IA4	4 Point 120V AC Input	Allen-Bradley	Digital	

The FLUIDMechatronixTM system uses 3 expansion modules. Recall that the location of modules were the following:

Module 1: Embedded I/O Module 2: IE4C Module 3: IT2I Module 4: VHSC24

Use the Select Module Type window to search for modules 2-4. Start with module 2, by searchine IE4C.

Search Text for	Module Type Cle	ear Filters	÷			Hide Filters	*
M	lodule Type Category Filters			Module Type \	/endor Filters	0	
Analog Digital Other Specialty		>>>	Advanced	Micro Controls Inc	: (AMCI)		
log Number	Description	Vendor	2	Category			^
734-232ASC	RS232 ASCII Interface Module	Allen-Bra	adley	Specialty			
734-485ASC	RS485 ASCII Module	Allen-Bra	adley	Specialty			
734-8CFG	8 Point 10V-28V DC Self-Configuring	Allen-Bra	adley	Digital			
				Digital			
			-	Digital			
734-IA4	4 Point 120V AC Input	Allen-Bra	adley	Digital			~
	M Analog Digital Other Specialty og Number 734-232ASC 734-825ASC 734-82FG 734-8CFG 734-8CFGDLX 734-1A2	Module Type Category Filters Analog Digital Other Speciality log Number Description 734-232ASC RS232 ASCII Interface Module 734-485ASC RS485 ASCII Module 734-8CFG 8 Point 10V-28V DC Self-Configuring 734-8CFGDLX 8 Point 10V-28V DC Self-Configuring 734-IA2 2 Point 120V AC Input	Module Type Category Filters Analog Digital Other Specialty Og Number Description Vendor 734-232ASC RS232 ASCII Interface Module Allen-Br 734-485ASC RS485 ASCII Module Allen-Br 734-8CFG 8 Point 10V-28V DC Self-Configuring 734-8CFGDLX 8 Point 10V-28V DC Self-Configuring Allen-Br 734-1A2 2 Point 120V AC Input	Module Type Category Filters Analog Digital Other Speciality Image: Speciality <	Module Type Category Filters Image: Category Filters Analog Image: Category Filters Digital Image: Category Filters Other Advanced Micro Controls Inc. Speciality Image: Category Filters log Number Description Og Number Description Vendor Category 734-232ASC RS232 ASCII Interface Module Allen-Bradley Speciality 734-485ASC RS485 ASCII Module 734-8CFG 8 Point 10V-28V DC Self-Configuring 734-8CFGDLX 8 Point 10V-28V DC Self-Configuring 734-1A2 2 Point 120V AC Input	Module Type Category Filters ✓ Module Type Vendor Filters Analog Digital Other ✓ Allen-Bradley ✓ Speciality ✓ Advanced Micro Controls Inc. (AMCI) ✓ Speciality ✓ log Number Description Vendor Category 734-232ASC RS232 ASCII Interface Module Allen-Bradley Speciality 734-485ASC RS485 ASCII Module Allen-Bradley Speciality 734-8CFG 8 Point 10V-28V DC Self-Configuring Allen-Bradley Digital 734-8CFGDLX 8 Point 10V-28V DC Self-Configuring Allen-Bradley Digital 734-1A2 2 Point 120V AC Input Allen-Bradley Digital	Module Type Category Filters Image: Category Filters Analog Image: Category Filters Digital Allen-Bradley Other Advanced Micro Controls Inc. (AMCI) Specialty Spectrum Controls, Inc. og Number Description Vendor Category 734-232ASC RS232 ASCII Interface Module Allen-Bradley Specialty 734-485ASC RS485 ASCII Module 734-8CFGDLX 8 Point 10V-28V DC Self-Configuring. Allen-Bradley Digital 734-1A2 2 Point 120V AC Input

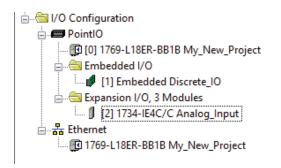
The search is narrowed down to 2 modules of this type. Select the correct module and click **create** to add it to the program.

E4C	;		Clear Filters			Hide Filters
•	N	Module Type Category Filters		Module Typ	pe Vendor Filters	5
> > > >	Analog Digital Other Specialty		Advar	Bradley aced Micro Controls rum Controls, Inc.	Inc. (AMCI)	
Cat	alog Number	Description	Vendor	Category		
	1734-IE4C 1734sc-IE4CH	4 Channel Analog Current Input 4-Channel HART Analog Input	Allen-Bradley Spectrum Cont	Analog ro Analog		

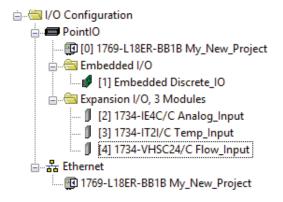
A new window will pop up allowing you to configure the module. At this point, all we need to do is give the module a name and click OK.

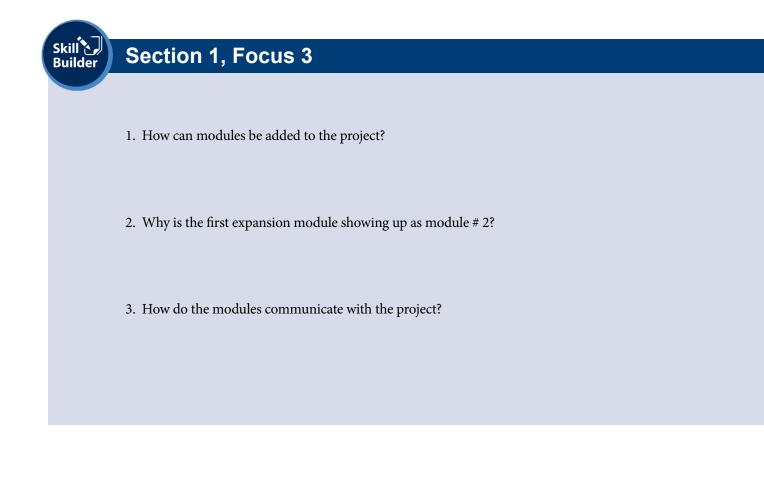
General* Cor	nnection	Module Info	Configuration	Alarm Configu	ration	Calibration			
Type: Vendor:		IE4C 4 Chann Bradley	el Analog Curre	ent Input					
Parent:	Local								
Name:	Analo	og_Input					Slot:	2	¥
Description:					~				
Module De	finition								
Series:		С							
Revision:		3.001							
Electronic	Keying:	Compatible M	lodule						
Connectio	n	Data							
Data Form	nat	Integer							
				Change					

The module configuration should show the module in the correct location with proper naming.



Repeat the steps to add the IT2I and VHSC24 Module. Note: It is important that the module specified is in the proper location on the slot. Improper slot specification can lead to dangerous outcomes during runtime. The Module list should show the following:

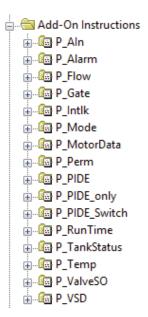




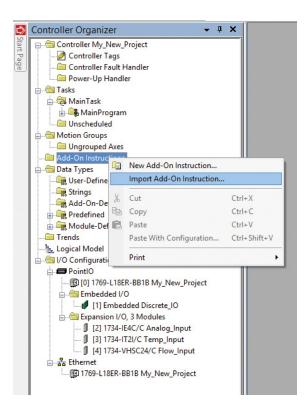
Focus 4: Function Block Diagram

The main purpose of the FBD programming is to simplify logic programming. FBDs not only add simplicity but also provide a graphical diagram of the circuit schematic. FBD instructions sometimes have both Structured Text and Ladder Logic programming. Many of the AOIs used in Studio 5000 are based on FBD programming. This makes complicated programming appear simple, saving companies money by not paying for programmers. FLUID*Mechatronix*[™] utilizes the FBD instructions for most of the programming in the system.

Recall My_New_Project where we created a sub routine under the main routine with FBD programming. Now we can begin populate the first page of My_New_Project with FBD instructions.



In order to use Add-On Instructions, they must be added to the project. Right click on the Add-On Instructions folder and select **Import Add-On Instruction**.



Locate the PlantPAX add-on instructions in the FLUID*Mechatronix*[™] Support Documents. PlantPAX v3.1 is included with the system. The latest version can be downloaded from Rockwell Automation support. Once you locate the PlantPAX folder, browse to Files > Process Objects Library > Process Add-On Instructions > P_AIn_3_1-02_AOI.L5X. An import window will open, change settings if desired and click OK.

•	Import Cor	nfiguration - P_AIn_3_1-02_AOI.L	5X	×
Find: Find Within: Final Name	✓ ▲ ▲	Find/Replace		
Import Content:				
- Add-On Instructions	Configure Add-Or	Instruction Properties		
P_AIn Parameters and Local Tags	Import Name:	P_AIn		
Routines	Operation:	Create 🗸 🗸		
		(i) References will be imported as configured in the References folders		
Add-On Instructions	Final Name:	P_AIn 🗸	Properties	
. 🔯 Errors/Warnings	Description:	Analog Input		
		~		
	Revision:	v3.102 Release		
	Revision Note:	See V3.1-xx Release Notes for details		
	Vendor:	Rockwell Automation		
< >				
			OK Cancel Help	
Ready				

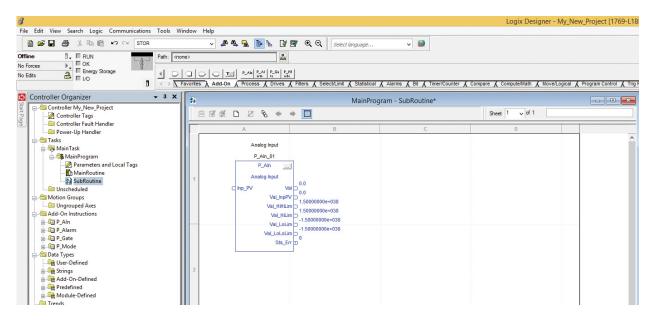
The AOI and other required components are imported. The new imports are shown below.

🚍 🖶 🖂 Add-On Instructions
🕀 💼 P_Aln
🕀 🕼 P_Alarm
🕀 🕼 P_Gate
🗄 📠 P_Mode

After importing the AOI, the Add-On tab will show the new modules. The new FBD modules can now be selected from the toolbar.

3											
File Edit View	Search	Logic	Communications	Tools	Window	Help					
1	🖨 %	Þa 🖪	STOR			~	≜ 86. 886 _→	9	TE= Its	29	Q
Offline	0. E F	RUN	L_A	Path:	<none></none>					몲	
No Forces	▶ = 5		- P -				r			<u> </u>	
No Edits	2	Energy Sto	rage				그리	P_AIn	P_AI P_(arm te	Ga P_M ode	
		0	٥	< >	\ Favorite	s) Ad	ld-On 🔏	Proc	ess 🖌 Dr	ives 🖌 Fi	iters /

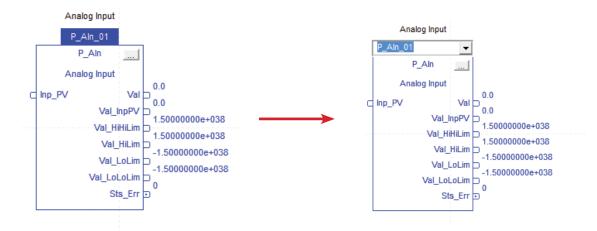
Select the P_AIn and the AOI will be placed in the SubRoutine sheet 1.



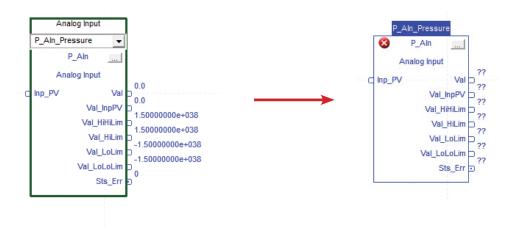
Rename the sheet as Analog Input.

4	MainProgram - SubRoutine*		- • ×
E 🛙 🖉 🗅 🗵 🗞 🚸 🗰 🔲		Sheet 1 v of 1 Analog Input	
A	ВС	D	
Analog Input P_AIn_01 P_AIn Analog Input I D_ D_ V Val D_ D_ D	++038 e+038		

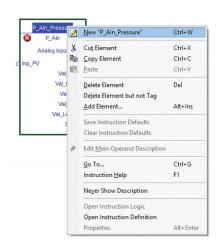
Double-click on the title for the AOI. This will allow you to edit the tag name of the AOI.



Change the instruction tag name to P_AIn_Pressure and the tag name will change. There will be an X denoting that there is a problem with the tag name. The tag is still stored in the local tags for the MainRoutine.



Next, we want to make sure that the tag is stored in the controller. Right-click on the tag and select **New "P_AIn_Pressure"**



The New Tag window will open. Click on **Create** to make a new tag under the Scope of My_New_ Project. The new tag will be created as a Controller Tag, not a local MainRoutine tag. Storing tags in this manner will help the communication process between Studio 5000 programming and FactoryTalk View ME.

		New Tag	×
P_AIn_Pressure	Name:	P_AIn_Pressure	Create 🛛 🕶
Analog Input	Description:	·	Cancel
C Inp_PV Val 2 ?? Val_InpPV 2 ?? Val_HiHiLim 2		~	Help
Val_HiLim D Yal_LoLim D	Usage:	<controller></controller>	
Val_LoLoLim D	Туре:	Base v Connection	
Sts_Err D	Alias For:		
	Data Type:	P_Aln	
	Parameter Connection:	<u></u>	
	Scope:	My_New_Project	
	External Access:	Read/Write 🗸	
	Style:	×	
	Constant		
	Sequencir	ng	
	Open Con		
	Open Para	ameter Connections	

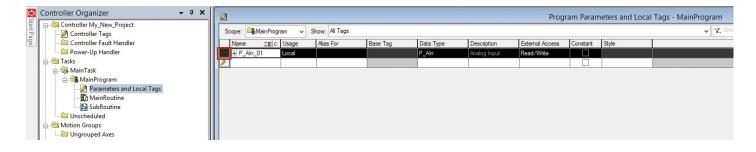
Double-Click on Controller Tags. You will see the new tag P_AIn_Pressure in the list.

Controller Organizer - 4 ×			Controller Tags	- My_New_Project(controller)
Controller My, New Project Page Controller Tags Controller Fault Handler	Scope: ∰My_New_Project ∨ Show:	All Tags		V T. Ente.
Controller Fault Handler	Name <u>=</u> ∎ △ Alias For	Base Tag Data Type Descript	tion External Access Constant Style	
Power-Up Handler	+ Local:1:C	AB:Embedded_Di	Read/Write	
🖕 🗠 🔄 Tasks	+-Local:1:I	AB:Embedded_Di	Read/Write	
🚔 🚭 MainTask	+-Local:1:0	AB:Embedded_Di	Read/Write	
🖻 🚔 MainProgram	+ Local:2:C	AB:1734_IE4:C:0	Read/Write	
Parameters and Local Tags	+ Local:2:1	AB:1734_IE4:I:0	Read/Write	
- 🛅 MainRoutine	+ Local:3:C	AB:1734_IT2I:C:0	Read/Write	
-B SubRoutine	+ Local:3:	AB:1734 IT2I:I:0	Read/Write	
Unscheduled	+ Local:4:C	AB:1734_VHSC:C:0	Read/Write	
😑 🔄 Motion Groups	+ Local:4:1	AB:1734_VHSC:I:0	Read/Write	
Ungrouped Axes	+ Local:4:0	AB:1734_VHSC:0:0	Read/Write	
Add-On Instructions	+-P_AIn_Pressure	P_AIn Analog I		
i ⊕ i i i i i i i i i i i i i i i i i				
⊕í@ P_Alarm				
P Mode				

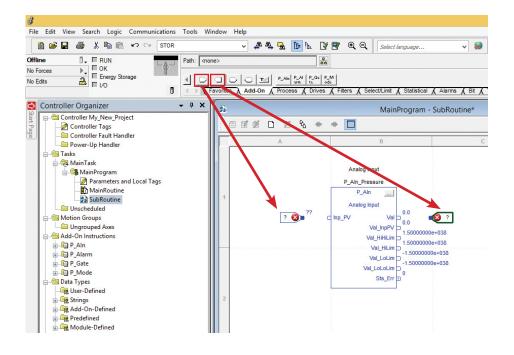
Next delete the local tag stored in the MainProgram > Parameters and Local Tags.

Controller Organizer - T ×	2						Prog	ram Parar	neters and Local	Tags - MainProgram
Controller My_New_Project	Scope: MainProgra	am 🗸 S	how: All Tags							v 7.
🖲 🔚 Controller Fault Handler	Name 📰 🛆	Usage	Alias For	Base Tag	Data Type	Description	External Access	Constant	Style	
Power-Up Handler	+-P_Aln_01	Local			P_Aln	Analog Input	Read/Write			
👝 🔄 Tasks	>									
A main Task A main Task A main Program A main Program D Main Routine D Main Routine D Unscheduled Motion Groups Langung Axes										

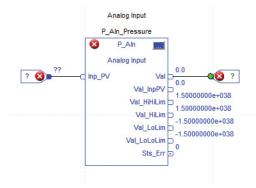
Click on the left square next to the tag that needs to be deleted. The whole tag will be highlighted in black. Right-click on the square and select Delete.



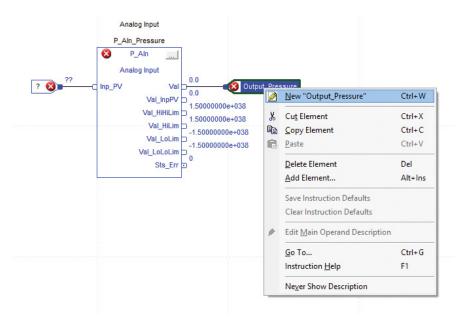
Go back to the SubRoutine window. Drag an input and output connection to the Analog Input page.



Wire the input and output of the P_AIn_Pressure FBD. Click on the nodes and select the desired location to wire the tag to.



Rename the output tag to Output_Pressure. Create a new tag in the scope of My_New_Project and select **Create**. The red X next to Output_Pressure will go away.

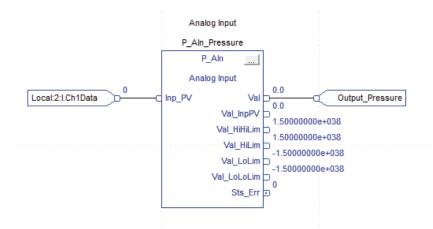


Next we need to wire the input to the proper channel on the 1734-IE4C. Recall that the pressure tranducer was connected to.

Wire the input and output of the P_AIn_Pressure FBD. Click on the nodes and select the desired location to wire the tag to. Locate the **Local:2:I.Ch1Data** tag and select it to connect the tag to the Inp_PV.

Analog Input P_AIn_Pressure P_AIn Analog Input Inp_PV Val O.0 Output_P	Analog Input P_AIn_Pressure P_AIn Analog Input Analog Input Local:2:I.Ch1Data, PV Val 0.0 Output_F	Pressu
Y. Enter Name Riter V Show: All Tags V	T. Enter Name Riter V Show: All Tags V	
Name Data Type Usage Descripti ▲ ●Local:1:C AB:Embedded <controller> ●Local:1:I AB:Embedded <controller> ●Local:1:O AB:Embedded <controller> ●Local:2:C AB:1734_E4: <controller> ●Local:2:I AB:1734_E4:I:O <controller> ●Local:2:I ●Local:2:I</controller></controller></controller></controller></controller></controller></controller></controller></controller></controller></controller></controller>	Name Image Data Type Usage Descripti Image: Description of the state of the s	
P Show controller tags Show MainProgram tags now parameters from other program: inone>	Show controller tags Show MainProgram tags Show parameters from other program: </td <td></td>	

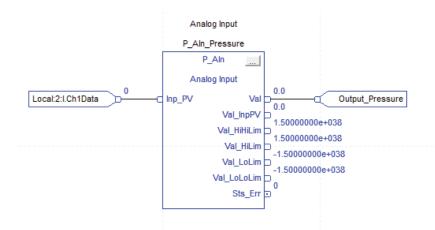
The red X error will go away and the P_AIn_Pressure FBD is wired correctly.



Wire the input and output of the P_AIn_Pressure FBD. Click on the nodes and select the desired location to wire the tag to. Locate the **Local:2:I.Ch1Data** tag and select it to connect the tag to the Inp_PV.

Analog Input P_AIn_Pressure P_AIn Analog Input Analog Input O.0 O.0	Output_Pressure	Analog Input P_AIn_Pressure P_AIn Analog Input pcal:2:I.Ch1DataPV Val	Pressu
y. Enter Name Filter v Show: All Tags	~	T. Enter Name Filter V Show: All Tags V	
Name Data Type Usage D →-Local:1:C AB:Embedded <controller> →-Local:1:I AB:Embedded <controller> →-Local:1:O AB:Embedded <controller> →-Local:2:C AB:1734_E4: <controller> →-Local:2:I AB:1734_E4:L0 <controller></controller></controller></controller></controller></controller>		Name Data Type Usage Descripti □-Local:2:1 AB:1734_IE4::0 <controller> □-Local:2:1.Fault DINT □-Local:2:1.Ch0Data INT □-Local:2:1.Ch2Data INT</controller>	
Show controller tags Show MainProgram tags now parameters from other program: none>	s] Show controller tags] Show MainProgram tags now parameters from other program: none> v	

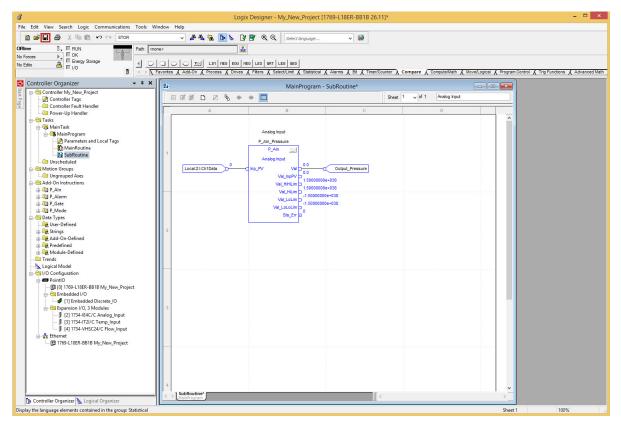
The red X error will go away and the P_AIn_Pressure FBD is wired correctly.



Previous

Section 1: Studio 5000 Logix Designer Programming Environment

Save My_New_Project by clicking the save button. Your new project should look like the diagram below:





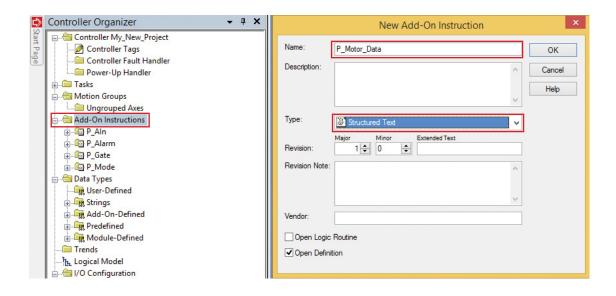
Section 1, Focus 4

- 1. How does a function block diagram simplify programming?
- 2. Describe how a function block diagram communicates tag data with a PAC.
- 3. Does the programmer need to know ladder logic to use the function block diagrams?

Focus 5: Structured Text

Structured Text programming is another feature of the Logix Designer programming environment. It allows the programmer addional ways to acomplish a process task. Structured text can also be used to build custom AOIs that can be used on the FBD.

Next we can create a new AOI for the motor data. Right-click on Add-On Instructions to add a new instruction. Name the instruction **P_Motor_Data**. Be sure to select the **Structured Text** for the Type of instruction.



The new AOI definition window will pop up. Here we can add a description and enter the local tags that will be needed for the instructions.

General Paran	neters Local Tags Scan Modes	Signature Change History Help	
Name:	P_Motor_Data		
Description:		~	
		~	
Type:	Structured Text	Change Type	
Revision:	Major Minor Extended	Text	
Revision Note		^	
		~	
Vendor:			

Next, create the new tags for the AOI. Be sure to select the proper usage for the tag (Input vs Output) It is sometimes useful to label inputs and outputs directly in the tag. It can avoid confusion when writing the programming for the local tags.

- Current_In
- Current_Out
- Efficiency_Out
- Horsepower_Out
- Power_In
- Power_In_Output
- Power_Out_Output
- RPM_In
- RPM_Out
- Torque_Out_Ftlb
- Torque_Out_Nm
- Voltage_In
- Voltage_Out

cope:	P_M	otor_Data	¥	Show	: All Tags			~	Y. Enter Name I	ilter		
ata Context:	P_M	otor_Data <	definition> 🗸	*								
Name	-8 4	Usage	Alias For		Data Type	Description	External Access	Constant	Style	^	Properties	
Current_In	u I	Input			REAL		Read/Write		Float		31 2↓ 🗉 🗲	Extended Pr
Current_0	ut (Output			REAL		Read/Write		Float		General	
Efficiency	Out (Output			REAL		Read/Write		Float		 Data 	
EnableIn	1	Input			BOOL	Enable Input - Sys	Read Only		Decimal		Value	
EnableOut	t (Output			BOOL	Enable Output - S	Read Only		Decimal		Force Mask	
Horsepow	er_Out	Output			REAL		Read/Write		Float		Produced Con	nection
Power_In	1	Input			REAL		Read/Write		Float		Consumed Connection	
Power_In_	Out	Output			REAL		Read/Write		Float		Parameter Con	nections {0:0
Power_Ou		Output			REAL		Read/Write		Float			
RPM_In		Input			REAL		Read/Write		Float			
RPM_Out	(Output			REAL		Read/Write		Float			
Torque_0	ut_Ftlb (Output			REAL		Read/Write		Float			
Torque_0	ut_Nm (Output			REAL		Read/Write		Float			
Voltage_Ir	n l	Input			REAL		Read/Write		Float			
Voltage_C)ut (Output			REAL		Read/Write		Float			

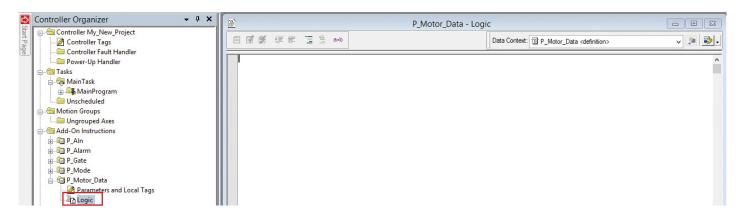
In order to write the code for the P_Motor_Data structured text, we must first investigate some basic programming techniques. This form of programming can account for many different scenarios. This is universal (to some extent) to most programming languages. The main structured text statements in FLUID*Mechatronix*[™] are:

lf Then Else Elsif End_lf

The main goal of the P_Motor_Data AOI is to create outputs for the types of motor data that the FLUID*Mechatronix*[™] system requires. This same task can be accomplished by many of the built in FBD instructions such as less than, greater than, and logic. However, structured text can put all of these variables into one FBD. This consolidates programming and saves the PAC memory space.

In programming, data is read from left to right, top to bottom (just like reading a book). The first structured text programming that needs to be written is the RPM Output code.

Open the structured text Logic window by double-clicking Logic in the P_Motor_Data AOI.



Write the following text in that window: (*RPM Output*)

If the code is written correctly it will turn green. This code is strictly a means of labeling code within the structured text. It is not necessary for the program to operate, but allows the programmer to quickly navigate the code if any changes or problems occur. After placing the label, we can write the specific code below for the RPM data:

```
If RPM_In > 0 Then
RPM Out := RPM In;
```

Local tags in the structured text will automatically turn red. This lets the programmer know that an eligible tag has been selected for the specific task. Keep in mind, structured text programming also has a built in error checker similar to the other types of programming.

This code is simply checking to see IF RPM data is available (any number greater than 0) on RPM_ In. Then, RPM_Out is defined to be equal to (:=) RPM_In. The semi-colon basically says that the specific part of the instruction is finished. This may seem redundant, but this is needed to show proper motor output. Next we need to define when the output of RPM_Out is zero using the Else programming statement. In this case Else represents all other cases from above. Our next line of code is the following:

```
Else

<u>RPM_Out</u> := 0;

End_lf;
```

Else RPM_Out is defined to be equal to zero. This means that all other cases for RPM_In will provide an RPM output of zero. With this code, the HMI can display a value of 0 when the motor is not running. If we didn't have this code, the RPM_Out value would be empty or blank on the HMI control screen. The End_If code is there to mark the end of the If statement. The end must be marked so the processor knows when that code has been executed. It is like the period at the end of the sentence. It must be defined in order to move on to the next sentence.

The final RPM programming statement should be the following:

Next we can use similar arguments to define the motor data for voltage and current. Write the code as followes:

```
(*Voltage Current Output*)
If Voltage_In > 0 Then
    Voltage_Out := Voltage_In / 10;
    Current_Out := Current_In / 100;
Else
    Voltage_Out := 0;
    Current_Out := 0;
End_If;
```

The data from the PowerFlex 525 is scaled (multiplied) at the output. The code above will be used to take that data in, and scale it back to its proper output units. When voltage is present, current is also present so we don't need two different If statements. Structured text programming allows computation of basic mathematics on local tags. Voltage_In / 10 takes the input voltage and divides the number by 10. This gets the motor voltage data to display the voltage correctly in terms of Volts. Current_In / 100 takes the input current and divides the number by 100. This gets the motor current data to display the current in terms of Amps. Note: In this case the If and Else statement have more than 1 condition. Each condition needs to be ended with a semi-colon or the processor will get confused.

The next section of code is a little more complicated. It uses the same basic features, but more math is needed to calculate power, torque, and efficiency. Write the code below into My_New_Project.

```
(*Torque Power Output*)
If Power_In > 0 Then
    Torque_Out_Nm := (9.5488 * Power_In * 10 / RPM_In);
    Power_In_Output := (Current_In / 100) * (Voltage_In / 10);
    Power_Out_Output := Power_In * 10;
    Efficiency_Out := (Power_Out_Output / Power_In_Output)*100;
Else
    Torque_Out_Nm := 0;
    Power_In_Output := 0;
    Power_Out_Output := 0;
    Efficiency_Out := 0;
    End_IF;
```

In order to understand the mathematics, we must first investigate torque. Basically, torque measures the amount of force required to make an object rotate. The force causes the shaft to move a specific distance, so it is very similar to work. The torque equation is as follows:

$$\tau = \frac{9.5488 \times P_{OUTPUT}}{RPM}$$

In this case, the power output from the VFD would need to be in kW. The VFD outputs its power value in hecto watts hW. The value is scaled to output data with the best resolution. In order to get our values kW, the power needs to be multiplied by 10. That is why the equation has Power_In * 10. Therefore,

```
Torque_Out_Nm := (9.5488*Power_In * 10 / RPM_In)
```

We know from Unit 1 that Power = Current X Voltage. We can calculate the power that the motor uses by taking the current and voltage values that run the motor. The next line of code is using the power equation. Like before, we need to divide current by 100 and voltage by 10 because the VFD is actually scaling the output values. Therefore,

```
Power_In_Output := (Current_In / 100) * (Voltage_In / 10)
```

The other power values output from the VFD (Power_In) are the amount of power that the motor is creating. Since energy is lost in all mechanical processes, this value is less than the power required to drive the motor. Like before, we needed to multiply this by 10 because of the output scaling that the VFD requires.

Using the two power values, we can calculate the efficiency of the moter by using the following equation. Power_Out_Output is the useful power that we get from the motor and is considered power out in the equation. Power_In_Output is the amount of power required to run the motor. Efficiency can be calculated with the following equation:

$$E = \frac{P_{OUT}}{P_{IN}} \times 100\%$$

In all other cases, we want these values to be zero, so the rest of the code will be the following:

Else Torque_Out_Nm := 0; Power_In_Output := 0; Power_Out_Output := 0; Efficiency_Out := 0; End IF;

The last piece of structured text from this section is the horsepower of the system. In automation plants, many of the motors and components are rated in horsepower. It is sometimes beneficial to show the horsepower that you get from the motor. This is power that the motor is producing (Power_In_Output), and is different than the power required to run the motor (Power_Out_Output). We can use the horsepower equation to set up the code. Power must be in kW.

$$HP = 1.341 \times P$$

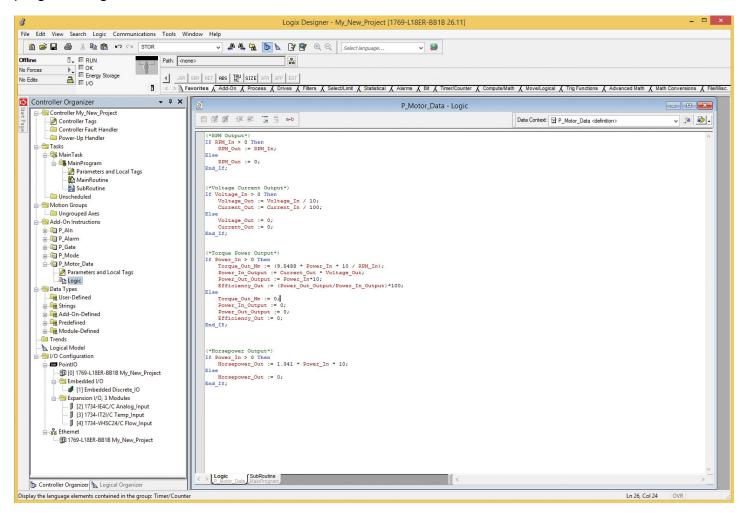
The structured text code would be the following:

```
(*Horsepower Output*)
If Power_In > 0 Then
    Horsepower_Out := 1.341 * Power_In * 10;
Else
    Horsepower_Out := 0;
End If;
```

The result is the amount of useful work that the system can use over time. Note: this is different from the 3 HP rating on the motor.

The final structured text for the P_Motor_Data AOI is below:

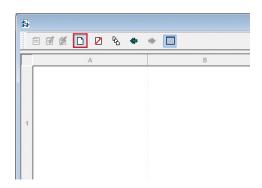
Keep in mind, this is not the only way to accomplish the task of monitoring motor data. This is setup with structured text to show how mathematics and equations can easily be handled with this form of programming.



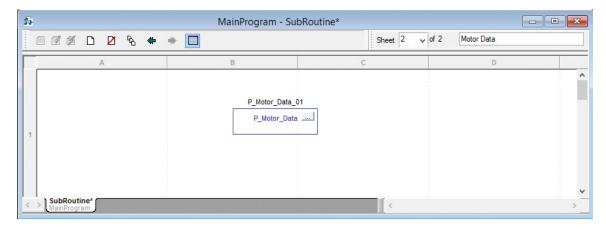
The next step in programming is to wire the P_Motor_Data AOI in the function block. Close the logic window for the AOI and open the function block diagram by clicking on the SubRoutine.

3	Logix Designer - My_New_Project	[1769-L18ER-BB1B 26.11]		- 🗆 🗙
File Edit View Search Logic Communications Tools W	ndow Help v 🎄 🗛 强 📴 🏗 📴 🐨 🔍 🔍 Select language	v 😡		
Offline I RUN Path: mo No Forces ▶ I OK I </th <th></th> <th>λ Alarms λ Bit λ Timer/Counter λ Compute/Ma</th> <th>th 🖌 MovelLogical 🔏 Trig Functions 🔏 Advanced Math</th> <th>✓ Math Conversions ✓ File/Misc.</th>		λ Alarms λ Bit λ Timer/Counter λ Compute/Ma	th 🖌 MovelLogical 🔏 Trig Functions 🔏 Advanced Math	✓ Math Conversions ✓ File/Misc.
Controller Organizer • 1 × Controller My_New_Project Controller Tags Controller Tags Controller Audu Handler Power-Up Handler WiniTask MainRoutine Local Tags MainRoutine Unscheduled		P_Motor_Data - Logic	Data Context: P_Motor_Data ‹definition>	

Open a new sheet in the function block diagram by clicking on the sheet image. Name the new sheet Motor Data.



Navigate to the add-on toolbar above the sheet. P_Motor_Data should now show up as a button. Click on the button and P_Motor_Data AOI will be added to the new function block diagram. The AOI will look like it is empty. We need to enable all of the inputs and outputs that were customized with structured text.



Right-click on the AOI and select Properties. P_Motor_Data Properties window will open.

🚯 MainProgram - Su	bRoutine*
86102 % 🗢 🛎	Sheet 2 v of 2 Motor Data
A B	C D
P_Motor_Data_0 P_Motor_Data	
2	Save Instruction Defaults Clear Instruction Defaults Edit Main Operand Description Go To Ctrl+G Instruction Help F1 Neger Show Description
SubRoutine* MainProgram	Open Instruction Logic Open Instruction Definition
(ManProgram)	Properties Alt+Enter

Enable the desired items by clicking on the checkboxes in the Vis column. Click OK to enable the items. Note: there is a column to the left showing I/O data. This may be useful for finding inputs or outputs quickly.

	Vis	Name	Argument	Value	Data Type	Description
1*	-	EnableIn		1	BOOL	Enable Input - System Defined Paramete
0.	-	EnableOut		0	BOOL	Enable Output - System Defined Param
1*	-	RPM_In		0.0	REAL	
0-	-	Torque_Out_Nm		0.0	REAL	
0*	-	Torque_Out_Ftlb		0.0	REAL	
1*	✓	Current_In		0.0	REAL	
1*	-	Voltage_In		0.0	REAL	
۰0	-	RPM_Out		0.0	REAL	
0-	✓	Voltage_Out		0.0	REAL	
0*	-	Current_Out		0.0	0.0 REAL	
۰0	✓	Power_In_Output		0.0	REAL	
0-	✓	Power_Out_Output		0.0	REAL	
۱.	✓	Power_In		0.0	REAL	
0.	✓	Efficiency_Out		0.0	REAL	
0-	✓	Horsepower_Out		0.0	REAL	
_ So	ort Pa	rameters				Insert Instruction Defaults
						insert Definition Defaults
						Save Instruction Defaults

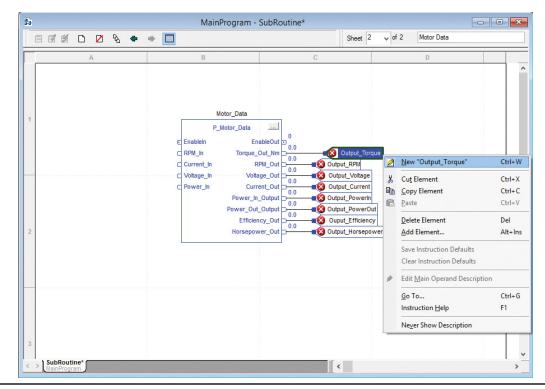
The AOI should now show the desired I/O data.

6	MainProgram - SubRoutin	ie*	-	
834 🗅 🗹 🗞 🗢	*	Sheet 2 v of	2 Motor Data	
A	В	С	D	
				l
	P_Motor_Data_01			
1		eOut		
	RPM_In Torque_Out	0.0		
	Current_In Torque_Out	_Ftlb 0.0 _Out 0		
_	Power_In Voltage	0.0		
	Current Power_In_O	Out		
	Power_Out_O	0.0		
	Efficiency	Out		
2	Horsepower	_Out D		
> SubRoutine*		<		>

Rename the AOI Motor_Data and create a new tag Motor Data in the scope of My_New_Project. Delete P_Motor_Data_01 from MainProgram > Parameters and Local Tags.

24	MainProgram - SubRou	tine*		
8 8 Ø D 🛛 🗞 🌩 🐡 🔲		Sheet 2	✓ of 2 Motor Data	
A	B Motor_Dats P_Motor_Dat EnableIn RPM_In Torq Current_In Torq	C New "Motor_Data" Cut Element Copy Element Paste	Ctrl+W Ctrl+X Ctrl+C Ctrl+V	
2	C Votage_in Power_in Power Power Eff Horse	Delete Element Delete Element but not Tag Add Element Save Instruction Defaults Clear Instruction Defaults Edit Main Operand Descrip	Alt+ins	
		Go To Instruction Help Never Show Description Open Instruction Logic Open Instruction Definition Properties	Ctrl+G F1	

Drag 8 output tags from the toolbar to the fuction block diagram. Name all of the output tags as shown below. Right-click on each tag and add a new output tag to the scope My_New_Project. This allows the PLC access to the data calculated from the AOI (since AOI data is local).



The finished AOI should look like the diagram below. It should also be free of errors. We are not going to add inputs at this time. Inputs will be added in Unit 6: VFD.

34					MainProgram	n - SubRoutine*			- • ×
	888	D	Z	ዔ	*	Sheet 2	✓ of 2	Motor Data	
Г		ļ	A			В		С	
2) SubRout	ine* (tor_Data otor_Data 0 EnableOut 0.0 Torque_Out_Nm 0.0 Torque_Out_Nm 0.0 RPM_Out 0.0 Voltage_Out 0.0 Current_Out 0.0 Power_In_Output 0.0 Power_Out_Output 0.0 Efficiency_Out 0.0 Horsepower_Out 0.0		Dutput_Torque hutput_RPM Dutput_Voltage Dutput_Current hutput_PowerIn Output_PowerOut Output_Efficiency hutput_Horsepower	~
<	MainProgr	am						<	>

Section 1, Focus 4 Builder

Skill

- 1. Explain why structured text can simplify and replace function block diagram mathematics.
- 2. Describe how structured text can be used in conjuction with a function block diagram.
- 3. Why is structured text is similar to other programming languages?