Micro800 Programming Basics

For Classroom Use Only!
Important User Information

This documentation, whether, illustrative, printed, “online” or electronic (hereinafter “Documentation”) is intended for use only as a learning aid when using Rockwell Automation approved demonstration hardware, software and firmware. The Documentation should only be used as a learning tool by qualified professionals.

The variety of uses for the hardware, software and firmware (hereinafter “Products”) described in this Documentation, mandates that those responsible for the application and use of those Products must satisfy themselves that all necessary steps have been taken to ensure that each application and actual use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards in addition to any applicable technical documents.

In no event will Rockwell Automation, Inc., or any of its affiliate or subsidiary companies (hereinafter “Rockwell Automation”) be responsible or liable for any indirect or consequential damages resulting from the use or application of the Products described in this Documentation. Rockwell Automation does not assume responsibility or liability for damages of any kind based on the alleged use of, or reliance on, this Documentation.

No patent liability is assumed by Rockwell Automation with respect to use of information, circuits, equipment, or software described in the Documentation.

Except as specifically agreed in writing as part of a maintenance or support contract, equipment users are responsible for:

• properly using, calibrating, operating, monitoring and maintaining all Products consistent with all Rockwell Automation or third-party provided instructions, warnings, recommendations and documentation;
• ensuring that only properly trained personnel use, operate and maintain the Products at all times;
• staying informed of all Product updates and alerts and implementing all updates and fixes; and
• all other factors affecting the Products that are outside of the direct control of Rockwell Automation.

Reproduction of the contents of the Documentation, in whole or in part, without written permission of Rockwell Automation is prohibited.

Throughout this manual we use the following notes to make you aware of safety considerations:

| WARNING | Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss. |
| IMPORTANT | Identifies information that is critical for successful application and understanding of the product. |
| ATTENTION | Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you:
• identify a hazard
• avoid a hazard
• recognize the consequence |
| SHOCK HAZARD | Labels may be located on or inside the drive to alert people that dangerous voltage may be present. |
| BURN HAZARD | Labels may be located on or inside the drive to alert people that surfaces may be dangerous temperatures. |
Micro800 Programming Basics

Contents

Before you begin .......................................................................................................................... 5
About this lab .............................................................................................................................. 5
Tools & prerequisites ................................................................................................................. 5
Get familiar with the Connected Components Workbench design environment ..................... 6
Create a CCW project and program a Micro850 controller ...................................................... 12
Build and Download your Micro850 Application .................................................................. 21
Debug your Micro850 program ............................................................................................... 25
Learn about Variables and Data Types ................................................................................... 30
Data types .................................................................................................................................. 31
Learn how to create variables........................................................................................................32
Learn how to Implement an Instruction Block ..............................................................................34
Learn about Controller Security ....................................................................................................45
Learn how to add a plug-in module .................................................................................................46
Learn about User Defined Function Blocks ....................................................................................50
(OPTIONAL) PanelView Component HMI Design using Connected Components Workbench .......57
Add a PanelView Component to Your Connected Components Workbench Project....................58
Configure Your PanelView Component’s Communication Settings ................................................62
Create HMI Tags ..............................................................................................................................65
Create a Screen .................................................................................................................................69
Create a Pushbutton .........................................................................................................................72
Add an Image to Your Application ..................................................................................................78
Create a Goto Config Pushbutton ....................................................................................................82
Create a Numeric Display Object .....................................................................................................84
Create a Bar Graph Object ..............................................................................................................86
Create an Alarm ...............................................................................................................................89
Before you begin

About this lab
Connected Components Workbench (CCW) is the integrated design environment software package that is used to program, design, and configure your Rockwell Automation Connected Components devices such as, Micro800 programmable logic controllers, PowerFlex drives, SMC soft-starters, and PanelView Component operator interface terminals.

This lab will demonstrate and help guide you on how to use and program a Micro850 controller using the CCW software.

This lab takes approximately 60 minutes to complete.

Tools & prerequisites
- Software: Connected Components Workbench v2.00.00
- Hardware: Micro850 Programmable Logic Controller, Catalog 2080-LC50-24QBB
Get familiar with the Connected Components Workbench design environment

This section will help get you familiar with the Connected Components Workbench design environment. As our goal to help simplify your engineering efforts, we’ve developed CCW using the Microsoft Visual Studio Shell. This common and popular software shell provides you the benefits of a common look, feel, and design environment when transitioning from other similar software packages.

Let’s take a couple minutes to get familiar with the CCW design environment.

1. Start the Connected Components Workbench software.
   Double-click the Connected Components Workbench shortcut icon on your desktop.

   ![Connected Components Workbench icon]

   You can also launch the program from your Windows Start Menu by going to: Start > All Programs > Rockwell Automation > CCW > Connected Components Workbench
2. Get familiar with the CCW design environment.

This is the default project layout. Below are descriptions of each of the panels’ contents and the general task the pane is used for.
3. Project Organizer

The Project Organizer displays the contents of your project in an organized tree view, providing access to each of the devices and project elements. From the Project Organizer, you can add, move, or delete devices and project elements, as well as double-click them to display their contents.

If your project contains a Micro800 controller, the Project Organizer also displays the logic programs, variables, and user-defined function blocks associated with that Micro800 controller.
4. Device Toolbox

The Device Toolbox displays all of the devices that you can add to your Connected Components Workbench™ project. From the Device Toolbox, you can select devices for your project from the following two tabs:

- Discover - discovers devices that are connected to your computer and recognized by Connected Components Workbench.
- Catalog - browses a catalog of devices that are included with Connected Components Workbench.
5. Toolbox

The Toolbox displays icons for items that you can add to programs. From the Toolbox, you can drag and drop individual Toolbox elements onto a design view surface or copy and paste these into a code editor. Each of these actions adds the fundamental code to create an instance of the Toolbox item in the active project file.
6. Output

In the Output window, you can view and manage general purpose and debug messages generated by the various features of Connected Components Workbench. From the Output window, you can do the following:

- Review status messages
- Locate errors within programs
Create a CCW project and program a Micro850 controller

In this section, you will create a CCW project and learn how to create a program for a Micro850 programmable logic controller.

You will learn how to:
- Create a CCW project
- Add a Micro850 controller to your project
- Program a simple motor control seal-in circuit

1. Start the Connected Components Workbench software.
   Double-click the Connected Components Workbench shortcut icon on your desktop.

2. Add a Micro850 to your project.
   Locate the Device Toolbox (upper right-hand corner). Expand Catalog and locate the Controllers folder. Expand the Controllers folder and locate the Micro850 controller catalog 2080-LC50-24QBB. Double-click this Micro850 controller catalog. This will add a Micro850 controller to your Project.
3. Notice that the Micro850 shows up in your Project Organizer on the left-hand side.

Alternatively, you can drag and drop the controller from the Device Toolbox into the Project Organizer.
4. Add a Ladder Diagram program.

Right-click Programs under the Micro850 in your Project Organizer, and select Add → New LD : Ladder Diagram.
5. Notice a new Ladder Diagram program called **UntitledLD** will be added under Programs.

Micro800 controllers allow you to create multiple programs as well as use multiple types of programs (such as Structured Text or Function Block Diagram) in the same controller application.

Since we’ll be creating a Motor Circuit in this program, let’s rename it **Motor_Circuit**.
6. Right-click the **UntitledLD** program and select **Rename**.
7. Rename the program, **Motor_Circuit**.

8. **Create a motor seal-in circuit in your Motor_Circuit Ladder Diagram program.**

   This circuit will use the DI0 switch on the Demo as your Start button, and the DI1 switch as your Stop button. The pilot light wired to DO8 will simulate your motor coil.
9. Double-click the **Motor_Circuit** program. A ladder diagram editor will appear in the main project workspace with one empty rung.

10. Locate the **Direct Contact** instruction in the Toolbox pane (right-hand side), and drag-and-drop it onto the left side of the rung.

After inserting the Direct Contact instruction, the Variable Selector will automatically pop-up, allowing you to select the variable or I/O point to assign to this instruction.
11. We will be assigning an embedded I/O point to this instruction; therefore we want to select the I/O – Micro850 tab to show the available embedded I/O points. Then select _IO_EM_DI_00, and click OK.

12. Your rung should look like the following.
13. Locate the **Reverse Contact** instruction in the Toolbox and drag-and-drop it onto your rung, just to the right of the Direct Contact you inserted in the previous step. Assign it to the embedded I/O point, _IO_EM_DI_01. Your rung should look like the following.

14. Locate the **Direct Coil** instruction in the Toolbox, and drag-and-drop it onto the far right side of the rung, and assign it to the embedded I/O point, _IO_EM_DO_08. Your rung should look like the following.

15. Locate the **Branch** instruction in the Toolbox and drag-and-drop it on top of the Direct Contact on the far left of the rung. Your rung should look like the following.

16. Drag-and-drop a **Direct Contact** in to the Branch that you just added, and assign it to the embedded I/O point, _IO_EM_DO_08. Your rung should look like the following.

17. You’ve completed creating your motor seal-in circuit. Save your project and proceed to the next section to build and download your application to the Micro850 controller.
Build and Download your Micro850 Application

In this section, you will learn how to build and download your Micro850 application to the controller.

Before you can download an application to the controller, you must build it to verify that there are no errors with the programming.

1. Build your application by right-clicking the Micro850 in your Project Organizer, and selecting Build.
2. When the build is complete, you will see a message in the **Output** panel that the build has succeeded. If there were errors in your programming, then they would be listed in the **Output** panel as well – and clicking on the error would direct you to the error in your program.

Now that your build has completed, you can download the program to your controller.

3. Download your program to your Micro850 by right-clicking the Micro850 in your Project Organizer, and selecting **Download**.
4. The Connection Browser will appear. Browse for your controller by expanding **USB** and selecting **16, Micro850, Micro850**, then clicking **OK**.

![Connection Browser](image)

5. If the Controller is in Run Mode, you will be prompted to change it to Remote Program mode. Click **Yes**.

![Download](image)

6. The download will proceed. When the download is complete, you will be prompted to put the controller back in to Run Mode. Click **Yes**.

![Download Confirmation](image)
7. Notice the messages in the Output panel that indicate the Download has completed successfully.

8. You have completed downloading to your Micro850 controller. Proceed to the next section to test and debug your application.
Debug your Micro850 program

In this section, we will demonstrate how to debug your Micro850 program. By debugging your program, you can view your program visually in real-time and watch values change in the program, as well as visually debug your Ladder Logic or Function Block Diagram.

1. Click the “Play” button in the Toolbar at the top of the workspace environment. This will put your program into “Debug” mode.

2. You should see the Ladder Diagram change color. The rung will turn blue, and any Boolean instructions that are active will turn Red.
3. Turn and release the DI0 switch on the Demo hardware. Notice the _IO_EM_DI_00 Direct Contact instruction turn red as you turn on the switch, and then turn blue as you release it (if you turn and release the switch too fast, you may not see it update in the ladder diagram). Then notice the _IO_EM_DO_08 Direct Contact and Direct Coil instructions turn red. You should also notice that the DO8 light on the Demo is now on.

![Motor Circuit Diagram](image)

This is a typical motor seal-in circuit (and can also be applied in non-motor circuits as well). The Output Coil is turned on using a Direct Contact and then the active state of the Output Coil seals in the circuit. The circuit is unsealed when a Reverse Contact (normally closed) is opened.

4. Turn and release the DI1 switch on the Demo to turn off the output. Notice the light on your Demo turn off and the corresponding changes in your Ladder Diagram.

5. So far, you’ve debugged your program primarily by viewing real-time changes in the Ladder Diagram editor. In some instances, you may just want to view the real-time changes in a list format. You can do this by looking at them in the Variables list.

Since the variables we’re working with right now are embedded I/O points, we need to open the Global Variables list.
6. Double-click **Global Variables** in your Project Organizer. The Global Variables pane will launch in a new tab in the main project workspace.

The Global Variables pane is a list of all the Global Variables in your program. You will learn more about Global Variables and other types of variables in the next section.

7. Locate the **_IO_EM_DO_08** embedded I/O variable in the Global Variable list, and notice that the logical value checkbox is empty.
8. Turn and release the DI0 switch on the Demo. Notice there is now a checkmark in the Logical Value for _IO_EM_DO_08, and the light on the Demo is on. You may have also noticed a checkmark appear in the _IO_EM_DI_00 logical value as you turned on the switch and then noticed the checkmark go away as you released the switch.

9. Turn and release the DI1 switch on the Demo. Notice the checkmark go away in the logical value for _IO_EM_DO_08, and the light turn off on the Demo.
10. You have completed debugging your program. To exit Debug mode, click the Stop button in the toolbar.

Being able to debug a program in real-time is a very valuable tool in the programming and trouble-shooting process, and Connected Components Workbench makes this design step very simple and easy.
Learn about Variables and Data Types

In this section, we will discuss what a Variable is, and the different Data Types available.

A variable is a unique identifier of data. A basic example of a variable is what you've already been referencing in the lab for Embedded I/O points. The Embedded I/O variables are Boolean data types that are direct references to the embedded input and outputs on the controller. They are identified by variables that start with the prefix _IO_EM, and are globally scoped. We will discuss variable scope a little later.

Micro800 controllers also have System Variables of varying data types that reference internal system values of the controller that a user may want to use in their programming, or for troubleshooting purposes. System Variables start with the prefix __SYSVA. An example of a system variable that is commonly used is the __SYSVA_FIRST_SCAN variable. This is a Boolean variable that is TRUE when the Micro800 controller is going through its first scan of the program – typically used for programming startup routines.

Variables can be created dynamically as you need them, and they can be named anything you want (as long as it's not a reserved name). You can also create variables for local program use only, or you can create them for Global use (for all programs to use) – this is what we refer to as variable scope. Global Variables are created in the Global Variables list, and Local Variables are created in the Local Variables list of the specific program.

Being able to create variables dynamically and use custom names provides you, as a programmer, great flexibility
and customization that will help you create code and troubleshoot faster.

**Data types**

When you create a variable, you have to specify its data type. A data type defines the type of data that the variable represents, such as an integer, real (floating point), Boolean, time, double integer, etc. Data types can also be data structures of an Instruction Block.

CCW supports the 19 elementary IEC 61131-3 data types below.

- BOOL
- SINT
- USINT
- BYTE
- INT
- UINT
- WORD
- DINT
- UDINT
- DWORD
- LINT
- ULINT
- LWORD
- REAL
- LREAL
- TIME
- DATE
- STRING
Learn how to create variables

In this section of the lab, you will learn how to create variables for use in your program. The variables you create in this section of the lab will be used in the next section of the lab.

1. Double-click Local Variables in your Motor_Circuit program to launch the Variables panel.

![Project Organizer]

2. Create a variable called Motor_On_Time of Data Type TIME.

3. Create a variable called Motor_On_Time.ms of Data Type INT and with an Initial Value of 5000.
4. Create a variable called Motor_Timer of Data Type TON.

5. You have completed creating variables to be used in the next section of the lab.

A TON data type is actually the data structure of a Timer-on-Delay Instruction Block. We will discuss Instruction Blocks in the next section.
Learn how to Implement an Instruction Block

An Instruction Block is essentially a function block that has been predefined to perform a specific task or function. Instruction Blocks include functions such as Timer-on-delay, Timer-off-delay, Math instructions, Data-type conversions, Motion instructions, and so forth.

In this section of the lab, you will learn how to implement a Timer-On-Delay Instruction Block (TON). This instruction block will be inserted into your motor circuit and will turn on the motor coil, and then automatically turn off the motor coil after 5 seconds.

You will also learn how to implement an ANY_TO_TIME Data Conversion Instruction Block to convert an Integer to a Time value.

1. Drag-and-drop a **Branch** instruction to right side of the rung, wrapping around the coil instruction.

2. Locate the **Block** instruction in the Toolbox.
3. Drag-and-drop this Block instruction into the branch that you just added.

4. The Instruction Block Selector will appear. This is where you can select the type of Instruction Block you would like to use. As you can see, there is a long list of different types of instruction blocks that you can choose from. Feel free to take a minute to scroll through this list to see what types of instruction blocks are available.
5. You can filter the instruction blocks by Name, Category, or Type. Since we want to use a Timer-On-Delay instruction block, type TON in the Name filter box at the top of the Name column. This will filter the choices to only Instruction Blocks that start with TON.

6. Select the TON Instruction Block – this is the Timer-on-Delay. Then select the Instance combo box pull-down, and select the variable instance, Motor_Timer that you created in the previous section, and click OK.
7. Your ladder program should look like the following.

8. Next, hover the mouse cursor over the blue box next to the PT parameter of the Motor_Timer TON instruction. You will notice a light blue highlighted box appear.

9. Click this box, and a pull down combo box will appear. Find and select the variable Motor_On_Time and then press the Enter key.
10. Your ladder program should look like the following.

![Ladder Diagram]

11. Insert a Reverse Contactor after the _IO_EM_DI_01 Reverse Contactor, as shown below.

![Ladder Diagram]

12. The Variable Selector will display. Select the Local Variables – Motor_Circuit tab, and then click the empty cell shown below.

![Variable Selector]

13. Expand the variable, **Motor_Timer** and select, **Motor_Timer.Q**. Then click **OK**.

14. Your ladder program should look like the following.
15. Locate the rung instruction in the Toolbox.

16. Drag and drop the **Rung** instruction below Rung 1.
17. Your program should look like the following.

18. Insert a **Block** instruction into the rung you just created, and select the **ANY_TO_TIME** Instruction Block. Then click **OK**.
19. Your program should look like the following.

20. Select the variable `Motor_On_Time_ms` for the i1 parameter.
21. Select the variable **Motor_On_Time** for the o1 parameter.

22. The ANY_TO_TIME instruction block is being used to convert an integer value into a time value that is used as the preset time for the Motor_Timer. The integer value represents time in milliseconds.

23. Your program should look like the following.

24. Build your program, and download it to the Micro850 (if you forgot how to do this, go back and reference the section **Build and Download your Micro850 Application**).

25. After completing the download, put the program into Debug mode by clicking the play button (or pressing the F5 key).

26. Now test your program. Turn the DI0 switch on, and watch the DO8 light turn on. After 5 seconds, the light should turn off.
27. Change the value of the variable, **Motor_On_Time_ms**, to change the amount of time the light stays on to 10 seconds (remember we enter the value in milliseconds). Make sure to press enter after changing the value.

![Variable Table]

28. Now test your program again. Turn the DI0 switch on. The DO8 light should now stay on for 10 seconds, before turning off.

29. Click the Stop button to exit Debug Mode (or press Shift+F5).

30. You have completed this section of the lab.
Learn about Controller Security

In this section, you will learn about controller security for the Micro800. You can implement controller security by specifying a password for the controller. This helps protect your Micro800 controller from unauthorized access, such as downloading and uploading.

You will not actually set the controller security, but instead, just learn how and where to configure it.

To set, change, and clear the password on your Micro800 controller, you will need to open the controller’s General Properties Pane by double-clicking on the Micro800 icon in your project organizer. Then by clicking the Secure icon, you will be given the option to Set, Change, or Clear the password on the controller.
Learn how to add a plug-in module

In this section of the lab, you will learn how to add an Analog Input plug-in module. A plug-in module is a module that you can plug into the Micro800 chassis to allow you to add additional I/O or Communications Options to your controller.

1. Double-click your Micro850 controller in the Project Organizer.

   ![Project Organizer](image)

   This will bring up the General Controller Properties in the main project window.

2. Right-click the second plug-in module slot, and select 2080-IF4.

   ![Micro850](image)
3. Notice that the 2080-IF4 module is now added to your chassis. The configuration properties should also show up in the pane below it.

4. Configure the Input Type for Channel 0 to Voltage.
5. Configure the input state for Channel 1, 2, and 3 to **Disabled**.

6. Build and download your program to the Micro850.

7. Start debugging by clicking Play, or by pressing the F5 key.
8. Double-click **Global Variables** in the Project Organizer.

![Project Organizer]

9. Locate the variable _IO_P2_AI_00. This is the raw data value in relation to the voltage that is wired to Channel 0. The value should range from 0 to 65535 in relation to a 0 to 10 volt input.

![Variable Table]

10. On your demo hardware, turn the potentiometer labeled **Speed Command** and notice the value of _IO_P2_AI_00 change.

11. Stop debugging by clicking the Stop button, or by pressing Shift+F5.

12. You have completed this section of the lab.
Learn about User Defined Function Blocks

In this section of the lab, you will learn about a User Defined Function Block (UDFB), and how to create one using Structured Text.

A User-Defined Function Block is a user defined program that can be packaged into an Instruction Block and reused within your Micro800 project. A UDFB can be written in Ladder, Function Block, or Structured Text.

You will be creating a UDFB to calculate the volume of a cylinder based on an inputted radius and height value.

1. In your current project, right click User-Defined Function Blocks and select Add → New ST: Structured Text.
2. A program called **UntitledST** will be created under User-Defined Function Blocks.

3. Rename this UDFB, **Calc_Volume**.

4. Double-click Local Variables under **Calc_Volume**.
5. Create the following variables. Take careful note to properly configure the **Direction** property. This property defines whether the variable is an Input, Output, or standard Variable.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius</td>
<td>REAL</td>
<td>VarInput</td>
</tr>
<tr>
<td>Height</td>
<td>REAL</td>
<td>VarInput</td>
</tr>
<tr>
<td>Volume</td>
<td>REAL</td>
<td>VarOutput</td>
</tr>
</tbody>
</table>

6. Next, double-click the **Calc_Volume** UDFB to launch the program editor in the main project window.

7. Add the following line of code to the program.

   ```
   Volume := 3.14 * Radius * Radius * Height;
   ```

8. You have completed creating your UDFB.

9. Save your program.
10. Next, create a new ladder diagram program called **Tank_Volume**.

![Project Organizer diagram](image)

11. Open the Local Variables for the Tank_Volume program, and create the following variables. Notice the Data Type for the variable **Calc_Tank_Volume** is the Calc_Volume UDFB you created.

![Tank_Volume VARIABLES](image)
12. Next, open the **Tank_Volume** program, and add a **Block** Instruction to the first rung.

![Tank_Volume program](image1)

13. Select the **Calc_Volume** UDFB, and specify the Instance **Calc_Tank_Volume**. Then click **OK**.

![Calc_Volume UDFB](image2)
14. Next, specify the following variables for each parameter of the Block.

![Block Diagram]

15. Save your project.

16. Build and download your program to your Micro850 controller.

17. Once your download is complete, press the F5 key to enter Debug Mode.

18. Open the Local Variables of your Tank_Volume program, and set the value of Radius to 5, and Height to 10.

![Variables Table]

19. The value of Volume should read 785.0.

![Variables Table]
20. Change the value of Radius to 7.

<table>
<thead>
<tr>
<th>Name</th>
<th>Logical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius</td>
<td>7.0</td>
</tr>
<tr>
<td>Height</td>
<td>10.0</td>
</tr>
<tr>
<td>Volume</td>
<td>1538.6</td>
</tr>
</tbody>
</table>

21. The value of Volume should read 1538.6.

22. Press Shift+F5 to exit Debug Mode.

23. You have completed this section of the lab.
(OPTIONAL) PanelView Component HMI Design using Connected Components Workbench

This section of the lab will demonstrate how to create a simple PanelView Component (PVc) application using the new DesignStation 3.0 that has been incorporated into the Connected Components Workbench software package.
Add a PanelView Component to Your Connected Components Workbench Project

In this section of the lab, you will learn how to add a PanelView Component to an existing CCW project.

1. Double-click the Connected Components Workbench icon on your desktop to launch the Connected Components Workbench software.

2. Click on File ➔ Open

3. Browse to the following directory: C:\Lab Files\Basic PLC Programming with Micro800 Controller\PVc Lab Files\Starting_Project\

4. Select Starting_Project.ccwsln and click Open.

5. The Starting Project will open, and your workspace should look like the following.
6. Locate the **Device Toolbox** on the right most corner of the screen.
7. Expand the **HMI folder** to display a list of available catalog models that can be added to your project.

8. Locate the PanelView Component catalog **2711-T6T** and drag-and-drop it into your Project Organizer on the left hand side.
9. A PanelView Component application called **PVcApplication1** will be added to your Project Organizer.
Configure Your PanelView Component's Communication Settings

In this section, you will learn how to configure your PanelView Component to communicate to a Micro850 controller over Ethernet.

1. Double-click on PVcApplication1 to launch the HMI Settings Panel.

2. You will automatically be taken to the Communication settings pane.
3. Under the **Protocol** configuration, select **Ethernet** and choose **Allen-Bradley CIP** from the dropdown box.
4. By default, a controller node has already been created and added to your application, but we will still need to specify the IP Address of the controller. Even though we aren’t going to run the application in this lab, go ahead and set the IP Address to **192.168.100.4**.
Create HMI Tags

In this section, you will learn how to create HMI tags to bring back data from a Micro850 controller.

The Tag Editor tab is where you create, view, and modify tags used by your HMI application. The tag types are external, memory, system and global. Each type has a different data source. In this section we will learn how create a few individual tags. These tags can either be typed in one by one, or they can be copied and pasted from an external location, such as Microsoft Office Excel.

1. Double-click HMI Tags in your Project Organizer to launch the HMI Tag Editor panel.

2. Click Add to add a tag to your PanelView Component application.
3. Rename the tag to **Start_Sealer** and change the Data Type to Boolean.

4. Select the **Address** field and then select the **ellipses** to browse the Micro850 controller program for the controller address.
5. Select the variable `Start_Sealer` and then click OK.

6. Select the Controller field and select the controller PLC-1 from the dropdown.
7. You have completed adding a tag. Repeat these steps to add the following tags.

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Data Type</th>
<th>Address</th>
<th>Controller</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start_Sealer</td>
<td>Boolean</td>
<td>Start_Sealer</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Stop_Sealer</td>
<td>Boolean</td>
<td>Stop_Sealer</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Prepare</td>
<td>Boolean</td>
<td>Prepare</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Machine_Ready</td>
<td>Boolean</td>
<td>Machine_Ready</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Glue_Temp_Setpoint</td>
<td>Real</td>
<td>Glue_Temp_Setpoint</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Carton_Temp_Setpoint</td>
<td>Real</td>
<td>Carton_Temp_Setpoint</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Carton_Count_Setpoint</td>
<td>16 bit integer</td>
<td>Carton_Count_Setpoint</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Cartons_Rejected</td>
<td>16 bit integer</td>
<td>Cartons_Rejected</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Glue_Temp</td>
<td>Real</td>
<td>Glue_Temp</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>Boolean</td>
<td>Reset</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Seal_Fault_Alarm</td>
<td>Boolean</td>
<td>Seal_Fault_Alarm</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Cartons_Accepted</td>
<td>16 bit integer</td>
<td>Cartons_Accepted</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Seal_Fault</td>
<td>Boolean</td>
<td>Seal_Fault</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Machine_Running</td>
<td>Boolean</td>
<td>Machine_Running</td>
<td>PLC-1</td>
<td></td>
</tr>
<tr>
<td>Preparing_Machine</td>
<td>Boolean</td>
<td>Preparing_Machine</td>
<td>PLC-1</td>
<td></td>
</tr>
</tbody>
</table>
Create a Screen

This section of the lab will show you how to create a screen for your PanelView Component application.

1. By default, a screen is already added to your application called Screen_1. Rename this screen to Case Sealer by right-clicking on it, and selecting Rename.

2. Right Click on Case Sealer and set it up as a Startup Screen.
3. Double-click the Case_Sealer display icon to open the display for editing.

4. Open the Toolbox to access screen objects to use for designing your display. Select View → Toolbox from the menu bar.
5. The Toolbox should appear in the lower right hand corner of your workspace.
Create a Pushbutton

1. Locate the **Momentary Push Button** in your Toolbox, and drag-and-drop it onto your screen.

2. Once the push button is on the screen, double-click on it to change the appearance and properties of the push button states.
3. Under **Background Color** column, click on the (…) ellipse to bring up the Color Selector, and change the color of State 1 to green.

4. Repeat the previous step to change the background color of State 2 to **green** also.

5. Change the **Caption Text** of State 1 to “Start Sealer” and State 2 to “Sealer Started”.

6. Click **OK** to close the State Editor and apply the changes.
7. Right-click on the push button and select **Properties** to launch the object’s Properties pane.

8. The object’s Properties pane will open on the right hand side.
9. Configure the following Properties to the values specified below.

   - Height: 50
   - Width: 70
   - Left: 15
   - Top: 20

10. Click the **Indicator Tag** drop down in the Properties pane, and select **Machine_Running** from the list of tags.

11. Repeat the following step to set the Visibility Tag to **MachineReady**, and the Write Tag to **StartSealer**.
12. You have completed creating your first push button. If you hover the cursor over the momentary push button you can see its tag connections.

13. Add another momentary push button next to the Start Sealer push button and configure the States of this pushbutton as shown below.
14. Open the pushbutton’s properties pane and configure the following properties.

   Height: 50
   Width: 70
   Left: 90
   Top: 20

15. Configure the push button’s following tag connections.

   Indicator Tag: Cartons_Accepted
   Visibility Tag: Carton_Rejected
   Write Tag: Stop_Sealer

16. Your screen should look like the picture below.
Add an Image to Your Application

In this section of the lab we will be importing an external image into the application.

1. Locate the Image tool in the Toolbox and drag-and-drop it onto the screen in the lower left corner.
2. Double-click the Image object you just added. This will launch the Image Import prompt.

3. Click Import.

4. Browse to C:\Lab Files\Basic PLC Programming with Micro800 Controller\PVc Lab Files\Images\
5. Select **Glue Gun Box.bmp** and click **Open**.

6. Click on **Select**
7. Right-click on the image to bring up its Properties panel.

8. When the Properties panel opens on the right, please fill in the tag information as shown below in the picture.

9. If you hover over the image you can see what tag its connected to.
Create a Goto Config Pushbutton

Returning to the Configuration screen of the PanelView Component is very important for all applications. The configuration screen allows users to change terminal settings, as well as observe its communication, set up and memory usage. Follow the steps below to create a Goto Config pushbutton.

1. Locate the **Goto Config** pushbutton in your Toolbox.

2. Drag-and-drop it onto your screen, just below your Stop Sealer pushbutton.
3. Double-click the Goto Config pushbutton and change the text to **Config**, then click **OK**.

![](image1)

4. Change the button's width to 70 and height to 50. Your screen should look like the following.
Create a Numeric Display Object

A numeric display shows the value of the tag that it is connected to. For this application, we would like to display the number of cartons that need to be sealed, as well as the number of cartons that have been sealed successfully.

1. Locate the **Numeric Display** object in your Toolbox.

2. Drag-and-drop the Numeric Display object onto your screen, next to the Stop Sealer pushbutton.

3. Change the properties of the Numeric display by right clicking on the button and choosing properties.
4. Change the following Property values.
   Width: 81
   Height: 33

5. Set the Read Tag to Carton_Count_Setpoint.
Create a Bar Graph Object

The Bar Graph object allows a user to easily see the value of a tag using a bar graph format. In this example, we can use a Bar Graph to show the operator the temperature of the glue.

1. Locate the **Bar Graph** object in the Toolbox.

![Toolbox with Bar Graph highlighted]

2. Drag-and-drop the Bar Graph object onto your screen, just below the numeric display that you added in the previous section.

![Screen with Bar Graph and numeric display]
3. Right-click the Bar Graph, and select **Properties**.

4. Set the following Property values.

   - **Background Color**: Yellow
   - **Maximum Value**: 20
   - **Show Scale**: True
5. Set the Read Tag to Glue_Temp.

6. You have completed adding a Bar Graph Object to your screen.
Create an Alarm

Alarms allow you to notify an operator of events in your control system based on configured setpoints or values specified in the Alarm.

1. Double-click Alarms in your Project Organizer.

![Project Organizer screenshot showing Alarms]

2. The HMI Alarms panel will launch in the main project window. Click Add Alarm.

![Alarm panel showing Add Alarm and Seal_Fault_Alarm]

3. Select Seal_Fault_Alarm from in the Trigger column and configure the Message to Machine Stopped.

![Alarm configuration showing Seal_Fault_Alarm and Machine Stopped]

89 of 91
4. The Seal_Fault_Alarm is a Bit Type alarm, so it will trigger based on a True condition of the trigger tag.

5. Add another Alarm for **Cartons_Rejected**.

6. Change the following settings for this alarm.

   - **Alarm Type**: Numeric
   - **Value**: 3
   - **Message**: Nearing Emergency Seal Fault

7. The Cartons_Rejected alarm is a Numeric Type Alarm. If the value of the trigger goes higher or is equal to the threshold value, then the alarm will be triggered.

8. You have completed creating an alarm.

You have completed this lab.
After you have completed step #8 on page 91- if you’d like to load your Panelview screens on to the demo- do the following steps

1- Build your PLC program
2- Save your program- (give the program your name)
3- Download your program to the M850
4- While downloading, ask your instructor for a USB Memory stick
5- When download is complete, put the PLC into Run mode
6- Plug the USB memory stick into your laptop
7- In your CCW application, right click on your PanelView Component application and select “Export Application”
8- Set the export path to the USB memory stick
9- Export
10- When the export is complete- remove the USB memory stick and plug it into the USB port on the Demo labeled ‘CHMI USB Port’.
11- In the PVC config menu, copy your program from USB to internal
12- Run your program

To see if your page is working- go into ‘Debug’ mode on the M850 and manipulate the addresses that you used for the PVC screen