Chapter 15 HUMAN MACHINE INTERFACE

Introduction

Communications between processor and HMI (human machine interface) is an important subject as well as constructing an operator interface. The chapter includes procedures for attaching computers as HMI devices to the CompactLogix processor from A-B and the Siemens 1200 processor. Graphic control packages used are A-B’s RSView ME and Siemens’ WinCC. Other packages exist and were not excluded based on their capabilities. The ones used are among the more common and popular ones used today.

The following HMI panels communicating to PLC processors will be discussed in this chapter followed by a lab that can show advantages and disadvantages of each.

![HMI Graphic Control Packages](image1)

In each case, the emphasis is on getting a simple application operational and then expanding from that, remembering the analogy of the kite flying with a simple string over Niagara River. Later, the more difficult applications are discussed but only after a single button is programmed from the HMI and communicates successfully to the PLC.

Historical Panel Design

The design of an operator panel requires much coordination with the programming of the PLC and the design of the machine being controlled. Before the computer-designed systems, there were individual component systems that were hard-wired to the control devices inside the panel.

![A Simple Control Panel with Push Buttons and Switches with Indicator Lights](image2)
Fig. 15-3 This Printer is used for alarms for a process. Each alarm was recorded at the time of occurrence and printed as a single line of data to be analyzed by a process engineer or controls engineer.

Fig. 15-4 This panel shows many discrete devices as well as mimic panels showing process lines.

Meters show levels or flows of various devices.

Alarms are shown in grids of illuminated push buttons.
Fig. 15-5  Alarm panels were designed with discrete panels that lit or blinked with each alarm. Buttons were used to acknowledge each alarm point.

Fig. 15-6  Data was collected with recording devices similar to the above. Multiple points were individually recorded and studied.
Fig. 15-7 A handheld thermocouple readout device with paper recording output.

Fig. 15-8 Several discrete controller devices for process control. Each device is capable of solving a single or multiple loops of process data executing a PID formula and controlling the output of the control loop.
Fig. 15-9

Whatever the appearance of the outside of the panel, the inside many times looks similar to the panel shown at left. It is too easy for the panel to look like this after a short time even though the original plan showed a neat design with well-organized wiring layout. This is not just a rare bad example.

Fig. 15-10

The picture below is of a panel designed in 1980 for a chemical batching system. It should be seen as what ‘was’ and not as what today should be a good design. It cost approximately $100K then and was a divider between the operator room and the instrument control room for the process.
ISA-101 Human Machine Interfaces for Process Automation Systems

ISA-101 (officially ANSI/ISA-101.01-2015) is the HMI specification meant to describe proper development of an HMI panel. The specification is under development and should be referenced as it may have input for your HMI design.

Hardware to Complement the Software

The following Siemens panels are available for selection to replace the various panels above. Software allows the design of these ‘soft’ panels to be adapted to represent all the designs shown above. Siemens, which provides a complete offering of hardware operator interface units to complement the software requirements.

<table>
<thead>
<tr>
<th>Type of operation</th>
<th>KP8 PN</th>
<th>KP8F PN</th>
<th>KP32F PN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function keys (programmable)</td>
<td>8</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>Output type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LED color modes</td>
<td>5 (green, red, yellow, blue, white)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Typical service life</td>
<td>1,500,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short-stroke keys (in number of switching cycles)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light-emitting diodes (ON period in %)</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interfaces</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digital inputs / outputs</td>
<td>8</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Fail-safe inputs SIL 2 / SIL 3</td>
<td>– / –</td>
<td>2 / 1</td>
<td>4 / 2</td>
</tr>
<tr>
<td>PROFINET</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Functionality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pushbutton and lamp test</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree of protection</td>
<td>IP65 / IP20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 15-63 Siemens HMI
### Siemens HMI

#### Operating mode
- 4" Touch + Key
- 7" Touch + Key
- 9" Touch + Key
- 12" Touch + Key
- 3.5" Key 4" Key

#### Display
- Widescreen TFT, 65k colors, LED backlighting
- PSTN-ECD Black/White Widescreen TFT

#### Size (in inches)
- 4.3"  
- 7"  
- 9"  
- 12.1"  
- 3.5"  

#### Resolution (W x H in pixels)
- 480 x 272  
- 800 x 480  
- 1,280 x 800  
- 1,280 x 800  
- 240 x 80

#### MTBF (in h)
- 20,000  
- 20,000  
- 20,000  
- 20,000  
- 20,000

#### Front dimensions (in mm)
- 141 x 116  
- 214 x 158  
- 267 x 182  
- 330 x 245  
- 165 x 97

#### Operator controls
- Touch screen and tactile keys
- Touch screen and tactile keys
- Touch screen and tactile keys
- Touch screen and tactile keys
- Tactile keys

#### Function keys (programmable) / system keys
- 8 / 8

#### Usable memory
- 10 MB
- 10 MB
- 10 MB
- 10 MB
- 1 MB

#### Memory for options / recipes
- / 256 KB
- / 256 KB
- / 256 KB
- / 256 KB
- / 40 KB

#### Alarm buffer
- *

#### Interfaces
- Serial / MPI / PROFIBUS DP / PROFINET (Ethernet)
- USB host / USB device
- Slot for CF / MultimediaSD

#### Functionality (when configured with WinCC TIA Portal)
- Signaling system (number of messages / message classes)
- Process pictures
- Tags
- Vector graphics
- Bar charts / trend diagrams
- Faceplates
- Recipes
- Archiving / Visual Basic scripts
- PG functions

---

**Fig. 15-64 Siemens HMI**

---

**Fig. 15-65 Siemens HMI**
### SIMATIC HMI Mobile Panels

<table>
<thead>
<tr>
<th>2nd Generation</th>
<th>2nd Generation</th>
<th>2nd Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>KTP400F Mobile</td>
<td>KTP700 Mobile</td>
<td>KTP900 Mobile</td>
</tr>
<tr>
<td>4” Touch + Key</td>
<td>7” Touch + Key</td>
<td>9” Touch + Key</td>
</tr>
<tr>
<td>Widescreen TFT, 16 million colors, LED backlighting</td>
<td>TFT display (LCD), 64k colors</td>
<td>Display</td>
</tr>
<tr>
<td>4.3”</td>
<td>7”</td>
<td>9”</td>
</tr>
<tr>
<td>480 x 272</td>
<td>800 x 480</td>
<td>800 x 480</td>
</tr>
<tr>
<td>640 x 480</td>
<td>Resolution (W x H in pixels)</td>
<td></td>
</tr>
<tr>
<td>50,000</td>
<td>50,000</td>
<td>50,000</td>
</tr>
<tr>
<td>50,000</td>
<td>MTBF * back-lighting (in h)</td>
<td></td>
</tr>
<tr>
<td>194 x 166</td>
<td>248 x 172</td>
<td>307 x 201</td>
</tr>
<tr>
<td>248 x 195</td>
<td>307 x 224</td>
<td>Diameter 290</td>
</tr>
<tr>
<td>Touch screen and tactile keys</td>
<td>Touch screen and tactile keys</td>
<td>Touch screen and tactile keys</td>
</tr>
<tr>
<td>4 (with LED) / –</td>
<td>8 (with LED) / –</td>
<td>10 (with LED) / –</td>
</tr>
<tr>
<td>18 / –</td>
<td>Function keys (programmable) / system keys</td>
<td></td>
</tr>
<tr>
<td>4 MB</td>
<td>12 MB</td>
<td>12 MB</td>
</tr>
<tr>
<td>6 MB</td>
<td>User memory</td>
<td></td>
</tr>
<tr>
<td>4 MB / 512 KB</td>
<td>12 MB / 2 MB</td>
<td>12 MB / 2 MB</td>
</tr>
<tr>
<td>1,024 KB / 64 KB</td>
<td>Memory for options / recipes *</td>
<td></td>
</tr>
<tr>
<td>–</td>
<td>Alarm buffer</td>
<td></td>
</tr>
<tr>
<td>Interfaces</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>–/–/–/1</td>
<td>–/–/1</td>
<td>–/–/1</td>
</tr>
<tr>
<td>–/–/–/1</td>
<td>–/–/1</td>
<td>–/–/1</td>
</tr>
<tr>
<td>1/–</td>
<td>1/–</td>
<td>1/–</td>
</tr>
<tr>
<td>–/–/–/1</td>
<td>–/–/1</td>
<td>–/–/1</td>
</tr>
<tr>
<td>–/–/–/1</td>
<td>–/–/1</td>
<td>–/–/1</td>
</tr>
<tr>
<td>Serial / MPI / PROFIBUS DP / PROFINET (Ethernet)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB host / USB device</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slot for CF / Multimedia / SD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signaling system (number of messages / message classes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process pictures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tags</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vector graphics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bar charts / trend diagrams</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faceplates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recipes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Archiving / Visual Basic scripts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status / Control, diagnostics viewer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 15-66 Siemens HMI
Ch 15 Human Machine Interface
SIMATIC IPC677D

- Processor: Intel Core i3 6100T (2.3 GHz), 4 MB cache, 48 GB DDR4 RAM, 500 GB SSD
- Main memory: 8 GB DDR4-2666 SDRAM
- Operating system: Windows 10 IoT Enterprise
- USB connectors: 1 x USB 3.0, 1 x USB 2.0, 1 x USB 2.0 (with Touch screen only)
- Ethernet: 1 x Gigabit Ethernet
- Power supply: 24 V DC, 50...60 Hz, 140 W
- Display: 15" Touch, 60 Hz, 1280 x 1024
- Resolution: 1920 x 1080 pixels
- Interface: PROFINET RT over Ethernet
- Dimensions: 445 x 365 x 120 mm
- Weight: 10 kg

SIMATIC IPC477D

- Processor: Intel Core i3 6100T (2.3 GHz), 4 MB cache, 48 GB DDR4 RAM, 500 GB SSD
- Main memory: 8 GB DDR4-2666 SDRAM
- Operating system: Windows 10 IoT Enterprise
- USB connectors: 1 x USB 3.0, 1 x USB 2.0, 1 x USB 2.0 (with Touch screen only)
- Ethernet: 1 x Gigabit Ethernet
- Power supply: 24 V DC, 50...60 Hz, 140 W
- Display: 15" Touch, 60 Hz, 1280 x 1024
- Resolution: 1920 x 1080 pixels
- Interface: PROFINET RT over Ethernet
- Dimensions: 445 x 365 x 120 mm
- Weight: 10 kg

General features

- Resolution: 1920 x 1080 pixels (widescreen)
- Processor: Intel Core i3 6100T (2.3 GHz), 4 MB cache, 48 GB DDR4 RAM, 500 GB SSD
- Main memory: 8 GB DDR4-2666 SDRAM
- Operating system: Windows 10 IoT Enterprise
- USB connectors: 1 x USB 3.0, 1 x USB 2.0, 1 x USB 2.0 (with Touch screen only)
- Ethernet: 1 x Gigabit Ethernet
- Power supply: 24 V DC, 50...60 Hz, 140 W
- Display: 15" Touch, 60 Hz, 1280 x 1024
- Resolution: 1920 x 1080 pixels
- Interface: PROFINET RT over Ethernet
- Dimensions: 445 x 365 x 120 mm
- Weight: 10 kg

MTBF backlighting up to 50,000 h

Ch 15 Human Machine Interface
The hardware above shows the variable nature of hardware panels available for use. The cell phone is also available with an app installed that communicates with the system software. Each device gives a level of access to the PLC program and give visual feedback to the engineer or plant personnel.

**The Software for Development**

We begin the software development of the three programs. Two programs exist in the A-B platform and one is available from Siemens. We will also discuss an OPC-UA application for a total of four different programming platforms.

From a white paper featuring Siemens’ new TIA Portal design, the concept of design of tags is used:

**Common Tags**

The most obvious advantage of using TIA Portal is the universal accessibility of data tags. Tags created in any tool for any device are automatically and immediately accessible to other devices. If, for example, a user creates a new tag in the PLC to measure temperature, that tag is automatically created in the operator panel at the same time. This saves valuable engineering time compared to conventional methods that require the tag to be created in each device. Should the user wish to modify that tag’s properties, he or she can change parameters from whichever tool is currently being used just by changing the view. In any case, the data is universally accessible.
Allen-Bradley’s New View Designer

Allen-Bradley recently released a new graphic HMI Development Environment. It is found on Studio 5000 as ‘View’.

We can open it and use the Getting Started tab under ‘Help’ to begin using it. However, here are some of the more helpful start-up buttons used for programs we require.

First is the application program linking the HMI design to the program resident in a PLC. Note, there can also be an emulator PLC which runs in the PC instead. We will not use the emulator for project development here but it may be useful in program development when hardware is not present or not present yet.

The following shows the programming environment for View. It is similar to the other two packages introduced previously with some interesting differences.
We are always interested in the design of a button. A-B gives some classic-looking buttons but we would like to first use a rectangle as our first button. This is possible using the rectangle from the toolbox at the left. The rectangle then is tied to an event with the button set to set the chosen tag to 1 when pressed and reset to 0 when released. This is shown below:

We would like to also set a multi-state button similar to the one found in RSView Studio ME.
This button is created by setting state to 2 in the Animations tab. This design is demonstrated below:

The RSView ME Stand-Alone Application

RSView ME is one of a number of software programs furnished to collect and display information from the factory floor, first to the operator and then to the business itself.

To open RSView Studio and gain access to RSView ME, open the program and click on the New tab. For Application name, the following example program used test.
The A-B product uses RSLinx Enterprise. This means that RSLinx Enterprise has the ability to allow this software package to browse directly for the tag database and link existing tags and not requiring a tag created in the HMI to complement the tag in the PLC.

When starting a new application, select Objects from the main menu. Notice the types of objects selectable. Under each selection are sub-menu selections. In the case of Push Buttons, the four sub-menu selections are:

- Momentary
- Maintained
- Latched
- Multistate
- etc

The pushbutton is a good example of a type of object that can be used for a number of different operations. For example, momentary is the most used button with an exact simulation of a real button in which the operator pushes the button and a ‘1’ appears in the data address of the device. When the button is released, the button returns to a ‘0’. As simple as this is, it is a very profound device in that the device continues to write a ‘1’ to the data tag until the device de-activates and the device writes a ‘0’. Other button types have various other characteristics and these characteristics allow the programmer flexibility in programming of the various functions surrounding the logic of the pushbutton. More will be discussed later regarding some of the button types.

Remember that input and output bits used for a HMI are tags with internal addresses, not hard-wired inputs or outputs. For A-B, the device should have a tag referring to an internal location. For Siemens, the address should have an M prefix.
RSView Machine Edition (ME) is designed for the machine-level HMI and supports operator interface solutions for the monitoring and controlling of individual machines or for small processes.

The system tree at left shows the graphical application and is organized by area. These include:

- System
- HMI Tags
- Graphics
- Alarms
- Information
- Logic and Control
- Data Log
- RecipePlus

For the simple assignments of this text, the user can simply add a display and create the appropriate graphics on the form. To test or run the form requires a link to a PLC. The procedure for linking to a PLC is included next. The tags created for the PLC are used in the graphic and any button or indicator will reference the PLC tag.

Fig. 15-70 RSView Project Tree

Fig. 15-78 Activation of Factory Talk RSLinx Enterprise
Then, configure the local directory per administration sign-in configuration of your pc.

![Reconfigure FactoryTalk Local Directory](image)

Fig. 15-79  Administrative Signin User Name/Password

After this configuration is complete, you may proceed with your HMI application and connect the HMI screens to a PLC via RSView Enterprise as described above.

Creation of tags is accomplished either in the program tag or controller tag areas. Either may be accessed by FactoryTalk. The type of tag must be considered since numeric tags must be scaled and configured for proper display. Boolean tags must also be configured properly.
An Example RSView Display

The suspension bridge at Niagara Falls was started by flying a kite with a string attached across the Niagara River. When wind conditions were favorable, the kite was flown across the river. Then a string was attached to the thread and a bridge was the eventual result. Likewise, programs in this chapter can be started with small threads and then expanded. It is best to get a simple device such as a button programmed and fully working and then adding the rest of the project after the button has been proved to thoroughly work in all modes.

The following is an example much like that in Ch. 7 for the Siemens’ HMI. The example shows the first use of A-B’s RSView linking a simple button to the PLC. This is much like the kite example for the bridge at Niagara Falls. If the process is completed for one element successfully, the remaining portions of the HMI may be more confidently programmed with success.

The project is begun with the following screen present. The project name is ‘test1’.

![First Screen with RSView Studio](image)

**Fig. 15-82** First Screen with RSView Studio
Ch 15 Human Machine Interface

Fig. 15-83 Selecting the Main Display

Fig. 15-84 Building a Momentary Push Button
Fig. 15-85 Establishing Communication with RSLinx Enterprise

RSLinx Enterprise Configuration Wizard

A configuration file contains information about devices, drivers, and networks. Select the source for this offline configuration file:

- Create a new configuration.
- Copy an existing configuration from a previously created project.
- Copy the configuration that is currently running on this workstation.

Fig. 15-86 After Selecting Communication Setup under RSLinx Enterprise
Fig. 15-87  Click ‘Add’ for Device Shortcut, Enter Name ‘test1’

Fig. 15-88  Highlight the Ethernet Device – PLC
When APPLY IS HIGHLIGHTED - CLICK IT
Fig. 15-89 The Local Path is Determined for the Application

Fig. 15-90 The Local Path is set with ‘OK’
Fig. 15-91  Back at the Ranch (I mean Button)

Fig. 15-92  Working with the Button – Tab 1 - General
Fig. 15-93 Establishing the PLC Connection to the Button

Fig. 15-94 Click ‘Tag’ and the Link to the PLC ‘test1’ is Displayed
Fig. 15-95 Choose ‘Refresh All Folders’

Fig. 15-96 PLC Tags are Shown in the Online Tags
The Tag shows the Tag/Expression with the PLC Tag

Color and Captions Modified under States Tab
It is time to try the button with the connection to the PLC. Run the display by choosing the triangle in the command line. Test the screen with the button in the off and on position. View the bit in the PLC online program.

Use the triangle for testing single screens. To run all screens together in local mode, run the little man. There may be problems with this operation as it invokes a program called ‘KEP’ that may interfere with other applications. Be careful when running the little man.

While building a screen or series of screens requires more instruction, we leave the A-B software to discuss Siemens’ HMI interface.

*Siemens Win CC*
Work Area

In the work area we edit the objects of the project. All elements of WinCC flexible are arranged around the work area. In the work area, we edit the project data either in the form of tables (for example, variables), or graphically (for example, a process display). The upper part of the work area contains a symbol bar. Here, the font, the font color or functions such as Rotate, Align, etc. can be selected.
The Root Screen

Tools

The tool window provides a selection of objects that you can insert in your screens; for example, graphic objects and operating elements. In addition, the tool window contains libraries with assembled library objects and collections of picture blocks. The objects are dragged and dropped into the work area.

Properties Window

The properties of objects are edited in the properties window; for example, the color of screen objects. The properties window is available only in certain editors. The properties of the selected object are displayed in the properties window, arranged according to categories. Value changes become effective as soon as an entry field is exited. If you are entering an invalid value, it is color-enhanced.

By using QuickInfo, information is provided about the valid value range, for example. In the properties window, animations and events of the selected object are configured also; here, for example, a screen change when releasing the button.
Details View

In the Details view, additional details about the object highlighted in project navigation are displayed.

Operating Screens and Connections

A screen can consist of static and dynamic components. Static components, such as text and graphs, are not updated by the control system. Dynamic components are connected to the control system and visualize current values from the control system’s memory. Visualization can be in the form of alphanumerical displays, curves and bars. Inputs at the operator panel that are written to the memory of the control system are also dynamic components. They are interfaced with the control system by means of variables. Initially, we are only generating a screen for our conveyor control.

Root Screen or Start Screen

This screen was set up automatically and defined as start screen. Here, the entire plant is represented. Buttons can be used to do the following: switching the operating mode between
automatic and manual; starting and stopping the conveyor motor, and exchanging the box. The movement of the bottle on the conveyor belt and the fill level of the box are represented graphically.

Using button F6, we are jumping to the system screen:

![The System Screen](Fig. 15-116)

**Connections to S7 Control Systems**

In the case of operator objects and display objects that access the process values of a control system, a connection to the control system has to be configured first. Here we specify how the panel communicates with the control system, and with which interface.

In Project navigation, click on Connections. Because of the settings in the hardware configuration, all parameters are already set.

![Establishing the PLC Connection](Fig. 15-117)

The IP address has to be assigned to the panel also. By means of Accessible devices, read out the panel’s MAC address, or read it on the back of the panel.
Assigning the IP Address

After inputting the MAC address, the IP address can be assigned under Online & diagnostics. The panel has to be in the Transfer Mode in this case. Change the IP address to **192.168.0.5** for all applications.

**Note**
The IP address can also be checked or entered on the panel in the system control under Control Panel at Profinet.

**Configuring the Root Screen**
Clicking on the button “System screens” displays the system screen. We want to transfer the function of the button System screens to the function key F6.
Select System screens and in the Properties window below copy the function Activate screen at Events Release.

**Fig. 15-120**
A Start-Up or System Screen

**Function Key F6**

Select function key F6 and in the Properties window below, insert the function Activate screen at Events Release key. Then, delete the text field in the center, and delete or remove the button System screens.

**Fig. 15-121**
Defining a Function Key

The yellow corner on the F6 key indicates that the key is configured.
Configuring the Buttons Automatic and Manual

Drag a button into the work area of the root screen.

Fig. 15-122 Configuring a Button

At Label, enter Automatic. **Caution!** Don’t press the input key; otherwise, a second line is generated.

Fig. 15-123 Adding a Label

Under Layout, enter position and size.

Fig. 15-124 Modifying a Label’s Size
Under Events Press select the function SetBitWhileKeyPressed under Edit bits.

Then, click on the field Tags (input/output) and using “…” button, open the tag window. Here, it is also possible to access the interface declaration of data blocks. As tag, select auto from the Conveyor_DB [DB1].

Now, the button is to flash in the automatic mode, and change color. With a double click, select under Animations\New animation Appearance.
As tag, select `automan` of `Conveyor_DB [DB1]`.

The button is to change color in the automatic mode; that means, when the variable `automan` has the value 1. For the color change to become visible, change the foreground color at Appearance to White and the background color to Green. At Flashing, say Yes.

Copy and paste the button `Automatic`. Place the inserted button under the Automatic button.

At Label, enter `Manual`. **Caution!** Don’t press the input key; otherwise, a second line is generated.

Under Events Press, select `man` from `Conveyor_DB [DB1]` as tag. The variable has to be selected, because only then will the new HMI tag be generated.
The button is to change color in the manual mode; that means when the variable `automan` has the value 0. For the color change to become visible, change the foreground color at Appearance to White and the background color to Blue. Set Flashing to No.
Visualization is opened in the RT simulator.

![RT Simulator](image)

**Fig. 15-136 Running the Simulator**

Test the project of the conveyor control. The automatic or manual mode is now pre-selected on the panel.

![Start and Stop Buttons](image)

**Fig. 15-137**

**Start and Stop Buttons**

Next, we configure the start and stop buttons. The Start button is created exactly like the automatic and manual buttons. The Stop button has a break contact function and has to remove the signal when operated.

- Create the Start button
- Set the background color to Green
- Under Events Press, select under bit editing the function SetBitWhileKeyPressed.
- Then select the tag on in Conveyor_DB [DB1].
Next, do the following: Create the Stop button. Set the background color to Red.

Under Events, at Press, select under Bit editing the function ResetBit and at Release the function SetBit with the tag off in Conveyor_DB [DB1].
Fig. 15-142  Screen with Start and Stop Buttons
Allen-Bradley Button Configuration

The choice of button type indicates the type of function desired. There is no need to program both press and reset but rather only type and the function is performed.

Fig. 15-143 Choice of Button Type Determines Function

The tag or expression may be programmed for the value as well as for an indicator. The indicator actually is a second button with the state desiring to be displayed. This value may be the same address as the value entry or a second address. To perform a similar function, Siemens may require two buttons overlaying each other.

Fig. 15-144 Connections Tab on Push Button Function
Before we test the visualization, first another change has to be made in the Step7 program. In OB1, remove the assignment S3 and S4 when calling FB1.

Save and load the modified program.

Load the configuration to the panel, and test the Start and Stop buttons.

![Fig. 15-146](image)

**Inserting Graphics from the Graphics Folder**

In the tool box under Graphics, open the directory tree `WinCC graphics folder\SymbolFactory 256Colors\Conveyors, Misc`.

Drag and drop the graphic of the conveyor belt to the root screen.
In the tool box under Graphics, open the directory tree WinCC Graphic folder\SymbolFactory 256 Colors\Food.

Then drag and drop the picture of the beer bottle in the root screen.
Change the size and the position of the bottle.
Note

All screen objects have to be within the work area (320x240 pixels).
Control Program for Simulating the Bottle Movement

To simulate the bottle movement and the bottle sensor, we create a new block. The FB2 (simulation) below with tag declaration and networks consists of a counter that, through a start signal, always counts up from 0 to 50.

In Network 1, the CTU (count upward) is inserted as multi-instance. In Network 2, a bottle sensor pulse signal is read out when the count 50 is reached. This simulates when a bottle leaves the conveyor.

![Network 1 and Network 2 diagrams](image-url)

**Fig. 15-151**
Configuring the Bottle Movement

Highlight the bottle and select under the tab Properties at Animations - Horizontal movement (double click).

Fig. 15-154
As variable, select COUNT_VALUE of the Simulation_DB (DB2). For range, enter from 0 to 50.

Change the bottle’s target position up to the conveyor end X150.

Fig. 15-155
Allen-Bradley Animation

The Allen-Bradley animation proceeds much the same as Siemens in that the device edited may be animated in a number of ways including horizontal position. The position is a function of a number in a location which is monitored and the beer bottle moved accordingly.

Fig. 15-156

Fig. 15-157
In the project window, select the HMI tags.

![Project window with HMI tags selected](image1)

Drag the slider in the window to the right in order to get to the column Acquisition cycle. Set the acquisition cycle of the HMI tag to 100ms.
Then save the project, load it to the panel and test it.

After 20 bottles, the conveyor motor stops. The bottle counter has to be reset before the next start.
Resetting the Bottle Counter

Drag a button into the root screen.

As text, enter Exchange beer case and adjust the color Position & size to the button.
Under Events Press, select under bit editing the function SetBitWhileKeyPressed. Select the tag reset_counter from Conveyor_DB [DB1].

Set the acquisition cycle of the new HMI tag to 100ms. Then save the project, load it to the panel and test it.
Drawing the Beer Case

Draw a rectangle with a transparent background. Enter the width of the border, the position and size.

Fig. 15-166
Draw a vertical line at a distance of 30 pixels.

Fig. 15-167

Note

Although the measurements of the line are correct, it is drawn beyond the rectangle. Change the form of the line ends to flush, and shorten the line by one pixel from 150 to 149.
Draw a horizontal line spaced at 30 pixels

![Diagram of a human machine interface with buttons and a graph]

Fig. 15-168

With copy and paste, create the remaining lines spaced at 30 pixels.
Highlight the beer case by dragging a border around it with the mouse.

![Fig. 15-169](image)

In the menu Edit select the function Group.

![Fig. 15-170](image)
We don’t want to display the rectangle and the lines when the beer case is exchanged. At Rectangle_1 and at the lines, generate the animation Visibility using the tag Conveyor_DB_reset_counter at value 1 Invisible. For the lines, the animation can also be copied and pasted.

Then save the project, load it to the panel and test it.
Drawing Bottles in the Case

Enlarge the view and draw a circle in the lower right field of the box.

Fig. 15-173

Fig. 15-174
Draw a second circle.

Fig. 15-175

Group the two inserted cycles.

Fig. 15-176
At Circle_1 and Circle_2, generate the animation Visibility with the tag
Conveyor_DB_IEC Counter_0_COUNT_VALUE value range 0 to 19 Visible.

Fig. 15-177

Fig. 15-178
Copy and paste the bottle.

For both circles, under Visibility change the value range of the tag `Conveyor_DB_IEC_Counter_0_COUNT_VALUE` to 0 to 18 Visible.
Animation with visibility is available with Allen-Bradley as well. It is shown below with visibility as a property with a tag or an expression with tags available to provide logic for the visibility of a device, in this case, a circle.

Fig. 15-181

Fig. 15-182
The expression editor gives the ability of the programmer to add logic to select the visibility of an object.

Copy and paste the individual bottles.

At the animation Visibility of both circles decrease the value to by 1.
The last bottle has the value range from 0 to 0.
Set the acquisition cycle of the new HMI tag to 100ms. Then save the project, load it to the panel and test it.

Fig. 15-185
OPC and Visual Basic

OPC is short for OLE for Process Control. OLE is short for Object Linking and Embedding. OPC strives to connect industrial automation with software programs (sometimes referred to as enterprise systems) to share data. OPC is an open system with shared standard approaches. Currently seven standards comprise the OPC system. OPC Foundation is the organization to oversee the adoption and creation of these standards.

OPC is a program that works with OLE (Object Linking and Embedding) a technology developed by Microsoft for the purpose of embedding and linking to documents. OLE stands for OLE for Process Control. Included in OPC are devices that provide aid to the programmer in areas such as:

- **Alarms and Event**

- **Redundancy**: Industrial applications frequently require high availability and reliability that can be easily achieved by implementing communication redundancy

- **Client Server Architecture**: The client/server nature of OPC enables users to architect connectivity solutions that would previously be prohibitively expensive.

- **Historical Data Access**: OPC Historical Data Access (OPC HDA) specification is used to archive and retrieve process data. Also included are trends reports using the OPC HDA client applications.

DDE and OPC are integrated into the Allen-Bradley product through RSLinx. See the opening tabs for these applications in RSLinx below:

Microsoft’s Access and Crystal Reports are examples of the power of using OPC with Visual Basic.
**Graphic Standards From Windows Standards**

Graphics come in many flavors but not all file formats are suitable for all purposes. How do you know which is best? Some standards exist for a specific company. Other standards for graphics are general and used by most. Some of these are:

**Practical Design of Logic with HMI**

From early in the chapter, A-B’s design allows for a Push Buttons with the selection Multistate. The memory circuit can be turned on or off from the multistate button. The better memory circuit can also be turned on or off from the program. The memory circuit then must be able to report to the HMI the present state. The program for this function is left as an exercise.

![Diagram of memory circuit control](image)

The use of multistate buttons to provide this logic is useful but not necessary. The use of a single button on the screen is an advantage in that one button can be used to turn on the memory, turn off the memory and display the present state of the memory. All three functions can be accomplished with a single button. All the time in advertising, we hear the ad for buy one, get one free or the ‘two-fer’ ad. This is a real ‘three-fer’. Buy one button and get the ‘start’ button, the ‘stop’ button and the ‘indicator’ light in the same button.
Where to Put the Logic

The following conveyor system has five outputs, lights for percent complete of packages going down conveyor 1 to conveyor 2. Write a program to turn on these lights based on the fact that packages must pass photo-eye 1 to enter the storage area and pass photo-eye 2 to exit. This program was solved in Ch. 8 using greater/less-than statements and discrete outputs. It is possible now to turn on these outputs in the HMI program with no statements in the Ladder other than the Up-Down counter.

You may find it easier or better to provide the logic in the HMI or Ladder. There is usually a preference in most companies for one or the other or you may decide for them.

Consider Multi-State Indicators for the application above using RSView Studio as an example:
Example: Expressions that return numeric values

For these examples, assume tag1 = 5 and tag2 = 25.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tag1</code></td>
<td>5</td>
</tr>
<tr>
<td><code>tag1 + tag2</code> (arithmetic operator)</td>
<td>30</td>
</tr>
<tr>
<td><code>~tag1</code> (bitwise operator)</td>
<td>-6</td>
</tr>
<tr>
<td><code>SQRT(tag2)</code> (mathematical function)</td>
<td></td>
</tr>
</tbody>
</table>

Example: Expressions that return true/false values

<table>
<thead>
<tr>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tag1 &gt; 20</code> (relational operator)</td>
<td>1 (true) if tag1 is greater than 20</td>
</tr>
<tr>
<td></td>
<td>0 (false) if tag1 is less than or equal to 20</td>
</tr>
<tr>
<td><code>Industry\Valve AND Municipal\Valve</code></td>
<td>1 (true) if both valves are open</td>
</tr>
<tr>
<td></td>
<td>0 (false) if one or both valves are closed</td>
</tr>
</tbody>
</table>
Example: Controlling visibility with If-Then-Else logic

To create a graphic object that is to be visible only when \( tag1 \) exceeds a specified value

1. Draw the object.
2. In the Visibility animation dialog type the expression:

   \[ \text{If } (tag1 > 55) \text{ Then 1 Else 0} \]

3. Specify that the object is to be visible when the expression is true.

From 1973, an HMI That Worked

In 1973, I was intensely involved in a project involving an HMI for three stations for a glass manufacturing line. The program was written entirely in assembler on a 16 k machine (IBM System 7). The HMI portion was to be six screens for the entire line including the three stations. What happened to that project was the complete explosion of need for a manual back-up that could over-ride the automatic system originally envisioned. The only response was to create a manual back-up system that expanded the original six screens to 66 screens. Along the way, a complete diagnostic system was also developed since electricians and operators could not be expected to keep the line operational without good data concerning the sensors on the machine and which if any were not operating correctly. The eventual HMI program consisted of about 23k assembly statements found in 66 individual over-lay programs running in a 384 word over-lay area of the computer.

The program was a tree-designed system with operators being asked a series of choices from which they successively moved further out a branch of the tree. In the end it worked and was accepted by the operators and plant engineers. It took way too long and enveloped my time for more than two years. Hopefully the lessons learned from experiences such as these would be helpful for future engineering efforts.

One of the two learned was the conflict between production and the hypothetical original design that insisted on complete automatic control. The manufacturing environment was not ready for the complete automation of the theorist. If a production supervisor found it more advantageous to have the destination of glass be on a take-off point on the right side instead of the left, he wanted that flexibility. The automatic system did not give that kind of flexibility and he refused to use the system until developed to his needs. The other lesson was that of complete diagnostic reporting. The operation was not going to succeed until every switch was operational (for glass tracking) and these switches were constantly falling into disrepair due to the moving glass. It was only after a dedicated line on each display showed the active status of any broken switch and its exact location was the tracking system allowed to function.

From 1998, an HMI System Divided

In 1998, I worked on a system for a steel company that saw the dividing of the project into multiple assignments with four engineers working simultaneously on two PLC/HMI programs. The PLCs were Modicon and the HMI was Wonderware. In this project I was responsible for the PLC program for process control. My counterpart worked on the HMI portion from India. Two individuals programmed the steel movement portion of the project from their office in Toronto, Canada. The most difficult portion was the HMI portion for the steel movement. It
involved a number of rules for the handling of the steel. These rules were total written from the Wonderware software platform.

The rules for the handling of the steel were similar to those of a baker with an oven big enough to handle multiple cakes with various baking times and temperatures. If a cake was in the back of the oven and was ready to come out, could a new cake be inserted in its place given the time overlap of a second or third cake presently in the oven or ready also to be placed in the oven? The rules allowed for proper bake time and temperature for each individual cake while controlling the temperature of the overall oven.

**Sophisticated HMI Design**

The design of an HMI panel or group of panels is very important and this chapter has opened only a small bit into the proper development of the HMI screen. Microsoft has developed a template for design of screens since they developed the original Windows OS and its successors. Flight training simulators also have been developed along those principles of excellence in HMI design since they are used to train pilots in proper use of the instrument panels prior to risking their lives and those of their passengers in air travel.

The following book and succeeding books as well as white papers by these authors give a good next step in the proper design of the HMI panel.

**The High Performance HMI Handbook** by Bill Hollifield, Dana Oliver, Ian Nimmo and Eddie Habibi

From the white paper:

“Five areas are primary in the successful design of a good HMI screen system. They are:

- Situation Awareness
- Using Color Effectively
- Interpreting the data
- Depicting Device State
- HMI Display Organization” and

“Applying SA terminology to HMI Graphics:

- Level 1 SA – P&ID representation with Live numbers
- Level 2 SA – Provide the operator with the relevant information they require to understand how the plant is operating
- Level 3 SA – Provide trending data so that the operator can extrapolate the plant’s performance to the future
- Level 2 and Level 3 SA reinforces the operators’ mental model of the plant or process”

What is situation awareness? We see it as the ability to design a panel that lets the operator (who may have been on the job only for a day or two) the best ability to control the machine or process without causing harm. This is a great responsibility!

While you may not immediately become a PLC programmer, your employer may more quickly thrust you into HMI design with the assumption that you can’t do as much harm in HMI design as in PLC design. This may not be true!
Summary

This chapter is the only chapter solely dedicated to the HMI graphic panel. In Ch. 7, we had a short tutorial involving getting the Siemens’ HMI attached to the S7-1200. That was an introduction to the HMI panel and useful for encouraging students to use the panels instead of wiring to buttons and lights. This chapter expands on that first experience in that both Siemens and A-B are discussed as well as types of product.

The basic panels for both manufactures are introduced and explored. Buttons as well as other devices are built. Some examples of how to use various graphics are included as well. The chapter ends with a discussion of graphics standards and a common problem that I commonly refer as the ‘three-fer’ button. The chapter is not meant to be an exhaustive study of HMI panels but as a starting point for students needing to learn some graphics before launching their careers.

While this chapter begins the broad development of HMI panels, the design of panel interfaces and screen interfaces continue in subsequent chapters, especially the chapter on motion and the chapter on pid control.
Lab 15.1  Revisit - The Cash Register

The basic lab is copied from Chapter 7 as follows:

Design a simple cash register similar to one found at McDonald’s or Burger King. To do this, determine a menu of five or six items from the restaurant. Also, include a Total button or a clear button or possibly both. Also, include a means for backing out of a mistake without starting over from zero. Display the cost of the total order in the PLC at an address in the data table. Use floating point math and you are encouraged to do so.

For example:

<table>
<thead>
<tr>
<th>Whopper Combo</th>
<th>Whopper</th>
<th>Cancel Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whopper Dbl Combo</td>
<td>Fries</td>
<td>New Order</td>
</tr>
<tr>
<td>Whopper Jr Combo</td>
<td>Drink</td>
<td>Total/Tax/Optional</td>
</tr>
</tbody>
</table>

Find the approximate prices from a McDonald’s or Burger King for the items chosen. When an item is entered, its count is incremented automatically by one. If a button is entered multiple times, the count is incremented to display the total count. If a mistake is made, the attendant must be able to back up at least one entry and erase the last item or decrement that item by one.

Hints to the base lab:

Notice that counters may be referenced as either Count Up or Count Down. If the count is counting up, the count is incremented in rung 0000. If the count is counted down, the count is decremented in rung 0001. Individual inputs are used to increment each product choice. However, to decrement the count, a separate button labeled “Cancel Last” is used. This button must remember the last product chosen and decrement that item. Use the logic in chapter 7 “Relay Instructions” to remember when a button was pushed.

Use the Count Up/Count Down logic for holding active counts for the various items in the cash register.

Make the following changes for the application:

1. Display the total price for the order on the screen. Use Floating Point numbers where possible. Display totals in $xx.xx format.
2. Add a second screen to allow the manager to change base prices for each item. Do not include a password to move from screen to screen.

3. Include a button to add 6.25% tax if not “To Go” for the order.

4. Include a ‘live’ count of the number of each item ordered.

5. Create means for going from Screen 1 to Screen 2.

6. Screens should resemble the following for **Lab 15.1**:

![Fig. 15-196](image1)

![Fig. 15-197](image2)

**Lab 15.2** Build the Conveyor application as described above for Siemens. Then build the same application for A-B. Compare the two. You will need to write PLC logic to move the elements and increment counts. You do not need to copy the programs included but may write your own programs.
Questions

1. How would you build the following pushbuttons in Siemens? Describe:
   - Momentary
   - Maintained
   - Latched
   - Multistate
   - Interlocked
   - Ramp

2. Describe how to demonstrate the flow of a liquid of red color through a series of pipes. Several hand valves are in the path of the flow. How would one describe this graphic with A-B, Siemens?

3. You are assigned the task of describing a ride at an amusement park graphically. The ride is a zip-line. You may place sensors at the beginning and end of the ride. Describe the graphic using A-B and then Siemens controllers and HMI.

4. For the following device, what is the animation property used to move the clock’s hand.

5. Why would one design a PLC and HMI system with OPC and Visual Basic rather than one of the packages described in this chapter?

6. Write code to accomplish the memory circuit in Fig. 15-188 using A-B, Siemens.

7. Describe how the programmer would animate the bottle on the conveyor moving right to left for Allen-Bradley, for Siemens.

8. Describe how A-B and Siemens shows each bottle being entered into the case of bottles.

9. When designing the Cash Register program, what differences were noted between how A-B and Siemens handle math data types?

10. Why is it important that a memory circuit such as Fig. 15-188 be able to be programmed?
11. Using either A-B or Siemens ladder logic, write a cash register program that uses an accompanying HMI program with only three items [hamburger, fries, drink]. Include logic to calculate the total price ignoring tax. Ignore the ‘cancel last’ button and combo logic as well.

12. Using A-B or Siemens ladder logic, write a program that could be used to show the movement of a bottle across a conveyor belt and populate a case. Use a case of 12. Use a start and stop button to start the operation. Use a reset button to set the bottle counter in the case back to 0 and allow another operation to fill the case with bottles.

13. If you were moving a program from real pushbuttons to an HMI, what one thing must you do?
Setting the PG/PC Interface for Runtime Simulation

In order to establish a connection between runtime simulation at the PG/PC and the S7-1200 CPU, first we have to set the PG/PC interface to TCP/IP.

<table>
<thead>
<tr>
<th>No.</th>
<th>How it’s done</th>
</tr>
</thead>
</table>
| 1   | Open the system control  
|     | • with "Start > System control"  
|     | (start menu for the simplified access to the programs under Windows XP)  
|     | • or with "Start > Settings > System control"  
|     | (for the classical start menu as in earlier Windows versions). |
| 2   | Now double click on the icon  
|     | "Set PG/PC interface" |
| 3   | In the tab "Access Path", set the following parameters:  
|     | 1. For the access point of the application, select from the drop down menu "S7ONLINE [STEP 7]".  
|     | 2. In the list of Interface Parameter Assignment Used, highlight the interface "TCP/IP(Auto) -> with your network card that is connected directly to the panel and the control system; for example, Intel(R) PRO/100 VE.  
|     | 3. Then click OK and confirm the message that follows with OK also. |

Starting the Configuration in Runtime

Click on the button Start runtime.
Alternately, the following must be set up to get the PID program to run properly with the HMI simulate mode:

Click on the SetPG/PC Interface box above:
We are going to use the Siemens program TIA Portal V14 in order to run the program given for the ball-in-tube lab. This program will be used to download the PLC program but not the HMI program.

Choose the third of the Broadcom choices. Click OK.

This allows the Siemens program to run the HMI program in simulate mode. Then download the program to the PLC. Do not download the HMI program since we do not have the HMI to download to.
Then click on the HMI’s Screen, Screen_1. Notice the Start Simulation button turn blue. It now allows the student to run the HMI via simulation mode from the screen of the pc.

We need to find the variables that need to be written to the historical data. Click on Laser Input and get the tag Laser_Percent. This tag is one to be written to the historical data logger.
Next, go to the Historical data tag under HMI tags. Set the variables from the HMI screen above that are to be saved. Fill in the appropriate fields and start the historical logger. This circular file will contain the data from the analog data saved. Then run the program and run the Historical data logger. It is a circular file and will wrap around after the table fills up.