## Contents

**Introduction** ......................................................... 3  
**User-Configurable Features** .................................. 3  
**Open System Features** ........................................ 5  

**System Components** .............................................. 7  
  Engineering Workstation (EWS) ............................... 8  
  Operator Workstation (OWS) ................................. 9  
  Field Control Unit (FCU) ...................................... 9  
  I/O Subsystem .................................................. 10  
  SCADA Data Server (SDS) ................................. 10  
  Process Historical Archiver (PHA) ....................... 10  
  Summary ..................................................... 11

**Project Configuration** ........................................... 13  
  Architecture Configuration ................................ 14  
  Device Diagram Configuration ............................ 18  
  Scripting ....................................................... 29  
  Graphic Editors ............................................... 30  
  Alarm and Logging Configuration ........................ 33  
  Security Configuration ..................................... 34  
  Command Windows and Group Displays ............... 35  
  Help System .................................................. 36

**Operator Interface** .............................................. 37  
  Project Graphics Screens .................................. 39  
  Command Windows and Group Displays ............... 40  
  Monitoring and Acknowledging Alarms ................. 41  
  Archiving ..................................................... 42  
  Logging and Trending ....................................... 43  
  Device Status and Diagnostics .......................... 44  
  Network Diagnostics ....................................... 45  
  Connecting to Other Software ............................ 46

**Communicating with Field Devices** ....................... 47  
  Field Control Unit (FCU) .................................. 47  
  SCADA Data Server (SDS) ............................... 48

**About Control Systems International (CSI)** ............ 49
Introduction

UCOS is a complete control system solution. It includes graphical development software, a graphical human-machine interface (HMI), and a logic processor — all based on User-Configurable, Open System standards.

The product’s most important features include the ability to:

- Use any traditional control system architecture, such as DCS, SCADA, PLC/HMI, or any combination thus achieving a high degree of configurability along with low cost and easy maintenance.
- Provide totally configurable regulatory, discrete, and sequential control.
- Use industry-standard hardware available from numerous suppliers, including more than 40 I/O families from more than 20 manufacturers.
- Provide connectivity to thousands of Windows applications through the use of Windows 2000/XP.
- Provide development software that lets you design, configure, test, and document projects faster and easier using patented, award-winning techniques.

UCOS is not an HMI stacked on a PLC. UCOS is a tightly coupled project development and run-time software product that lets you graphically configure object-oriented control logic and simultaneously generate HMI graphics. One object-oriented database stores both the control logic and graphics data for each project.

The UCOS development and run-time environments turn any PC into a scaleable workstation. In addition, the field control unit directly scans and controls multiple brands of industry-standard I/O, and the field data server allows the system to exchange data with most other software and hardware.

UCOS includes built-in diagnostics and component redundancy at all levels, if desired.

As a consequence, UCOS can significantly reduce the cost of developing, implementing, maintaining, and expanding a control system.

The following sections describe in more detail the concepts that underlie the user-configurable and open-system features of UCOS.

User-Configurable Features

Traditionally, control engineers choose configurable, DCS-based systems primarily for the implementation of regulatory control. They choose PLC-based systems primarily for the implementation of sequential control. They choose SCADA systems primarily to integrate a large number of remote locations and concentrate the logic processing at a central master station.

UCOS combines the best of all these architectures.

Like a DCS system, UCOS creates sophisticated regulatory control schemes that require little or no programming. Like a PLC system, UCOS provides flexibility in customizing every aspect of a project. Like a SCADA system UCOS integrates remote and master stations.

However, unlike these traditional technologies, UCOS creates regulatory, discrete, and sequential control schemes using one integrated development tool. This single tool addresses every aspect of the project:

- Configuring all system hardware, including I/O cards and tag assignment to points on cards
- Developing control logic
- Building HMI graphics and other displays for operators
- Documenting the entire project

…and all other aspects of a project.

UCOS eliminates the need to piece together a control system from multiple software packages that may or may not be designed to work together.

One integrated configuration tool does all the work and can eliminate the need for extensive programming. This integration allows UCOS to support a unified approach to regulatory, discrete, and sequential control – regardless of the architecture.
Introduction

For example, UCOS device objects provide all the capability of function blocks by incorporating tag definitions, faceplates, programming symbols, run-time graphics, and run-time dynamics. All this is immediately available when a device object is placed into a device diagram.

UCOS employs object-oriented technology to replace traditional function block and rung ladder logic programming.

![Figure 1 Relationship of Device Diagramming to Traditional Control Programming Techniques](image)

User-configurable devices provide a unique structure and approach for building regulatory, discrete, and sequential control for a device. This structure is an extension of state logic concepts. These concepts call for a device’s control logic to be organized into fundamental categories, such as modes, faults, states, alarms, and controls.

This device structure is supported by event-based, logic-solving capabilities in the field control units (FCUs) — the UCOS equivalent of the controller component in a traditional control system. The FCU runs under just about any POSIX-compliant operating system, such as Linux, QNX, UNIX, Windows, etc. They are also available in a variety of form factors, including high-performance PLCs and cost-efficient ruggedized industrial controllers. The FCU controller can directly scan and control multiple brands of I/O.

User-configurable devices and the reusable templates on which they are based replace rung ladder logic programming with a state-of-the-art, object-oriented technology that offers significant advantages.

For example, device objects are dropped into logical and/or physical relationships using graphical toolbars and menus — standard point-and-click Windows techniques. Minimal configuration is usually required to make each object unique and to place it into the control scheme.

![Figure 2 Drag-and-Drop Object Placement](image)

Although UCOS includes a wide range of standard control objects, it also supports extensive customization features via templating. The templates allow you to create control objects to suit any need.

Control schemes can be modified to fine tune or expand the system. Process I/O and operator workstations are easily added or deleted to meet changing control and business requirements.

Both the standard device objects and the user-definable device objects provide you with a library of starter control schemes that can be reused any number of times. It is easy to add project-specific templates to the UCOS library and to import those templates into projects, thus cutting development time.

More important, UCOS allows an organization to “package” its best practice process expertise and then deploy that expertise repeatedly resulting in control systems that are more consistent, more reliable, and easier to maintain throughout a project’s life cycle.

UCOS also automatically generates faceplates or command windows. You can, however customize command windows for user-defined devices.
UCOS Functional Overview

UCOS also provides automatically generated alarms, dynamic device graphic symbols, and other functions that are typically programmed or configured one-by-one in traditional control systems.

The automatically generated, dynamic device graphic symbols can be placed into operator display screens using a CAD-like graphics editor that also provides tools for adding other graphics elements, dynamics, and text.

As a result, the UCOS object-oriented, hybrid approach to control system development provides:

- More flexibility and control than DCS, PLC/HMI, or SCADA systems
- Significant reductions in the engineering time required to develop a project
- Enforced consistency within and among projects
- Ease of expansion and maintenance

Open System Features

The configurable features of UCOS afford many cost-saving advantages. The open system features of UCOS afford even more advantages.

UCOS is designed around open systems standards — from top to bottom.

- Engineering and operator workstations can be widely-available PCs.
- Real-time field control units (FCU) directly scan and control industry-standard I/O. They run under just about any POSIX-compliant operating system, such as Linux, QNX, UNIX, Windows, etc. and are available in a variety of form factors, including high-performance PLCs and cost-efficient ruggedized industrial controllers.
- Field data servers (FDS) can be ruggedized industrial processors available in a variety of form factors. These universal gateways interface data from intelligent devices, such as PLCs, Fieldbus technologies, and RTUs into the UCOS control system.
- All hardware is available from multiple vendors. In fact, multiple-vendor hardware can be mixed in a project — including multiple brands of I/O. Yet the configuration interface for all I/O is consistent across all brands.

With UCOS, hardware requirements are dictated by the requirements of your project, not by the control software. This balances flexibility and cost effectiveness without compromising either. As a result, both initial control system procurement costs and long-term maintenance costs are significantly reduced.
The UCOS open systems architecture also features:

- Microsoft Windows 2000/XP providing a reliable, secure platform for the engineering and operator workstations along with thousands of collateral applications, such as spreadsheets and word processors

- Open data exchange via OPC, ActiveX application programming interface (API), Open Database Connectivity (ODBC), and Microsoft Dynamic Data Exchange (DDE)

- Networking via satellite (VSAT), local area network (LAN), wide area network (WAN), radio, or microwave using TCP/IP Ethernet, Modbus, OPC, DDE, or proprietary protocols

- Intelligent use of report-by-exception communication schemes

- Hardware vendor independence at all levels

- Industry-standard, real-time operating system options for the field control units (FCUs)

- The ability to import project configuration data into the UCOS object-oriented database using Open Database Connectivity-compliant (ODBC) applications

- The ability to export historical data for use with ODBC-compliant applications

- Support for host-level control sequences in a true logic controller rather than in a script language

- No requirements to change logic programming when controllers are upgraded

- The ability to handle multiple I/O chains simultaneously

- The ability to modify control logic without restarting or rebooting the controller

- Highly flexible security elements that protect key system services and enforce corporate standards

UCOS provides an operator interface that is easy to use. High-resolution graphics and the familiar Windows GUI simplify interactive monitoring and control. Supervisory control of devices is consistent, regardless of device manufacturer. In addition, security is configurable.

Since UCOS standard components offer a wide choice of upgradeable features, the system provides scaleable price/performance to meet budgetary and phased implementation requirements.
System Components

UCOS supports several distinct, yet tightly coupled components:

- **Engineering Workstation** (EWS) for project development
- **Operator Workstation** (OWS) for operator interface
- **Field Control Unit** (FCU) for control logic execution and direct scanning of I/O
- **microFCU** for low-cost, low-power, or low-point-count applications
- **I/O Subsystem** supporting I/O from all industry standard suppliers
- **SCADA Data Server** (SDS) for interfacing data from intelligent devices, such as PLCs, Fieldbus technologies, RTUs, PLC I/O, and other third-party devices
- **Process Historical Archiver** (PHA) for storing and retrieving historical data collected by the FCU, SDS or any other intelligent device in the system

These components are connected via VSAT, LAN, WAN, radio, or microwave redundant or non-redundant networks using TCP/IP Ethernet, Modbus, OPC, DDE, or proprietary protocols.

**NOTE** Unless otherwise noted, in this document the term “UCOS” is intended to mean one or more of the above control system components.

This section describes the basic functionality and specifications of each of these components.
Engineering Workstation (EWS)

The EWS is the development tool where control schemes are configured then downloaded to the OWS, FCU, and SDS. The entire project is configured using a single integrated tool based on graphical Windows standards.

The highest security level allows definition of all of the system nodes along with the network structure. Graphical techniques are also used to define the logical relationships among the devices in a process area.

Project configuration begins by defining the system architecture: workstations, field control units (FCUs), I/O, networking, etc. You simply select a component for insertion into a graphical representation of the system architecture.

Graphical techniques are also used to define the logical relationships among the control elements for multiple devices. You can drag-and-drop graphical representations of device objects into a device diagram.

The device diagram acts as the project’s control logic, which is subsequently downloaded to the FCU. The device diagram includes standard devices, such as PID controllers and transmitters, as well as user-definable devices such as pumps, valves, and conveyors.

Each custom-configurable device includes many components: tag identifiers, defined logic schemes, and a graphical symbol inserted from a template. The standard logic of these configurable, “multi-state” devices can be modified and new devices can be created.

Also, graphical techniques are employed to modify or define the logic that determines device operation.

Figure 6 Drag-and-Drop Object Placement

Figure 7 Sample User-Definable Device Logic

As devices are defined and placed in relationships on the device diagram, the system automatically generates dynamic graphical symbols, faceplate command windows, and alarm displays. The supplied graphics editor can then be used to arrange the graphical symbols, configure additional static and dynamic elements, or modify the standard symbols, thus creating the dynamic displays for the project.

The project development tools also support security, grouping of command windows, logging, grouping of alarms, generation of project documentation, and a scripting language that allows you to customize processes and visual effects.

Entire projects or configuration changes can be downloaded to the operator workstations and FCUs. Changes can be tested on the EWS before downloading to the OWS.

The EWS has the following features:

- Project configuration including network definition and I/O selection
- Object-oriented control logic programming
- Windows 2000/XP operating system for secure, reliable interoperability
- Tower or desktop PC
- High resolution color monitor
- System communication via single, dual, or wireless TCP/IP Ethernet connection
Operator Workstation (OWS)

The OWS is used to monitor and control the process. It uses the project screens created during project development and animates them based on real-time data received from field control units and field data servers.

Authorized operators can monitor detailed activities for many types of devices and send commands using standard faceplate command windows and group displays. These displays are populated with the real-time control status of device tags received from field control units and SCADA data servers. Group displays may include the faceplate command windows for multi-state devices, transmitters, and controllers.

The OWS also allows operators to display and acknowledge current alarms or display historical alarms. Logs and trends can be accessed by menu selection. Authorized operators can change security status or monitor device logic as it is running in the FCU. They can also monitor the hardware through FCU, device, and rack tags that pass along information about the health of these devices and the connections between them. Device diagnostics show the current status of each device tag. Device tags can be toggled on-scan and off-scan, and current values can be overridden.

The operator interface features high-resolution colorgraphics and familiar Windows GUI interaction. The Windows environment supports the display of multiple project screens and windows. Data can be shared with other standard Windows applications via the SCADA data server.

The OWS assumes the functionality provided by supervisory control, HMI, or SCADA master/sub-master systems in traditional architectures.

The OWS can be displayed on a networked workstation or in an Internet browser located anywhere in the world.

Field Control Unit (FCU)

The FCU executes the control scheme configured on the EWS and directly scans industry-standard I/O. The FCU real-time controller runs under just about any POSIX-compliant operating system.

The FCU provides I/O services by monitoring and controlling I/O across standard networks and data highways. The FCU can provide simultaneous support for multiple vendors’ I/O and I/O networks. The variety of platform and form-factor options supported by the FCU allows incorporation of distributed, distinct I/O subsystems into common control strategies.

Logic processing is performed by the FCU according to the schemes developed during project configuration on the EWS.

The logic for a particular device is solved within one FCU or a redundant pair of FCUs. When a device is inserted into a device diagram during project configuration, it is associated with one FCU. In effect, each device is “owned” by a particular FCU. That FCU solves the logic for that device. The FCU also sends updates of data to operator workstations using exception-based reporting or shares the device’s data with other FCUs, if required by the control scheme.

Exception-based reporting allows data to be widely distributed without placing excessive demands on the network’s bandwidth. Previously manned sites can be converted to fully automated sites because exception-based reporting makes it technically and financially feasible to do so.
System Components

Alarms are also part of the device definition and are solved in the FCU and reported to the operator workstation.

The FCU features direct Ethernet network connections and standard interfaces to many specialized intelligent subsystems.

I/O Subsystem

The unique flexibility of this technology supports standard, off-the-shelf I/O subsystems offered by a variety of manufacturers.

The same logic can be solved to manipulate different I/O subsystems from different manufacturers without having to change any of the programming or operational parameters of the configured system. This allows a UCOS system to replace another control system without requiring replacement of the I/O. In addition, when the I/O is replaced, no change in logic is required since UCOS logic is hardware independent.

UCOS currently supports more than 40 families of I/O from more than 20 manufacturers.

SCADA Data Server (SDS)

The SDS is a controller and data concentrator in one unit.

It runs processes logic and directly scans supported I/O, as does the FCU.

For devices that cannot be directly scanned by the controller, the SDS offers a generalized, flexible, and user-configurable application designed to interface UCOS to other control systems, proprietary protocol drivers, or specialized software. Software protocol drivers may be supplied by CSI or by other third-party suppliers. The protocols run as separate tasks on the SDS and communicate to devices external to the control system, such as PLCs, RTUs, flow computers, intelligent valve controllers, Fieldbus, etc.

The data exchanged with the external devices or other software systems is then “served” to the control system database so the data can be used by logic, graphics, and other applications executing in the control system environment. Also, commands and setpoints initiated from the OWS are routed to the SDS application that, in turn, routes them to the appropriate external application.

The SDS has the following features:

- Supports both serial and TCP/IP connections
- Supports off-the-shelf, industry-standard ActiveX, OPC, and DDE drivers
- Communicates with any RS-232 compliant equipment
- Supports enabling/disabling of COM ports and polling lists
- Supports multiple ports
- Supports multi-drop slaves configuration on the RS-485 communication bus line
- Allows creation of a poll list for each port
- Supports unsolicited write

UCOS data servers are available in configurations that support low- or high-point count applications.

Process Historical Archiver (PHA)

Historical archiving involves copying the values of tags and other pertinent associated data, while a project is running, to an ODBC-compliant database. The values in the database can then be manipulated, examined, queried, used for rule-based decision making, and many other data analysis tasks.

The system supports run-time archiving of project data to a disk file (database) that is compatible with the Microsoft Open Database Connectivity (ODBC) standard. UCOS supports any standard ODBC-compliant database, such as Microsoft SQL Server, Oracle, etc.

ODBC-compliant database managers, report writers, expert systems, custom software, and other applications are then used to read and manipulate the archived data. The Trend Viewer module available in the UCOS control system is an example of an application that reads such a database.

The PHA determines which data to save to the database using exception-based archiving. If the data does not change, or does not change more than the user-configured deadband, then there is no archive entry to the database.

As a result, exception-based archiving minimizes the amount of data that is stored and solves the common problem of storing large amounts of static data. Pure exception-based archiving can be overridden for all tags or selected tags by configuring a minimum-write-rate variable to be a non-zero value. Tags configured with a minimum write rate other than zero will have their tag value archived at regular intervals, whether or not the tag value has changed.
Summary

UCOS is a single product that delivers total control system development, HMI, controller software, I/O scanning, archiving, and remote data gathering.

UCOS includes all the tools needed to develop and execute a control project. No other software products, tools, or utilities are required.

Yet UCOS is fully scalable, easily expanded, and benefits from the ability to work with a broad range of non-proprietary hardware.

UCOS makes the best of familiar technologies and expands on them. The next page compares and contrasts UCOS with traditional systems to help clarify some of the unique features and benefits offered by UCOS.

Figure 9
System Components in Example
Distributed Architecture
<table>
<thead>
<tr>
<th>UCOS</th>
<th>Traditional Control Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows regulatory, discrete, and sequential control to be created with a single tool through graphical and object-oriented software techniques. It produces the functional equivalent of function block and rung ladder logic programming.</td>
<td>Regulatory, discrete, and sequential control are often configured using multiple, separate tools.</td>
</tr>
<tr>
<td>Includes standard device objects and the ability to create user-definable device objects, all of which are reusable, yet unique.</td>
<td>Most systems do not have inherent support for templating or starter libraries of control logic. Any reuse of logic typically requires significant manual customization.</td>
</tr>
<tr>
<td>Encompasses all project components — from control logic to graphical symbols — in a single database.</td>
<td>Multiple development tools require multiple files to store various types of configuration information, such as tags, graphics, and logic.</td>
</tr>
<tr>
<td>Performs all addressing via structured tag names — no direct memory addresses are used.</td>
<td>PLC-based systems require direct memory addresses and complicated mnemonic naming conventions to maintain connections among rung ladder logic programs and the HMI software. Effectively, you must manage and coordinate two databases of configuration information.</td>
</tr>
<tr>
<td>Non-proprietary, off-the-shelf components at all levels – from workstations, to controllers, to I/O – are available from multiple vendors. Simultaneous support for multi-vendor I/O and competitive sourcing make for cost-effective implementation and expansion.</td>
<td>Proprietary and/or single-source components are often required. Non-competitive sourcing makes for higher costs at all levels.</td>
</tr>
<tr>
<td>OWS nodes, FCUs, I/O, devices, data servers, and instrumentation can be added or deleted to meet changing control system and business requirements. Multiple facilities can be connected to provide area-wide control.</td>
<td>Typically, the controller is a closed unit with no place to add cards. Sometimes minimal expansion is possible by adding other cards in the rack holding the controller.</td>
</tr>
<tr>
<td>Standard memory chips in the EWS, OWS, FCU, and data server provide low-cost expandability.</td>
<td>A new controller may need to be purchased to gain additional memory to hold new control programs or functionality.</td>
</tr>
<tr>
<td>The EWS and OWS run industry-standard Windows 2000/XP, which offers security along with connectivity to thousands of other software applications.</td>
<td>Engineering and operator workstations in DCS systems often run some form of UNIX.</td>
</tr>
<tr>
<td>The FCU runs under just about any POSIX-compliant operating system, such as Linux, QNX, UNIX, Windows, etc. which gives it the ability to run other software, and the ability to exchange data with other software using OPC, DDE, ODBC, and other data sharing protocols.</td>
<td>Controllers use some form of proprietary operating system (often no more than a low-level executive or task manager) that is typically embedded and not accessible. It has one purpose: to execute the proprietary control logic of that vendor’s system. Connectivity to third-party or custom applications is limited or nonexistent.</td>
</tr>
<tr>
<td>Communications based on non-proprietary TCP/IP that can operate over affordable, off-the-shelf communications equipment.</td>
<td>Frequently based on proprietary protocols that require expensive proprietary hardware.</td>
</tr>
<tr>
<td>Automatically incorporates report-by-exception communications that maximizes network throughput.</td>
<td>No pre-configured communications. Engineers must carefully program PLCs and the HMI to manage network traffic.</td>
</tr>
</tbody>
</table>
Project Configuration

Using an integrated, object-oriented configuration tool, a control engineer can implement complex regulatory, discrete, and sequential control without extensive programming.

Instead, graphical objects incorporating control capabilities and configured characteristics are dropped into place. This technique is used to configure:

- Hardware architecture including operator workstations, field control units, individual I/O points, data servers, etc.
- Logical control relationships and physical interconnections among devices within a facility or process
- Logic behind user-definable devices, such as pumps, valves, and motors

All configuration begins with the Project Configuration editor.

The Configure menu allows you to:

- Configure device logic, tag definitions, run-time graphical behavior, etc. for devices and store this device definition in a template. The template can be used to insert multiple new device definitions into control logic, which is called a device diagram. Each new device inherits all the characteristics of the template, yet each new template-based device is unique and can be further modified.
- Create control logic that provides all control instructions to the project’s field control units (FCUs). The control logic is organized by device. Each device contains all the control logic, tag definitions, and HMI graphics associated with that device. You can then arrange symbols representing each device in a device diagram to represent the control logic of a project.
- Create project graphic screens using a CAD-like drawing editor. The resulting screens can contain dynamic objects that change color, position, shape, etc. in response to run-time data.
- Organize command windows into group displays and then display these groups of command windows by name at run-time.
- Run-time command windows, or faceplates, provide a method of monitoring and controlling your process. UCOS creates command windows automatically for transmitters, controllers, and discrete devices.
- Configure destinations for alarm and logging messages.
- Configure Detectable Groups that can be used to enable/disable screen object clicks.
- Define communication parameters for each workstation that will be monitoring the Archiver.
- Specify or modify the project’s name, description, and network type.
- Perform global TCP/IP address changes.
- Configure F keys to show or hide screens in run-time.
Project Configuration

- Assign scripts to specific workstations. Scripts are programs written in a language specific to UCOS. Scripts can cause defined actions to take place through pre-programmed instructions in the script itself or operator action.

- Customize user-defined device command windows. Command windows can include data from multiple devices. This data can be displayed in digital or analog format or as trends.

- Use Dynamic Data Exchange (DDE) to pass information between the project and any DDE-compliant software package.

- Specify an FCU’s router address that can be used to detect whether an FCU or its router is down.

The Security menu allows you to:

- Configure project-specific security.
- Login/logout users.
- Set login default preferences.

Other menus allow you to:

- Customize symbol graphics.
- Download projects to workstations, field control units, and field data servers after configuration.
- Download updated device definitions and logic to FCUs without taking the project offline.
- Import and export hardware definitions.
- Print the project configuration, including the architecture drawing, the complete hardware configuration, control logic, and device configuration.

This section describes in detail the major functionality of Project Configuration.

Architecture Configuration

This level of configuration allows you to name the project, choose the type of network for the project, define the project’s hardware components, and set up the communication paths.

The project name and network type are specified when a new project is created via the Project > New menu item.

Workstations, printers, field control units, and data servers are inserted by clicking on the appropriate toolbar button or menu item. A dialog box appears in which you can enter configuration information. When you exit the dialog box, the new object is automatically inserted into a diagram of the architecture.

An FCU object or an OWS/data server object can be moved horizontally and can be modified by double-clicking on the object symbol or by selecting the object and choosing the appropriate Edit menu item.
Workstations, Archivers, Data Servers, and Printers

Workstations, Process Historical Archivers, and Data Servers require a node name, node address, and profile name. The workstation description is optional. The workstation profile associates a reusable set of security definitions with the workstation.

![Figure 13 Workstation Dialog Box](image)

From the **Workstation** dialog box you can:

- Configure communications parameters between this workstation and FCUs or PHAs, as well as optimize communications over wide-area networks (WAN).
- View and modify the DDE sources, logging devices, and alarm groups that are assigned to the workstation.
- Designate the run-time applications that will start automatically when Run-Time is initiated.

Printers can be associated with a selected workstation and require only a name and port designation.

Field Control Units and Associated I/O

FCUs and associated I/O are configured in one dialog box.

![Figure 14 FCU and I/O Hardware Definition](image)

This dialog box is designed to allow you to view as much or as little information as necessary and to add, change, and delete FCU and I/O definitions.

The left side of the **Hardware Definition** dialog box displays a telescoping list of all FCUs and I/O associated with the project. The relationships among FCUs, I/O cards, channels, racks, and modules are illustrated via a tree hierarchy.

The hierarchy can be viewed by FCU or by user-defined location within the facility.

![Figure 15 FCU and I/O List Organized by FCU View](image)
Project Configuration

For example, in Figure 15, three A-B 1771 remote I/O racks are connected on a single length of remote I/O communication cabling (“blue hose”). This is designated in the figure as “Chain No. 1.” The A-B I/O cards allocated to each rack can be assigned by selecting familiar part numbers or by selecting the type and voltage first, which filters the available part numbers. Throughout the setup process, UCOS provides filters to minimize errors in the selection of hardware settings.

UCOS allows you to create location names and specify locations for each FCU and each rack. This provides an alternate way of viewing and documenting FCU and I/O definitions.

The hierarchy tree can be expanded and contracted to show as little or as much of the hierarchy as you choose. For example, double-clicking on a rack in the tree displays a list of modules associated with that rack. Double-clicking on the rack again removes the list of modules from view.

The right side of the dialog box displays a detailed definition of the item selected in the hierarchy view. For example, if a rack is selected on the left side of the dialog box, the right side displays the definition of that rack.
The module definition shows, among other things, a list of points available for later assignment to tags. In Figure 20 Module Definition, the first three points have already been assigned.

I/O points are addressed via structured tag names throughout UCOS: in logic configuration on the EWS, for operator use on the OWS, and in the FCU for logic execution. This replaces the traditional method of direct memory addresses found in PLC rung ladder logic.

Project Configuration allows you to import or export hardware definitions. A hardware definition includes the configuration information for a workstation or an FCU. Hardware definitions are exported in the *.csv file format, which means you can modify them in a third-party application such as a text editor or a spreadsheet. This can decrease development time by allowing you to change hardware definitions for several components at a time, rather than one component at a time. You can also use the hardware import and export features to create a new project that is based on the hardware definitions of an existing project.

Tag names are actually assigned to I/O points in the Device Definition Editor on a device-by-device basis. If the appropriate FCU or I/O hardware has not yet been defined when the point assignment must be made, you can click a button to display the Hardware Definition dialog box. FCU and I/O hardware can be configured, as needed, during device configuration.

Backup points can be assigned for any points that need backup support. A backup module is assigned to a primary module. Once the first assignment is made to a backup module, any points added to the associated primary module are automatically made to the backup module.

You can also assign I/O to multiple devices by using Export on the Tools menu in the Device Diagram Editor. UCOS will sort the devices to user specifications and export the devices to a file. You can then open the device files in a third-party application and assign I/O to the devices.

Another feature of the Import/Export Devices function is reporting. Once you export device data from UCOS, you can import that data into an application appropriate for your reporting needs.
Device Diagram Configuration

In UCOS, device diagrams represent the master control logic programs that provide all control instructions to the project’s field control units (FCUs).

The device diagram combines the capabilities of traditional DCS function blocks with the sequential logic capabilities of user-definable devices that act as a replacement for PLC rung ladder logic. As such, regulatory, discrete, and sequential control are created with a single tool.

UCOS uses a drag-and-drop graphical interface to insert and configure devices from a library of user-defined device templates. These templates contain, among other things, all the logic and functionality underlying the devices represented in the diagram.

Additionally, the ability to customize the diagram symbols associated with devices allows device diagrams to graphically represent both logical control relationships and physical interconnections.

The symbols that appear on device diagrams represent equipment and logical control schemes. They can be connected together in a way that indicates how they interact with one another — both logical control interactions and physical interconnections.

This makes it potentially possible to glance at a device diagram, determine what role a device plays in control interactions, and determine the physical interconnections for a process or facility.
UCOS Functional Overview

UCOS Devices
Rather than requiring you to reprogram and reconfigure every device for every project, UCOS allows you to leverage industry standards when possible and create custom solutions when necessary.

UCOS accomplishes this by:

- Supplying a wide range of pre-configured devices with built-in functionality
- Allowing you to modify certain device configurations and functionality
- Allowing you to create custom devices with custom functionality

Devices fall into the following general categories:

- **Standard control devices**, such as transmitters and controllers. The functionality of each control device is built in and does not have to be programmed.

- **Standard computing devices**, such as selecting functions, limiting functions, and a calculation function. The functionality of each computing device is built in and does not have to be programmed, except for the calculation function, which allows you to specify a formula.

- **Static devices** have no functional characteristics but may be needed to fully represent the physical characteristics of the plant or process. The symbol used to represent a static device in a device diagram can be user-defined.

- **User-defined devices** such as pumps, valves, and conveyors. The logic, tag definitions, and symbols associated with these devices are fully configurable. User-defined devices replace rung ladder logic.

The symbols used to represent static and user-defined devices are fully configurable. UCOS supplies a wide range of pre-drawn symbols as well as a symbol editor that can be used to modify existing symbols and create new ones.

---

**Figure 25 Symbol Editor**

The symbol editor allows you to create graphics in Project Configuration or to import graphics from another drawing package.

**Device and Tag Names**
Uniqueness is enforced among devices via the device name, which is composed of a device type, a hyphen, and a device ID:

```
PUMP–3002A
```

Uniqueness is enforced among tag names by using the device name as a prefix and appending a period followed by a tag name extension:

```
PUMP–3002A.Running
```

This enforced tag name convention allows UCOS to:

- Support the ISA standard approach to naming control loops by specifying the functionality in the device type, the loop number in the device ID, and the parameter in the tag name extension.

- Support identical tag name extensions in similar devices while preserving unique tag names via the device name prefix. For example, UCOS can support several similar pump templates, each of which can be configured with slightly different logic but each of which uses the same tag name extensions to identify similar inputs and outputs.

- Support almost any other tag name convention you need to use.
Device Connections
Connection lines indicate the physical and/or logical relationships among devices (see Figure 24):

- Mechanical lines (thick solid) are not associated with any control functionality. They do, however, serve to indicate the physical interconnection among devices.
- Static lines (medium gray) indicate signal flow from the process to a device.
- Tag data flow lines (dashed) indicate flow of data among devices.

Although many devices have multiple inputs, UCOS makes it easy to ensure that only valid connections are made. For example, connection types are readily visible when Device Connection Mode is enabled and the cursor passes over a connection point.

![Figure 26 Static Connection on a Device](image)

Emulating Traditional Drawings
Using these basic components, a developer can use device diagrams to represent a facility or process any number of ways.

In addition, UCOS allows only valid connections. For example, UCOS will not allow the output of one device to be connected to the output of another device.

Control Devices
The underlying functional characteristics of control devices are based on widely accepted industry standards and, therefore, do not need to be defined.

For example, PID loop controllers perform an industry-standard function that applies to control of many types of facilities or processes.

Rather than requiring you to program each controller’s functionality, UCOS uses graphical techniques and fill-in-the-blank forms to configure each controller device.

You enter a unique name for the new controller device, the FCU where the logic for the controller device is solved, the point to which the controller device output is assigned, and a backup point.

You can also modify default tag names and definitions for alarms, setpoints, commands, and other defaulted parameters.

![Figure 27 Controller Device Definitions Dialog Box](image)
Once the controller device is configured, you simply click in the device diagram to indicate where the new device goes.

**Figure 28 Placing Controller in Device Diagram**

This completes the configuration of one controller. Other control devices are configured using the same procedure.

It is not necessary to program the PID algorithm that defines the functionality of a controller device. The FCU software is pre-programmed with the PID algorithm and executes it when called to do so. The same is true of other UCOS control devices.

UCOS supports the following control devices (shown here with the device diagram symbols):

<table>
<thead>
<tr>
<th>Control Device</th>
<th>Diagram Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
<tr>
<td>(Analog Input)</td>
<td></td>
</tr>
<tr>
<td>Controller</td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>(Analog Output)</td>
<td></td>
</tr>
</tbody>
</table>

**Computing Devices**

Most of the attributes of control devices also apply to computing devices:

- Functionality is pre-programmed for all but the f(x) function, which allows you to configure the formula.

- The procedure for inserting and configuring computing devices is similar to that for control devices.

- As with control devices, computing devices have several required and optional parameters that allow you to customize each device.

UCOS supports the following computing devices (shown here with the device diagram symbols):

<table>
<thead>
<tr>
<th>Computing Device</th>
<th>Diagram Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-Defined Function</td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>High Signal Select</td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td>Low Signal Select</td>
<td><img src="image5" alt="Diagram" /></td>
</tr>
<tr>
<td>High Signal Limit</td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
<tr>
<td>Low Signal Limit</td>
<td><img src="image7" alt="Diagram" /></td>
</tr>
</tbody>
</table>

The user-defined function f(x) allows you to specify a formula to be solved in the FCU. The formula supports up to three input variables and five constants, as well as standard operators and functions.

The input variables come from tags that can be specified in either of two ways. You can enter the name of any tag for any of the three input variables. Or, connect the output of a device symbol in the device diagram to one of the f(x) inputs. UCOS automatically inserts the connected tag name into the dialog box.
Project Configuration

After connection is made:

Before connection is made:

Figure 29 UCOS Automatically Inserts Tag Name for Input Variables (on right)

Static Devices

Static devices have no functional characteristics in the control scheme but are included to fully represent the physical characteristics of the plant or process.

Each static device requires a name and a symbol to represent it on the device diagram.

User-Defined Devices: Overview

UCOS supplies functionality and configurations for a wide range of user-defined devices such as pumps, valves, and conveyors. The functionality and configuration for a particular type of user-defined device are defined in a template and can include:

- The device logic executed by the FCU at run-time
- The names and characteristics of tags associated with the device
- The symbol used to represent the device in a device diagram (created/modified by the Symbol Editor)
- The dynamic, graphical behavior of a symbol used to represent the device in a project graphics screen at run-time (created/modified by the Template Graphic Editor)

Figure 30 Insert Static Device Dialog Box

You can select one of the supplied symbols or use the Symbol Editor from the Project Configuration window to create a custom symbol.

Figure 31 Members of User-Definable Device Template PDPUMP-TEMP

Inserting a user-defined device into a device diagram involves little more than specifying the template on which the new device will be based, as in the following procedure:

1. Select the UD button from the toolbar or the User Discrete Device menu item.

Figure 32 Inserting a User-Defined Device
2. In the dialog box, select the template on which the new device will be based. Enter a unique name for the new device (device type and ID). Select the FCU where the device logic is to be solved. The diagram symbol and execution order are defaulted and may be changed.

![Figure 33 Insert Discrete Device Dialog Box](image)

3. Click in the device diagram to indicate where the new device goes.

![Figure 34 Placing User-Defined Device in a Device Diagram](image)

This brief procedure copies all the functionality and configuration of the template to a new device as represented in Figure 35.

![Figure 35 Components of User-Defined Device PMP-1001, One Instance of Template PDPUMP-TEMP](image)

All the logic, tag definitions, and symbols of the template are inherited by the new device. They can be modified for the new device without affecting the functionality and configuration of other devices based on the same template.

For example, a device named PMP-3003 and a device named PMP-1237 can be inserted into a device diagram in addition to the device named PMP-1001. Each of these devices can be based on template PDPUMP-TEMP. The only major differences among them are the device names and tag name prefixes, which are automatically changed to reflect the unique device name.

![Figure 36 Device Naming Convention](image)

At this point the logic, tag definitions, and symbols of each device can be modified independently of one another. Each change affects only the inserted device for which the change is made. It does not affect other devices based on the same template, nor does it affect subsequent devices inserted from the same template.

The Device Definition Editor is used to modify the characteristics of inserted user-defined devices.

The Template Definition Editor is used to modify and create templates. The templates supplied with UCOS can be used as is or copied and modified to create similar templates. In addition, new templates can be created with custom logic, tag definitions, and symbols.

Creating templates is not required in UCOS. The templates supplied with UCOS may provide all the functionality required in a given project. Multiple new devices can be inserted into a device diagram from each of these supplied templates.

However, new templates may need to be created if, for example, a project will use several unique pumps with different permissives. A new template can be created for each type of pump prior to device diagramming.
You can also export and import templates to and from another project. This feature also allows you to use a spreadsheet or database manager to further document your project.

The **Import Templates** dialog box allows you to select a source project from which to import a template or group of templates. Once you have selected the source, simply select the templates from the source list, add them to the selected list, and then import them into the target project.

![Figure 37. Import Templates Dialog Box](image)

**The Template and Device Definition Editors**

The Template Definition Editor and the Device Definition Editor share nearly identical functionality and are described in the next sections as if they were one editor. Differences are noted.

**User-Defined Devices: Logic Categories**

UCOS provides a structure for user-defined device logic. To the extent that this structure exposes the base elements required for control of a device, it is unique to UCOS.

This logic structure is based on the premise that control of most equipment can be organized into five logic categories. These categories describe various operational aspects of any piece of equipment. In rung ladder logic programs, logic for each of these categories is often written but not necessarily identified as such. UCOS simply exposes these categories.

- Mode
- Fault
- State
- Control
- Alarm

Much of the value derived from logic categorization is the near-universal applicability of these categories to any piece of equipment that needs discrete outputs for actuation and control. By exposing these categories, UCOS encourages the control engineer to configure consistent logic, which results in a system with greater integrity and ease of maintenance.

Although there is no difference in the **type** of logic created for each category, there is a difference in the intended **use** of logic for each category. This difference is more than a convention. The logic solving software in the FCU provides slightly different behavior for the logic in each category with respect to the output tags associated with the logic.

Each logic category is detailed in the table on the following page, including the general interactions of the various logic categories.
### Logic Category | Intended Use | FCU Execution Support
--- | --- | ---
**Mode** | Mode logic defines how the device will transition into different modes. These modes of operation (e.g. Local/Remote, Auto/Manual) are examined during the execution of state logic. | The FCU enforces mutual exclusivity per mode group. The user designates each tag in the Mode folder as part of a mode group. The FCU allows only one tag per mode group to be set at any given instance. |
**Fault** | Fault logic describes severe abnormal conditions associated with the device. These fault conditions (e.g. very high bearing temperature) are examined during execution of state logic and are reported as severe alarms to the user. | The first tag listed in the Fault folder is set by the FCU as the logical OR of all the remaining fault tags. This first fault tag will normally be named ANYFAULT and will not be driven by user-defined logic. |
**State** | State logic relates to the observable, controllable states of the device. State logic examines the conditions of commands, modes, faults, setpoints, etc. to determine the current state of the device. The current device state is examined during execution of control logic. | The FCU enforces mutual exclusivity for all state tags. The FCU allows only one state tag to be set at any given instance. |
**Control** | Control logic examines conditions of states to determine the value of internal and real-world outputs. These control outputs operate the device. | Standard logic solving |
**Alarm** | Alarm logic describes less severe abnormal conditions associated with the device. These alarm conditions (e.g. bearing approaching high temperature) are not necessarily examined by the other categories of logic but are reported as alarms to the user. | Standard logic solving |

### User-Defined Devices: Logic Basics

The model for user-defined devices in UCOS requires that each device be in only one state at a time, such as Starting or Running, but not both simultaneously. Implementing this requirement in traditional control systems can require many rungs of ladder logic, the latching and unlatching of bits in memory, testing and retesting, and other tedious programming techniques to ensure that the device never achieves multiple states simultaneously.

UCOS requires no programming and no testing to enforce mutually exclusive states. Instead, the logic for enforcing mutually exclusive states is pre-programmed into the FCU logic-solving software.

You simply place one tag for each state onto the State tab. That is all that is required to ensure mutual exclusivity among state tags.

---

**Figure 38 State Tab in Tag Definition Dialog Box**

UCOS also enforces mutual exclusivity among mode groups. You associate each mode tag with a mode group, and the FCU enforces mutual exclusivity among mode groups.
For example, Mode Group 1 may have an Auto and Manual mode and Mode Group 2 may have a Local and Remote mode. In this example, the FCU will ensure that Auto and Manual are mutually exclusive with regard to each other and that Local and Remote are mutually exclusive with regard to each other.

**User-Defined Devices: Logic Configuration**

The functionality of a user-defined device is defined in the Template and Device Definition Editors. In traditional control systems, sequential or discrete control functionality is defined through rung ladder logic programming. UCOS does not use rung ladder logic programming.

Instead, the Template and Device Definition Editors employ a graphical approach to creating object-oriented discrete device logic based on an adaptation of state logic concepts. Although many of these concepts are new, the logic used to define the functionality of a user-defined device is familiar.

**Figure 41 User-Defined Device Fault Logic** illustrates a typical page of logic for one category of logic, in this case fault logic.

From the left, input tags feed data into elements such as logic gates, timers, and counters in the center. From these elements, the data is fed into output tags on the right.
UCOS supplies a default execution order that can be modified for each logic element. This helps ensure that logic is executed as you intend.

Figure 42 Displaying Execution Order

Logic for each category can be placed on a single, scrollable page or on multiple pages to aid readability. Each category of logic supports up to 16 pages with the Control category supporting up to 32 pages.

Figure 43 Categories of Logic

Logic elements and tags are inserted from a toolbar button or menu item and dropped into the desired location within the selected logic scheme.

Figure 44 Dropping an OR Gate Into a Logic Scheme

You can assign I/O to each device from the tabs in the Device Definition Editor. You can also import and export device definition data to and from the UCOS object-oriented database via comma-separated (*.csv) files. You can modify device and tag definitions in a third-party application, such as a spreadsheet or database. This can decrease development time by allowing you to change tag definitions and make I/O assignments for several devices at once, rather than one device at a time.

Import and export are not designed to create new devices. That is a task more quickly and easily accomplished with templates. Import and export do, however, simplify the process of making changes in a large group of existing devices.

Most device and tag elements, with the exception of device logic, can be modified through import and export.

User-Defined Devices: Logic Operators

UCOS supports a wide range of standard logic operators including logic gates, counters, timers, math operators, etc.

Figure 45 Gates, Counters, Timers, and Math

When inserted into a logic scheme, some of the symbols display small letters around the edges. These letters indicate the types of inputs and outputs that can be connected at those locations. These vary by type of logic element.

Figure 46 Letters Around Symbol Edge Indicate Type of Connection

Logic operators require little if any configuration beyond the fundamental parameters, such as presets or constant values.
Project Configuration

User-Defined Devices: Tags
I/O points are addressed via structured tag names throughout UCOS: in logic configuration on the EWS, for operator use on the OWS, and in the FCU for logic execution. This replaces the traditional method of direct memory addresses found in PLC rung ladder logic.

Each tag requires some definition that varies by tag type. When inserting tags into a logic scheme, the following tag types are available:

- **Internal Input/Output** (data originates/stays in UCOS)
- **Real-World Input** (data originates from field I/O for the device)
- **Real-World Output** (data sent to field I/O for the device)
- **Command Input** (operator initiated)
- **Setpoint Input**
- **Wildcard Input/Output** (placeholder tag to be resolved after inserting a device on a device diagram)
- **Other Device Input/Output Tag** (data originates from or is sent to another device; Device Definition Editor only)

Tags are defined in a dialog box that further categorizes them. (Numbers in parentheses indicate the maximum number of tags of that type that can be defined for a given device.)

- **State** tags (16)
- **Mode** tags (16)
- **Alarm** tags (16)
- **Fault** tags (16)
- **Display Point** tags (16) (Device Definition Editor only; these are tags from other devices displayed in the current device command)

window as a convenience to the operator; they are not used in logic.)

- **Setpoint** tags (24)
- **Command** tags (10) (Command tags appear as buttons on the run-time command window of the current device; to send a command to the tag, the operator clicks the button.)
- **Digital Control** tags (32)
- **Analog Control** tags (16)

All tags (except display points) can be used as inputs in any logic category any number of times. Only compatible tags can be inserted in each category of logic.

Digital control tags can be used as outputs in any logic category. But state, mode, alarm, and fault tags can be used as outputs only in the respective logic categories.

Figure 47 I/O Tags (“Other Device” Tags Appear in the Device Definition Editor Only)

Tags can be defined and modified at any time while in the Template/Device Definition Editors — including while inserting a tag into a logic scheme.

All tag conventions are strictly enforced. But you do not have to remember the conventions because UCOS limits your choices based on the context.

For example, in Figure 49, the alarm logic scheme is selected. When you click on the internal output toolbar button, the resulting dialog box offers only compatible tags for insertion.

Figure 48 Fault Logic and Tag Definitions

In Figure 48, note that the output tags in the fault logic scheme correspond to the tag definitions on the Faults tab in the dialog box. The tags defined on the Faults tab of the dialog box represent a list of possible fault output tags that can be used in the fault logic scheme. In fact, only certain output tags are allowed in a fault logic scheme, depending on whether the selection is an internal output or a real-world output tag.

Tags can be defined and modified at any time while in the Template/Device Definition Editors — including while inserting a tag into a logic scheme.

All tag conventions are strictly enforced. But you do not have to remember the conventions because UCOS limits your choices based on the context.

For example, in Figure 49, the alarm logic scheme is selected. When you click on the internal output toolbar button, the resulting dialog box offers only compatible tags for insertion.
Each tag definition is composed of several parameters. Each tag must be defined with a tag name extension that must be unique within the device. Other parameters specify the logging and alarm group to which the run-time activity of a tag is logged, the wording and color that identifies the tag in the device command window at run-time, security information for selected tags, etc. The specific parameters vary by tag type.

Wildcard tags are used in templates to represent tags that reside in devices that you have not yet added to your project. Wildcard tags allow templates to be used in many device diagrams and projects where many devices will be based on the same template. You can also use wildcard tags in devices.

At some point during device diagramming, you must replace wildcard tags with tags from actual devices. When you double click a wildcard tag within the Device Definition Editor, the **Wildcard Resolution** dialog box is displayed. This dialog box enables you to assign a tag and its associated device to the wildcard tag. The procedure of assigning a tag to a wildcard tag is known as resolving a wildcard tag.

The Report Unresolved Wildcards feature, which is part of the Device Diagram Editor, lists all the unresolved wildcard tags in a project. This saves you the trouble of searching through pages of device logic and locating the unresolved wildcard tags yourself.
This data is entered into the Script Text Area. When the script is complete, it must be compiled. If errors are encountered during the compile, they are listed in the Compiler Message Area along with the line numbers where they occurred. Double-clicking an error message will highlight the incorrect line in the script text area.

After a successful compile, all tags referenced in the script are listed in the Tag References area. You can then step through the script and watch how each tag value changes line by line. This can help troubleshoot the script.

Next, you will assign when and how the script is to run. Simply click the Add button below the Execution Conditions Area and select the conditions under which the script will run. These conditions can be opening or closing a screen, specific changes in data, a specified length of time, starting or closing the project, or a combination of these conditions.

The final step is to assign the script to a script group and one or more workstations.

Graphic Editors

The Screen Editor allows you to create the graphical user interface, or project screens, for the project. For example, you can create a control screen that displays certain pumps and valves, graphically represents their logical and/or physical relationships, and allows you to monitor their status and send commands to them via command windows or other means.

The Template Graphic Editor allows you to create and modify the graphic component of templates.

The Screen Editor and the Template Graphic Editor share nearly identical functionality and are described in the next sections as if they were one editor. Differences are noted but both editors are referred to as “the Editor” in this section.

Generally, the Editor is used to insert objects into the drawing area, manipulate the appearance and position of the objects, and apply run-time dynamics to the appropriate objects. Objects can be primitive graphics (lines, rectangles, circles, etc.), controls (buttons, check boxes, data-entry boxes, etc.), and bitmaps. In addition, previously created template graphics associated with specific devices can be inserted into the drawing area.

Template Graphics

Templates define certain default device characteristics, including logic, tag names, and graphics. When you insert a user-defined device into a device diagram, UCOS copies the default device characteristics from the appropriate template to the new device. The new device will have the same logic, tag name extensions, and graphics as the template and any other inserted device based on that same template. These characteristics can be changed for each inserted device.

Each UCOS template comes with a pre-configured component that specifies the run-time appearance and dynamics of that graphic component, i.e. how that device will look and act on a project graphic screen at run-time. You can modify this behavior and/or create new graphic components using the Template Graphic Editors.

As described in a previous section, device diagramming automatically generates two symbols for each device inserted into a diagram.

<table>
<thead>
<tr>
<th>A Device Diagram Symbol</th>
<th>A Project Graphics Screen Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="PD" /></td>
<td><img src="image2" alt="PD" /></td>
</tr>
</tbody>
</table>

The graphic component you create or modify is associated with the template.

As described in a previous section, device diagramming automatically generates two symbols for each device inserted into a diagram.
When you insert a device into a device diagram, a diagram symbol appropriate for that device appears in the device diagram. When you insert that device on a project graphics screen, a dynamic graphic symbol appropriate for the device appears on the project screen.

The remainder of this section describes in more detail various ways to draw objects and make them perform in a specific way at run-time.

**Project Screen Graphics**

Project screen graphics can be as elaborate or as simple as your needs require. The Editor allows you to produce a dynamic graphical representation of the process. You can configure the project screen to display active process and instrumentation values, manually control tag values, and dynamically illustrate the process.

![Figure 55 Project Screen Created in Screen Editor](image)

In addition to dynamic objects, project screens can include static objects such as text labels and screen navigation objects to allow the operator to move among multiple screens within a project. (Navigation is also possible using run-time menu items.)

**Graphic Objects**

You can combine device graphics, bitmaps, graphic primitives, and/or control objects to create objects on operator screens. Your operator screens can be simple or complex, depending upon your needs.

**Devices (Screen Editor only)**

You can select any user-defined device that appears in any of the current project’s device diagrams as long as the device has a template graphic associated with it. After you have selected the device, click where you want to put it in the drawing area.

**Bitmaps**

Bitmaps can be inserted into project screens or template graphics. Although they cannot be configured with dynamics, you can place a transparent object over a bitmap and configure dynamics for that object.

**Primitives**

Graphic primitives include such basic geometric shapes as lines, rectangles, and circles. These basic shapes can be combined to create more complex shapes.

**Controls**

Controls are graphics that look and act like standard Windows user interface objects. They include buttons, static text, text boxes, check boxes, radio buttons, and group boxes.

![Figure 56 Configurable Control Graphics](image)

**Manipulating Objects**

The Editor provides basic graphics manipulation tools for designing graphic objects and project screens. You can select, move, resize, reshape, and rotate objects. In addition, you can group multiple objects to make a single object to which you can assign dynamics. This object can be duplicated as needed, maintaining its dynamic properties if you so choose. You can also align objects, make them the same size, mirror, invert, or clone them individually or as a group, and change the Z Order (front-to-back order) of individual graphics or graphic components.

**Static Properties**

Static properties define the default appearance of an object at run-time. If the object is assigned no dynamic properties that change its appearance at run-time, then static properties define the way the object always looks at run-time. Otherwise, static properties define the initial appearance of the object before it is changed by dynamic properties. You can define the following static properties for an object:

- Interior fill (color, percent of fill, and fill direction)
- Text color, direction (vertical or horizontal), and font
- Edge color, width, and style
Project Configuration

Data-Initiated Dynamic Properties
Run-time data can dictate the appearance or behavior of objects at run-time. You can configure the following dynamic onscreen device characteristics to respond to run-time data:

- Interior fill color and percent of fill
- Visibility/invisibility (determined by the result of a single expression)
- Edge color and style
- Text color
- Text value
- Rotation
- Movement

You can trigger these changes with an expression in the dynamic properties of an object. An expression can be as simple as a single digital tag. When the tag is TRUE the expression is TRUE, and the dynamic property associated with that tag is implemented. Expressions can also be more complex and can include digital tags, analog tags, math operators, or any combination of the three.

In the following illustration, each alarm tag is associated with a different fill color, i.e. the interior color of the object. The Percent expression is associated with a transmitter scale value tag.

An expression can be a specific value or a formula. The formula will return a value that can vary, depending on the run-time values of the tags comprising the formula. An expression can consist of device or local tag names used as variables, operators (plus, minus, multiply, etc.), and parentheses used to force a specific order of precedence. The Expression Editor creates expressions with minimal typing for those fields that accept expressions.

Figure 57 Dynamic Fill Tab for Run-Time Graphics

At run-time, the object changes color depending on the prioritized list of expressions. For example, in the figure above, FIT-1007.HI_LIMIT has the highest priority. If both FIT-1007.HI_LIMIT and FIT-1007.LO_ALARM are HIGH, the object will assume the color and fill assigned to FIT-1007.HI_LIMIT because its priority is higher than the priority of FIT-1007.LO_ALARM in this expression.

Operator-Initiated Dynamic Properties
You can specify an action to perform when the user clicks an object at run-time. The following actions can be configured:

- Display a command window or group display
- Change the value of tags
- Show, hide, or close windows containing project screens
• Launch a UCOS run-time application or a third-party application

• Run a script

Static, data-initiated, and operator-initiated properties can be combined to allow behavior not specified above. For example, a mouse click can change the value of a tag that causes the fill color of an object to change. Most dynamics assume the characteristic of the highest priority expression that is TRUE in a list of expressions.

Changing the Color Palette
The Editor allows you to select colors from a palette of 126 colors. A default color palette is supplied with a wide range of colors that will suit most needs. If you have specialized needs, you can change the color palette.

Startup Screens (Screen Editor only)
You can specify which screens are loaded into memory when the project starts at run-time. This allows you to load only those screens that are necessary for routine operation. If other screens are required, they can be loaded when they are needed.

Zoom In / Zoom Out
You can change the cursor to a magnifying glass and zoom in or out each time the left mouse button is clicked in the drawing area. Zooming is centered on the area where the mouse is clicked. You can click multiple times to zoom in or out by multiple magnification levels.

Grid (Screen Editor only)
You specify the appearance and visibility of the grid, which can be used to visually align objects.

Alarm and Logging Configuration
Object-oriented techniques allow you to configure alarming and logging with minimal effort.

First, create one or more alarm groups and/or logging groups.

A logging group consists of a list of logging devices to be associated with that logging group. Each logging device sends data to a printer or disk file.
Finally, select the alarm group, alarm priority and/or the logging group to be associated with each tag, as applicable. The following table indicates which types of tags can be logged via which types of groups.

<table>
<thead>
<tr>
<th>Tag Type</th>
<th>Alarm Group</th>
<th>Logging Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>States</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Modes</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Alarms</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Faults</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Setpoints</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Commands</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Digital Controls</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Analog Controls</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>

As depicted in the previous figure, each tag can be associated with only one alarm group and/or one logging group (depending on the type of tag). However, each group can be associated with multiple tags and/or destinations. Each alarm group, alarm priority, logging device, and logging group will be defined only once in a project. However, once the item is defined, you can create alarm and logging associations for thousands of tags simply by selecting a name from a supplied list. UCOS handles the task of matching the proper workstation or logging device with each tag at run-time.

Security Configuration

UCOS provides a flexible set of security functions and easy-to-use dialog boxes for security definitions.

Certain functions are pre-defined as secure, i.e. they are protected from unauthorized use. For example, secure functions include developing a project, starting or stopping a run-time project, sending commands to a device at run-time, acknowledging alarms, and archiving data. Project and security configuration are also secure functions. These functions can be used only if the logged-in user has security clearance to use them.

All run-time project graphics screens are also considered secure. However, the level of security can vary from screen to screen and from workstation to workstation. A user may have access to a given screen on one workstation and no access to that same screen on a different workstation.

Security is configured using object-oriented techniques that provide significant flexibility while minimizing the amount of required configuration.
User Access
In order to start Project Configuration or Run-Time, a user must log in by providing the correct username and associated password.

Figure 65 Log In User Dialog Box

Security Groups
Security access is configured via security groups. Each user belongs to one security group. The security group determines who has access to the following tasks:

- Starting and stopping a project in run-time
- Archiving data and accessing that data
- Sending commands and setpoint changes
- Acknowledging alarms
- Toggling tags on and off
- Manually overriding tags in Device Diagnostics
- Developing a project
- Configuring security
- Accessing project screens in run-time

Figure 66 Security Groups Dialog Box

Workstation Profiles
Workstation profiles assign a set of security groups to each workstation. As a result, you can customize security for individual workstations.

Security Users
Each security user is assigned to a security group, username, and password.

Figure 67 Add User Dialog Box

When a user logs in to a workstation, he or she can use any secure function that was enabled for his or her associated security group.

Auto Log Off
Any EWS and OWS can be configured to automatically log off after a user-specified number of minutes of inactivity, that is no keyboard or mouse activity. This adds another level of security that is independent of the operator.

Command Windows and Group Displays
A project can be controlled at run-time by three types of displays:

- Command windows
- Group displays
- Project graphics screens

Command windows are one method of monitoring and controlling devices at run-time. They allow the operator to monitor the real-time values of all tags associated with a device and to send commands to a device (subject to security measures).

Group displays and project graphics screens provide two different ways to display command windows at run-time and also provide additional features. For information on project graphics screens, see “Project Screen Graphics” on page 31.

Command Windows are generated automatically by UCOS for each transmitter, controller, and user-defined device.
inserted into a device diagram. You can, however, customize command windows for user-defined devices.

Figure 68. Group Display Containing One Command Window Each for a Transmitter, a User-Defined Device, and a Controller

In Figure 69, the command window contains several elements added through customization. Customized command windows allow you to include tag data for the device with which the command window is associated, as well as tag data from other devices. Two types of tag elements are available for customization: tag sets and trends.

Figure 69 Command Window in Run-Time

- **Tag sets** are collections of tags and tag values displayed in a command window. Tag sets can show tags and tag values for analog or digital tags. If you select digital tags, you can choose to show the tags at all times or only when the tags are HIGH.

  Figure 70 Typical Tag Set

- **Trends** (plot or bar type) allow tags to be trended during run-time. Trends will show real-time data only. If you need to display archived data, use the Trend Viewer.

  Figure 71 A Graph Trend (left) and a Plot Trend (right)

In addition to the tag set and trend window, you can determine how to display commands and setpoints in run-time. You can choose to place the commands and setpoints on the command window as buttons, at either the right or left edge of the command window. You can also choose to display commands and setpoints in a menu or as both buttons and a menu.

Command Windows can be displayed in two ways. One way is to right-click an appropriate object on a project screen.

Figure 72 Selecting an Appropriate Device on a Project Screen Displays a Command Window

Another way is to call up a group display that contains the command window you want to see.

Figure 73 Group Display Command Window Selection

**Help System**

The UCOS help system provides reference help for Project Configuration and Run-Time.
Operator Interface

The operator interface features high-resolution graphics and familiar Windows GUI interaction.

The operator interface running on the OWS node:

- Allows operators to monitor detailed activity for many types of devices and send commands using command windows and group displays
- Displays optional project screens created during project development and makes them dynamic based on real-time data
- Logs alarms and other events, displays alarm and log messages, and lets operators acknowledge alarms
- Trends real-time and historical data
- Lets authorized operators monitor and manually override device logic
- Displays the status of communication between the OWS and FCU, which helps the operator diagnose the cause of communication interruptions

The OWS can also run on a field data server, allowing OWS functionality that does not include operator intervention, such as scripting, logging, etc. In addition, the OWS can be securely accessed using an Internet browser and Microsoft Terminal Services.

UCOS provides operators with high-resolution colorgraphics and the Windows graphical user interface.

Flexibility is ensured via selected operating characteristics that are configurable at run-time. Standard features include simultaneous display of multiple project graphics screens, drop-down menus, and run-time configuration dialog boxes.

Consistency is ensured because most monitoring and control functions are automatically generated. For example, command windows are generated automatically during project configuration and present a consistent appearance and functionality to the operator – even if the monitored devices are from different manufacturers.

UCOS provides built-in features for monitoring and controlling your project. You don’t have to do any programming or interface with any other software packages to get trend and alarm viewing, device and system point status, command windows and animated schema from which you can control your project. All this is automatically configured when you design the project. The following figures display several OWS monitoring and control displays.

Figure 74 Device Diagnostic – Device Points

Figure 75 Alarm Viewer
Figure 76 Animated Schema

Figure 77 Device Point Status

Figure 78 Group Display

Figure 79 Log Viewer

Figure 80 Trend Viewer

Figure 81 Network Diagnostics
Project Graphics Screens

Operators can display multiple project graphics screens at one time. Screens can be selected for display from a drop-down menu or accessed via embedded paging objects.

You can use F keys to show or hide screens at run-time. These keyboard shortcuts can be specific to each workstation and are set up during project configuration.

Command windows for monitoring and controlling detailed device parameters can be displayed at any time by clicking on appropriate devices.

Depending on how the screens were configured, they may display real-time data via color changes, movement, numbers, text, etc.

You can configure screen objects that are linked to the executable file of an application within UCOS or of a third-party application. UCOS starts the application when the operator clicks the appropriate screen object. Similarly, you can configure screen objects that open or close run-time screens.

The user’s level of access to each screen is controlled by the security definitions entered during configuration. The levels of screen access include no access (screen cannot be displayed); view-only access; or full access (operator can send commands via command windows, as defined by the user’s security profile).
Command Windows and Group Displays

Right-clicking a device on a project graphics screen displays a command window for that device. Up to 10 command windows can be viewed in each group display. The operator can access group displays by name via the Group Display dialog box or by clicking an appropriately configured screen object.

As you select each group display, you can see which command windows are included in the group display.

By default, command windows are displayed in partial view, as shown in Figure 84. In full view, additional tags associated with the device are accessible. Double-clicking on one of the setpoint tags (HiDisOrd) displays a dialog box (Set Value) which allows the operator to enter a value to be sent to that tag (See Figure 85).

The command window for a user-defined device can display tag sets, trends, and/or up to 10 command buttons that can be specified by the developer. Clicking one of the buttons momentarily sets the tag associated with that button. If that tag is used as an input in any logic scheme, then that logic is solved as a result of the tag being set. (You can require that the operator confirm selected commands prior to execution of the command. Also, commands are subject to security measures.)

Command windows for controllers display the current setpoint and process variable values of the device as side-by-side bar graphs. An output bar graph shows the current operating level of the device output on a user-defined scale. The scroll arrows to the left of the Output graph increase or decrease the output by user-defined amounts when in manual mode.

The Setpoint button allows an operator to enter new setpoints for the control loop. The Auto/Man button allows an operator to control the output value automatically (using a standard PID algorithm) or manually. In manual mode, the Output button allows the operator to enter a value for the output. The Dir/Rev button allows the operator to select direct or reverse acting mode.

A trend displays the process variable.
The transmitter command window displays the current value of the device as a number and as a bar graph. The raw value for the device is displayed as a percentage; a setpoint tag set displays all setpoint values.

![Transmitter Command Window – Full View](image)

The ability to send commands and change values is subject to security measures.

**Monitoring and Acknowledging Alarms**

The Alarm Viewer allows operators to monitor and acknowledge alarms and to customize the appearance of the window to suit the needs of a project or an operator’s personal preference.

![Alarm Viewer](image)

The bottom panel displays alarms that are in alarm and have been acknowledged or do not require acknowledgment.

One or more alarms can be acknowledged with a single button click.

The alarms that are displayed depend on both developer and operator choices:

- You associate each alarm with an alarm group and then choose which alarm groups can be displayed on each workstation.
- Of the alarm groups that you allow to be displayed on a given workstation, the operator can choose to display all of those alarm groups or selected alarm groups.

The operator can also control the background color and the order in which alarms are displayed.

These characteristics can be saved and recalled at any time on a window-by-window basis.

The operator can display multiple Alarm Viewer windows each with different characteristics – different alarms, different display order, etc.
Archiving

The Process Historical Archiver (PHA) includes a complete set of archive utilities that allow you to configure, implement, and monitor run-time archiving of project data to a disk file (database) that is compatible with the Microsoft Open Database Connectivity (ODBC) standard. The PHA supports any standard ODBC-compliant database that runs on Windows 2000/XP, such as Sybase SQL Server (single user), Microsoft SQL Server (multi-user), etc.

You can then use ODBC-compliant database managers, report writers, expert systems, custom software, and other applications to read and manipulate the archived data.

The Status Monitor displays real-time statistical information about the data passing through the PHA.

At run-time, the PHA filters the tags that are to be archived, based on the user-defined configuration. Only archivable tag changes that meet the user-defined parameters are stored in the buffer. The user-defined parameters are described in the following paragraphs.

If you do not specify otherwise, the PHA will use exception-based archiving to determine which data to store in the buffer. In exception-based archiving, the buffer contains data only for user-selected tags with values that have actually changed. Exception-based archiving eliminates unwanted transitional data and minimizes network traffic.

Exception-based archiving can be overridden for individual tags or groups of tags. You can use frequency groups to specify how often the PHA samples and stores tag values. Within a frequency group you can assign a minimum write rate and a sample rate. The minimum write rate forces the PHA to store tag values at an assigned time interval, and the sample rate determines how often the PHA samples tag values. Any value greater than zero in effect disables exception-based archiving for the affected tags.

For example, let’s say you create a frequency group with a minimum write rate of 1800 seconds (30 minutes). If you assign a tag to that frequency group, the PHA will store the tag’s value every 1800 seconds, regardless of whether or not the tag’s value has changed. You can create more than one frequency group for a project.

You can use deadband groups to specify the sensitivity of the PHA. Deadband groups define how much a tag’s value must change before an exception is generated. For example, let’s say you create a deadband group with a deadband value of 10 units. If you assign a tag to that deadband group and the tag’s value changes by only 5 units during run time, an exception will not be generated and the PHA will not store the tag’s value. On the other hand, if the tag’s value changes by 10 or more units, an exception will be generated and the PHA will store the tag’s value. You can create more than one deadband group for a project.

At the user-defined buffer release interval, the PHA sends a command that causes the buffered data to be archived to the ODBC-compliant database.

You can configure archiving for one or more OWS nodes in a project with each node handling the same or a different set of archiving parameters. Archiving configuration is performed on the OWS node that will actually perform the archiving.

Figure 89. PHA Data Transfer
Logging and Trending

UCOS provides extensive logging, archiving, and trending capabilities for process history and analysis. These functions are available online from the Run-Time Tools menu.

Log Viewer

The Log Viewer lets operators view one or more log files produced by the logging subsystem. This subsystem logs entries from such sources as the alarm subsystem and operator actions. (Log files are created only if you choose to send logable messages to one or more files. You can choose to have each message logged to a file, to a printer, to both, or to neither.)

Any number of log files can be viewed simultaneously and any part of a log file’s view can be copied to the clipboard or printed. Log file contents can be sorted for viewing in chronological or reverse chronological order.

Trending

Operators can format and view trends for real-time or archived data using Trend Viewer. Up to eight points can be plotted in a single trend window with user-selectable plot lines, sample markers, colors, grids, vertical axes, statistical functions, delta values, buffer size, etc.

Multiple trend windows can be displayed at any one time and the characteristics of each can be saved and recalled, as needed.
Device Status and Diagnostics

UCOS provides extensive custom and standard monitoring and control capabilities via command windows, group displays, and project graphics screens.

UCOS also provides the following tools to help test and debug running projects.

Device Point Status

The Device Point Status utility shows the current value for each tag associated with a selected device. Values are updated dynamically.

![Device Point Status Dialog Box](image)

- Displays a user-defined list of tags and tag values from one or more devices and allows the user to manually insert new values for selected tags

Any or all of these perspectives can be viewed individually or simultaneously in multiple windows. Color changes representing control logic status and value changes are updated dynamically in real-time.

Animated logic schemes look just like the logic configured in the Template and Device Definition Editors for user-defined devices. The user can choose to display logic for any or all of the five logic categories.

Device Diagnostics

The Device Diagnostics utility is used to monitor logic as it is executed in the FCU, monitor real-time tag values, and manually override tag values.

The utility offers three perspectives on the FCU’s activity. The utility:

- Uses colors and numbers to “animate” the logic for user-defined devices
- Displays all the tags and tag values associated with a device and allows the user to manually insert new values for selected tags
- User-defined colors indicate current input and output values. By default red indicates a discrete 0 or reset value, green indicates a discrete 1 or set value, and blue indicates an analog value.
- Numbers are also used to indicate current actual input and output values: discrete 0, discrete 1, or an analog number.
- Setpoint, alarm, and fault values can be entered manually, and commands can be sent by double-clicking an appropriate tag. Double-clicking timers, counters, and logic gates displays the execution order, presets, etc.
The Device Point Status window displays all the tags and current tag values for one device.

![Figure 94 Device Point Status in Device Diagnostics](image)

From this window, a tag can be taken off scan and new values can be entered manually (subject to security measures). These new values are inserted into the logic stream, and the results can be viewed in the animated logic schemes. Tags can then be put on scan. A separate dialog can display all tags currently off scan.

![Figure 95 Off Scan Tags](image)

The System Point Status window serves the same purpose and offers the same functionality as the Device Point Status window. The only difference is that the System Point Status window allows the user to specify which tags to display. The tags can come from multiple devices in the project. The list can be saved and recalled later.

![Figure 96 System Point Status in Device Diagnostics](image)

**Network Diagnostics**

The Network Diagnostics feature monitors the communication status between a workstation and its associated FCUs. It can even determine the route taken to a destination, similar to the Tracert function in Windows. This helps the operator diagnose the cause of a communication failure.

![Network Diagnostics Dialog Box](image)
Connecting to Other Software

UCOS is designed to share real-time and historical data seamlessly with external systems and applications.

Real-time data can be exchanged with systems external to UCOS using any of the following industry standards:

- Microsoft Dynamic Data Exchange (DDE)
- OPC (server and/or client, optionally redundant)
- Application Programming Interface via ActiveX

The exchange of data between UCOS and an external system can be accomplished through the use of standard third-party protocol drivers that support the standards listed above. In this case, UCOS usually serves data to the third-party protocol driver. The third-party driver then exchanges data with the external system via a serial or network protocol.

Alternatively, if the external system supports one of the standards listed above, then real-time data may be exchanged directly between UCOS and the external system. In this case, Ethernet is typically used as the physical data communications layer and TCP/IP is typically used as the protocol.

In addition to providing access to real time data, UCOS also provides access to historical archive data. The PHA uses standard, Open Database Connectivity (ODBC)-compliant relational databases to store historical process information. As such, standard Structured Query Language (SQL) calls can be used to query the PHA’s database.

Although UCOS supports a wide range of I/O with built-in I/O drivers, support of the data exchange standards listed above allows UCOS to exchange both real-time and historical data with virtually any external system. UCOS users have access to a large library of standard DDE, OPC, and ActiveX protocol drivers available from hardware manufacturers or third-parties. These proven protocol drivers provide read-write access to UCOS process data.

In addition, the support of these standards allows UCOS users to exchange data with a large selection of third-party software that is DDE-, OPC-, ActiveX-, and/or ODBC-compliant.

UCOS can also request tag values from other software, if so defined during Project Configuration.

This makes it possible to:
- Create custom reports
- Perform calculations
- Insert real-time data into memos and other word processor documents
- View data in a format not directly supported by the operator workstation
- Perform other functions via third-party or custom applications, such as PID loop tuning and expert systems, that support one or more of these protocols

For example, UCOS exposes all process point values to OPC via tag names. To read and/or write UCOS data, the project developer simply identifies those tags that will be available for reading, writing, or both. This is accomplished through fill-in-the-blank dialogs and pop-up tag lists.

In addition, UCOS can interface with third-party servers that communicate with hardware devices. Of course, the types of functions that can be performed on UCOS data depend on the capabilities of the third-party or custom software accessing the data.
Communicating with Field Devices

Field Control Unit (FCU)

The FCU is a PLC or personal computer that runs under just about any POSIX-compliant operating system, such as Linux, QNX, UNIX, Windows, etc. The controller software executes sequential and regulatory control schemes that are downloaded from the engineering workstation. It also provides real-time scanning and control of remote I/O.

UCOS supports selective redundancy at all levels, including redundant FCUs and redundant I/O points.

Each FCU is directly connected to the same network as the engineering and operator workstations. Flash disks, non-volatile memory, and an optional internal backup power supply help ensure continuous processing.

At run-time, the FCU software:

- Receives commands and data from one or more operator workstations and/or one or more input/output interfaces that communicate with field devices (e.g. pumps, valves, etc.)
- Processes and solves in real-time all logic generated during the project development phase
- Communicates the results to the operator workstation and/or input/output interfaces

Logic processing is performed by the FCU according to the schemes developed during project configuration on the EWS. The processing is completely event-based. Every device configured in a device diagram is represented in a device table residing on the appropriate FCU. All the logical relationships are represented in a device-specific event list that logically relates source (occurrence) and destination (triggered response) events.

When an event occurs at run-time, the logic processing engine uses the appropriate event list to determine what action (if any) to execute. That action may, in turn, trigger other events and actions. The real-time executive ensures that critical control functions take precedence over other operations, and that logic is executed in the correct sequence.

Real-time information and device control status are stored in the FCU’s logic solver. Data updates are reported by exception. This frees the system from the scan rate limitations that can affect real-time processing and reporting of data at the operator workstation. It also minimizes network traffic allowing FCUs to be located remotely.

The following figure shows the flow of data from project configuration to the FCU and OWS. The project is downloaded to the FCU, then to the OWS. The FCU solves the logic internally and sends the data to the I/O network.

![Figure 99 FCU Controller Subsystem](image_url)

The FCU provides significant support for the unique logic categories used by UCOS. For example, mutual exclusivity of device states is enforced by the FCU. The developer does not need to write code to support mutual exclusivity of device states.
Standard vendor-manufactured I/O scanner cards are plugged directly into the FCU. These cards provide access to I/O networks, which are scanned directly by the FCU.

UCOS supports multiple vendor I/O subsystems at all levels. A variety of distributed I/O networks can be accommodated in the same or different FCUs.

The FCU supports all required distributed control functionality, including:

- Control scheme execution
- Data acquisition and calculation
- Regulatory, discrete, and sequential control
- Interlocking
- Event-initiated processing
- I/O scanning
- Industry-standard, POSIX-compliant operating system
- System communication via single or dual wireless TCP/IP Ethernet connection
- Direct scanning of I/O

### SCADA Data Server (SDS)

The SDS can communicate with devices that cannot be directly scanned by the FCU.

The SDS is a generalized, flexible, and user-configurable application designed to interface UCOS to other control systems, proprietary protocol drivers, or specialized software. Software protocol drivers may be supplied by CSI or by other third-party suppliers. The protocols run as separate tasks on the SDS and communicate to devices external to the control system, such as PLCs, RTUs, flow computers, intelligent valve controllers, Fieldbus, etc.

The data exchanged with the external devices or other software systems is then “served” to the control system database so the data can be used by logic, graphics, and other applications executing in the control system environment. Also, commands and setpoints initiated from the OWS are routed to the SDS application that, in turn, routes them to the appropriate external application.

The SDS has the following features:

- Supports both serial and TCP/IP connections
- Supports off-the-shelf, industry-standard ActiveX, DDE, and OPC drivers
- Communicates with any RS-232 compliant equipment
- Supports enabling/disabling of COM ports and polling lists
- Supports multiple ports
- Supports multi-drop slaves configuration on the RS-485 communication bus line
- Allows creation of a poll list for each port
- Supports unsolicited write
- Includes an FCU and data concentrator in one unit
- Can be supplied in a redundant, low-, or high-point-count configurations
About Control Systems International (CSI)

UCOS is just one of the open systems solutions from Control Systems International, Inc. (CSI), a leading suppliers of industrial automation systems since 1968.

For more than 35 years, demanding industries have relied on CSI for automation and instrumentation solutions. Our primary strength and value for your project is our ability to design and implement total, cost-effective automation systems – and to plan and manage large, turnkey projects.

Founded in 1968, CSI pioneered the application of microprocessor and networking technology to real-time control problems. By the end of the 70s, we were supplying complete distributed control systems for plant control and information management. By the mid-80s, CSI was serving the demanding petrochemical industry with innovative solutions for total facility and distribution system automation.

Today, CSI is a global presence, offering comprehensive control and instrumentation solutions for a broad range of industries and applications, from water/wastewater management to the distribution of fuel and other bulk liquids. Our technical depth and specific industry experience complement your knowledge of your business. CSI's solutions – such as UCOS – are designed to meet your specified standards for performance and ease of use, maintenance, and expansion. With hundreds of systems installed worldwide, CSI's track record of success speaks for itself.

But these aren't the only reasons you should consider CSI for your next automation project. Unlike other suppliers of "total system" or "one stop shop" solutions, CSI does not manufacture hardware platforms or I/O subsystems. Our solutions are always designed to conform to open systems standards.

So, you are not tied to a single vendor for spare parts and service. Our engineers and software specialists are experienced with many types of open, compatible equipment that can enhance your core solution. You don't have to settle for a second-best or less cost-effective component. More importantly, you are assured that your solution can continue to evolve as technology advances.

CSI – delivering the best value in integrated, open process automation and information management systems.
Notice

All procedures, data, information, drawings, specifications or other material, whether accompanying this document or separately supplied in furtherance of this document, contain confidential and proprietary information which (i) is the property of Control Systems International, Inc. (“CSI”), (ii) is disclosed by CSI only in confidence, and (iii) except as CSI may otherwise permit in writing, is to be used, disclosed or copied only to the extent necessary for the evaluation and use thereof by the recipient. The foregoing shall not apply to any such material to the extent that the contents (i) are now or subsequently become available to the public without payment, (ii) were previously known to the recipient, or (iii) subsequently become otherwise known to the recipient without restriction.

This document is based on information available at the time of its publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware and software, nor to provide for every possible contingency in connection with installation, operation and maintenance.

CSI makes no representation or warranty, either expressed or implied, with respect to, and assumes no responsibility for, the accuracy, completeness, or sufficiency of the information contained herein. No warranties of merchantability or fitness for a particular purpose shall apply.

This document takes precedence over and supersedes in their entirety all previous versions or revisions.


All other product and company names/logos mentioned in this document are trademarks and/or registered trademarks of their respective holders.


8040 Nieman Road, Lenexa, KS 66214-1523

CSI Document No. K-9506-PTS-1-MKTG-11072005-MJK
CSI Form No. K-9906-DOT-6-MKTG-23081999-MJK-R02