

UCOS[®]

APPLICATION NOTES

Control Philosophy and Standards for UCOS Project Design and Development

Document: K-9206-APP-2-MKTG-06032002-RL

Applies to: All versions of UCOS

Authorized: Vice President for Research and Development

Coordinator: Director of Marketing Communications

Control Systems International, Inc.
8040 Nieman Road
Lenexa, Kansas 66214

Abstract

This document presents a guideform UCOS project development methodology for general, industrial control system applications. It is intended to be used by project engineers as an 10-step approach to the design of control system solutions using CSI's UCOS product. The approach presented herein is not application specific; it provides a generalized means to produce control systems which will meet specified functional requirements in the most efficient manner possible (i.e., with minimal engineering effort).

Release Summary

22-Mar-1999

Initial release

Contents

Purpose and Scope	3
Responsibilities	3
References.....	3
The 10 Major Steps to Develop a UCOS Project	4
Step 1: Identify Devices	6
Step 2: Identify All Relationships Among Devices	7
The Plant Control Hierarchy	7
The System FBD	8
Step 3: Design System Architecture	9
Step 4: Identify All Required Templates	9
Step 5: Build and Test Templates.....	10
Step 6: Generate Devices.....	10
Step 7: Assign Alarm/Log Groups and Real World I/O.....	10
Step 8: Build Device Diagrams	10
Step 9: Build Graphics	11
Step 10: Generate Project Documentation	11
Appendix A: Example Device List	12
Appendix B: Example Device Associations Document	13
Appendix C: Example Template Documentation	14

Purpose and Scope

This document presents a guideform UCOS project development methodology for general, industrial control system applications. It is intended to be used by project engineers as an 10-step approach to the design of control system solutions using CSI's UCOS product. The approach presented herein is not application specific; it provides a generalized means to produce control systems which will meet specified functional requirements in the most efficient manner possible (i.e., with minimal engineering effort).

To these ends, this document identifies common UCOS system design requirements and defines the vocabulary used within the product. Graphical techniques are also presented which assist with the design process and further ensure consistency among UCOS projects. The design approach presented also stresses time saving features of the product such as UCOS templates and the standard UCOS library of templates.

Responsibilities

This document is intended for use by CSI employees only. This document is considered the property of CSI and care should be taken to protect the information it contains.

References

For more information about developing UCOS projects, see the *EWS Reference Guide*.

The 10 Major Steps to Develop a UCOS Project

The following steps are detailed in this section of the document:

1. Identify all devices, such as each pump, valve, conveyor, etc.
2. Identify all relationships between devices. Establish the interfaces among the objects. What information or signals will be needed between the devices?
3. Design the system architecture. How many operator workstations (OWS) and field control units (FCU)? Define your I/O subsystem.
4. Identify all required templates. How many similar devices can be templated?
5. Import or build and test all templates. Make sure all logic and graphics work as designed before you mass-produce devices.
6. Generate devices. Create all devices from templates and any one-off devices.
7. Assign alarm/log groups and I/O. Use import/export to rapidly assign tag attributes.
8. Build device diagrams. Organize your project so other engineers can work on the project.
9. Build graphics. Design and build the operator interface.
10. Generate device documentation. Export and use Access to create device documentation.

The following flowchart illustrates these ten steps.

Control Philosophy and Standards for UCOS Project Design and Development

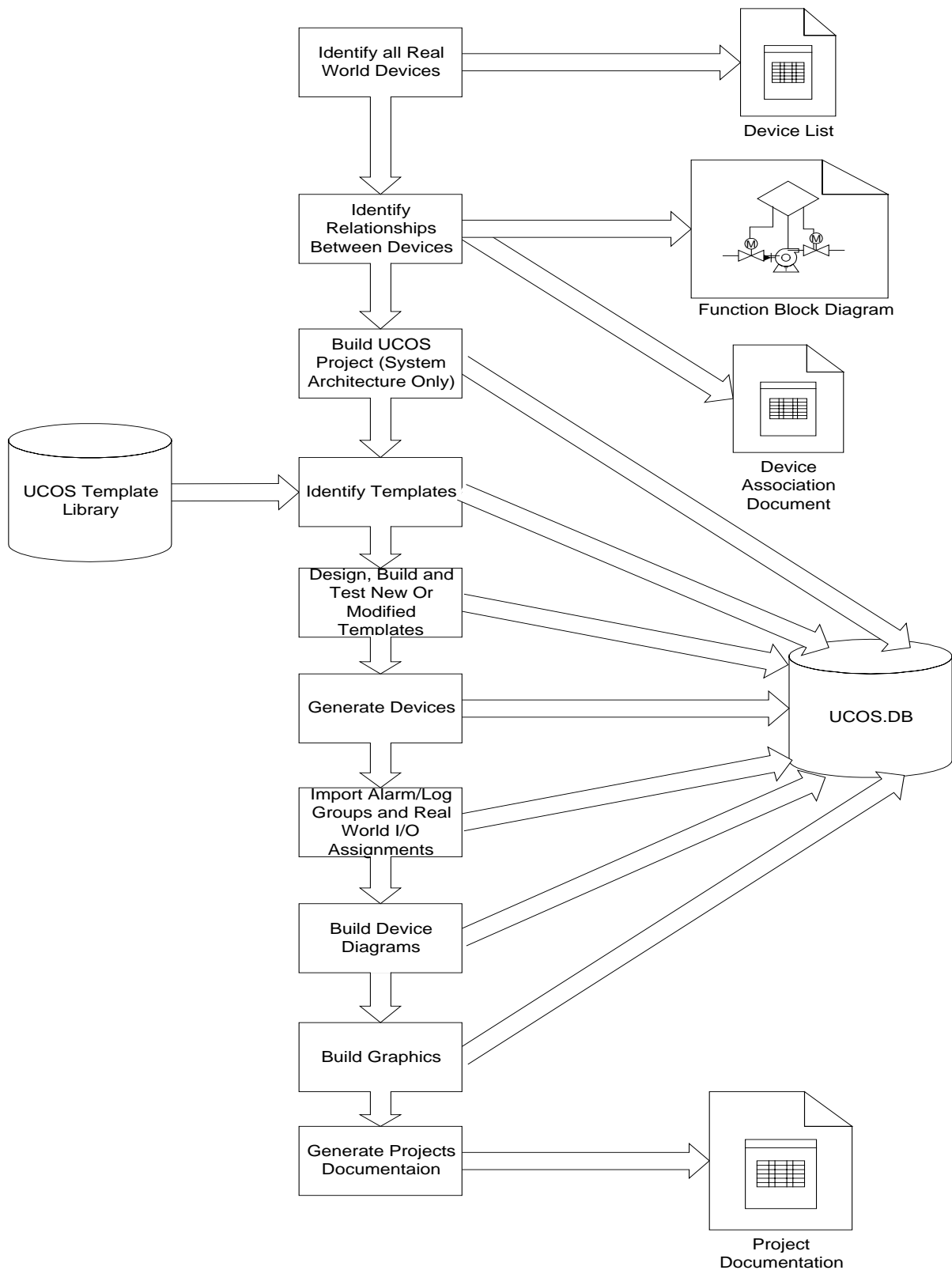


Figure 1. Ten Steps to Develop a UCOS Project

Step 1: Identify Devices

The first step in building a UCOS project is to define all of your real-world devices. The reason this is an important step is that the structure of a UCOS project parallels the way that engineers naturally look at industrial controls. To demonstrate this, picture in your mind a simple process with which you are very familiar. For the purpose of example, consider the following description of a simple process:

“A cooling water delivery system consists of a 400 GPM centrifugal pump with 4 inch motor operated suction and discharge valves. A 3 inch pressure control valve (PCV) is also located downstream of the pump discharge valve to maintain a maximum discharge pressure of 50 PSI downstream of the control valve. Both the suction and discharge valves are opened fully prior to pump start and are closed any time the pump stops. If the pump is stopped, the PCV is forced closed. Immediately after pump start, the constant discharge setpoint of 50 PSI is fed to the PCV control loop.”

In the preceding example, note how the simple process was described. It was described in terms of the devices which physically control the process. Also note that discrete and analog I/O was not used to describe the process.

CSI believes that engineers most naturally dissect industrial processes into the field devices which make up the process. Unfortunately, most industrial control systems force engineers to first describe processes in terms of devices, then further decompose the control of those devices in terms of controlling a disjointed set of I/O.

Again, using the preceding example of a cooling water delivery system, begin to assemble the logic, which would be required for a PLC program to control the process. Your thought process would probably proceed somewhat like this.

“Well, first I’ll need an I/O list. If the pump run output is off, then I’ll need to close both the suction and discharge valves by energizing the ‘close’ MOV contactors until the valves are fully closed. If I receive a pump run request, then I’ll have to energize the ‘open’ MOV contactors until the valves are fully open ...” and so forth.

This brief example is intended to demonstrate that the way in which a PLC is programmed forces an engineer to design and implement in terms of I/O. The result is a control program which is structured by I/O, not by devices.

In contrast, UCOS forces logic structures which reflect the real-world structure of a process, expressed in terms of devices. By enforcing this organization, control system problems are simplified and the resulting solutions are naturally reusable as development standards across projects. In place of a large, simple program which controls an entire plant or facility, UCOS allows a smaller collection of control devices to be established. Functional interrelationships between the devices then accomplish the overall plant or facility functionality.

Within software development, this approach to design is known as an “object-oriented” approach. The UCOS product was specifically designed to apply object-oriented methodologies to the implementation of industrial control systems. Therefore, UCOS dictates that an object-oriented approach is used throughout the project development process (requirement definition, design, and implementation).

The concepts introduced above are quite simple. However, the method and thoroughness used to implement this concept is very important. Tools and methods are addressed later in this document. As you continue, realize that this type of top-down design must be considered early in the project development cycle. This includes hardware and system design, I/O and controller component selection, FCU capacity limitations, device structure limitations, knowledge of the low-level UCOS instruction set, etc.

At this point, the most important thing to remember is that a UCOS project and the underlying logic must be structured to reflect individual real-world devices or sets of devices, and the functional relationships among the devices. UCOS product configuration must not commence until the structure (design) of the control system is completed using the UCOS project structure.

Before any functional relationships or logic can be defined, a complete device list must first be generated (or as complete as possible at this stage). Appendix A is an example of a device list, your first step in building a UCOS project.

Step 2: Identify All Relationships Among Devices

Once all devices have been identified, it is time to define all of the functional relationships among devices. The best tool to help you visualize and document these relationships is the Function Block Diagram (FBD). The FBD identifies the major devices to be controlled and illustrates a high-level, functional relationship between those devices. (An example FBD is provided in Figure 3.)

The Plant Control Hierarchy

Before constructing an example FBD, it is necessary to introduce (and understand) the concept of “the plant control hierarchy”. All process plants or facilities consist of a number of controlled units that carry out various functions. These units are typically electric drives and valves. They are usually grouped into sub-plants, which have a common function within the plant. To logically group these devices, the engineer should formally recognize the three levels: Plant, Group, and Unit. Figure 2 illustrates a typical Plant Control Hierarchy. This is mainly used as a convention to identify physical breakup of the plant for control purposes. Using this approach, devices are identified as belonging to one of the three specified levels. This grouping can be used later to help define Device Diagrams.

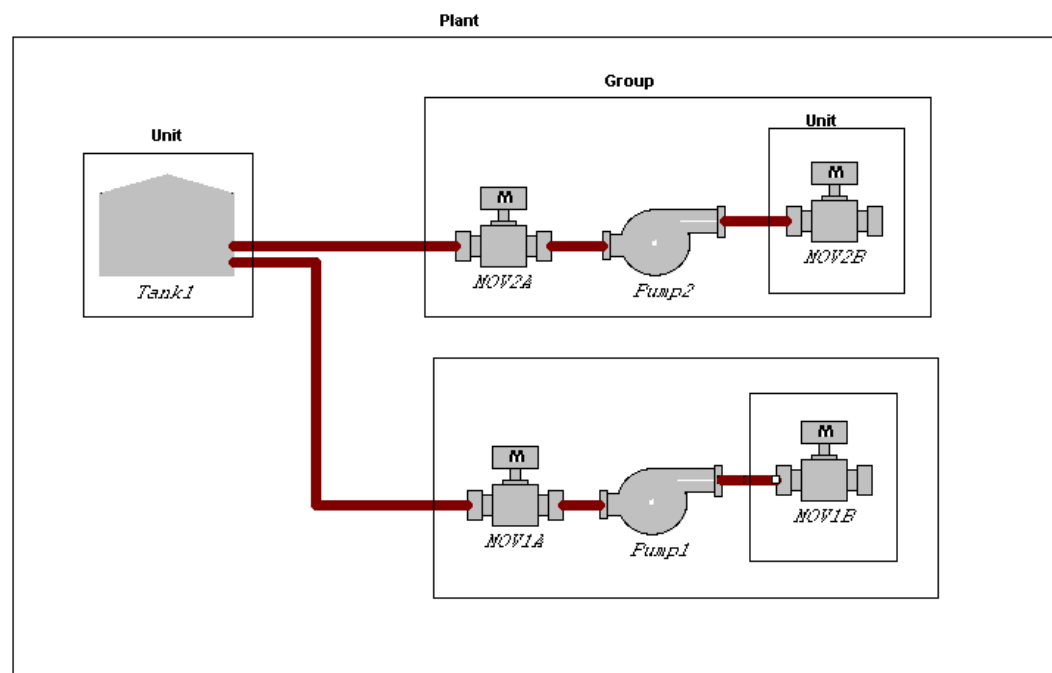


Figure 2. Plant Control Hierarchy

The System FBD

In Figure 2 there are three distinct levels of control.

The first level is Unit control. This is where all logic that pertains only to that device resides. This will include all of that device's states, alarms, and faults and any device specific modes like auto/manual or runtime-hours for a given pump.

The next level is Group. Let's say the start up sequence for Pump 1 is:

1. Open the suction valve
2. Start pump
3. Open the discharge valve

An additional device that contains all logic associated with this group sequence would perform this logic. This logic will again include all sequence-specific states, alarms, and faults, and if required any sequence specific modes like auto/manual. It is very likely that a project will require auto/manual for each device and also auto/manual for each sequence. This level will not include permissives or interlocks. Permissives and interlocks should go directly between the two units that have interdependencies.

The third level of control is Plant control. This would be an additional device that would handle sequencing among groups. Let's say the project required that we start the group with the fewest run hours. This plant controller device would examine the pump run hours of the two pumps and send a sequence start command to the group controller with the fewest run hours.

After you are able to decompose a plant into Plants, Groups, and Units, you are ready to construct FBDs.

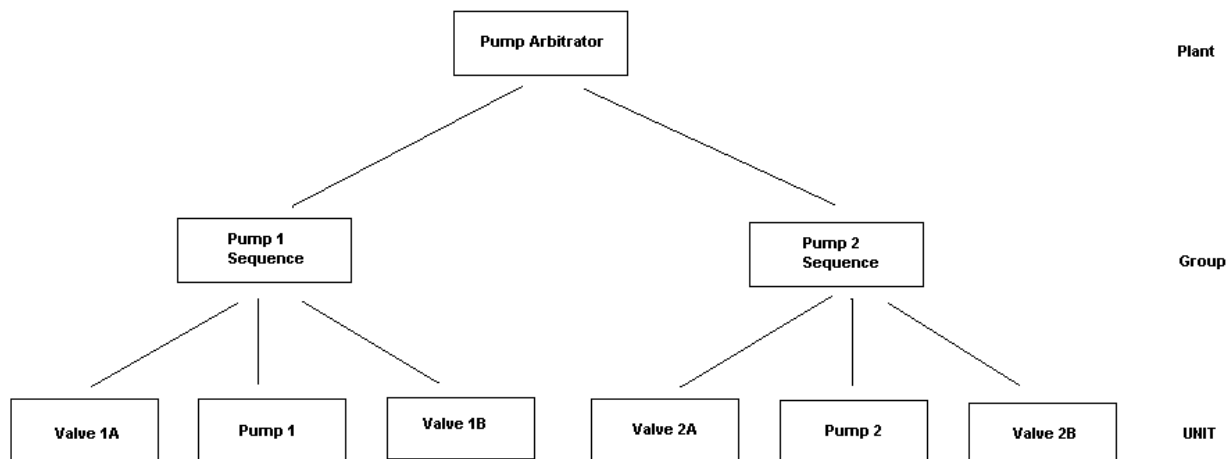


Figure 3. Pump Station Control System FBD

Figure 3 is a simple FBD for the pumps and MOV's in Figure 2. Note that at the sub-Plant level, a "Pump Arbitrator" is shown which will decide when Pump 1 or Pump 2 is to be run or stopped. At the Group level, "Pump Sequencers" are responsible for receiving run/stop requests from the Pump Arbitrator and subsequently commanding the pumps to run/stop and commanding the valves to open/close as required by the associated pump's operational state. Finally, at the Unit level, control and monitoring of individual devices is accomplished.

Control Philosophy and Standards for UCOS Project Design and Development

Now is the time to start thinking about how your unit control will be designed. For example, has the customer specified that each device (Unit) will have it's own auto/manual or will a Group auto/manual work best? This needs to be decided before templates can be chosen or built. As discussed earlier, it typically works best to have an auto/manual mode for each device and an auto/manual for each sequence.

It is important to note that not all data flows and no I/O should be included in the FBD. If the data flow is intuitive to a control engineer, then it is intentionally omitted to avoid cluttering the diagram. Later we will discuss how the Operations Requirement Specification (ORS) is used next to specify functionality, formulas, and other details which should not be contained in the FBD.

As part of the FBD we also need to identify and document Device/Unit control interlocks and permissives. An interlock is defined as a condition or set of conditions that are required to maintain a device's state and are also required to change the state of the device. A permissive is defined as a condition or set of conditions that must be met in order for a device to change state.

For example, in Figure 3 the pump will be allowed to start and maintain a running status, if the suction and discharges valves are open (interlock). Let's say that the pump requires that the case be vented at least 60 minutes before the pump is requested to start (permissive). In the previous examples the interlock would be describe in the FBD. However, the venting of the pump is associated with the pump device – not a sequenced device. Nevertheless, it has to be documented.

You will find that in some cases interlocks and/or permissives are devices. For example, a Pressure Switch, Flow Switch, Vibration Signal, etc. may dictate whether or not the pump can start or remain running. These real world devices should be included in the pump device as an interlock or permissive. In Appendix B you will find an example of the Device Associations Document (DAD) which will be used in documenting interlocks and permissives not indicated on the FBD.

Step 3: Design System Architecture

At this stage of project development we are ready to create our UCOS project and build the system architecture. This will include all OWSs, Archivers, and FCUs and their associated I/O subsystems. This is also the best time to define all logging groups, logging devices, alarm groups, alarm priorities, and security profiles.

Step 4: Identify All Required Templates

After the FBDs and DADs are developed for the project, cooperative brainstorming sessions should be held with members of other project teams that have completed similar projects. Using the FBD, everyone in the cooperative design session should be able to rapidly grasp the process to be controlled and the general functionality required of the system. Given this, the participants should identify UCOS device templates which were created or used in previous projects that could be donated to the new project. Assuming a healthy pool of donors, a significant portion of your Template Operations Requirement Specification (ORS - the next design tool) and device templates may already exist.

Step 5: Build and Test Templates

Once all required templates are identified, it is now time to build unique or modify existing templates. Building templates consists of several steps:

1. Defining functionality
2. Filling out tag definition sheets (ORS)
3. Defining logic
4. Generating graphic templates

Note: Steps 2 and 3 will probably be done in parallel.

Template Documentation should use the format defined in the example in Appendix C.

Step 6: Generate Devices

After the customer has approved all templates, we can now generate devices. All devices should be generated and placed in the device list if using UCOS 3.2 and before or by using the batch instantiator if using UCOS 3.2.1 or later. Do not place devices on Device Diagrams at this time.

Generate a report of all unresolved wild cards using the reporting tool in Device Diagrams and resolve all wild cards.

Now is the time to generate any one-off devices that do not make sense to have templated. These will typically be plant sequence devices that only appear once in a project or maybe a device that contains several devices. An example of a device that contains several devices might be in a project that has six high inputs that signify the lights are on in the parking lot. These inputs have no affect on any control sequence and are used only for indication on a graphic screen. These inputs can be placed in one device to keep from using six devices with little, if any, logic in each device.

Step 7: Assign Alarm/Log Groups and Real World I/O

After all of the devices in a UCOS project have been created, it is time to export all devices in the database. These devices will be exported by UCOS as comma separated variable (csv) files. Open these files with Excel and assign all alarm groups, log groups, real world I/O, and security levels to all commands and setpoints. Save them and re-import them back into UCOS.

While this assignment functionality exists in UCOS, you may have patterned needs that can be implemented faster using Excel.

Step 8: Build Device Diagrams

Device Diagrams are unique to the UCOS product. If you remove the Plant, Group, and Unit labels from Figure 2, a typical UCOS Device Diagram would result. By now you should have learned about UCOS Device Diagrams in previous UCOS training sessions. As you recall, the Device Diagram identifies within UCOS all of the devices which are to be controlled by the system. It also provides an illustration of how the devices are physically interconnected plus the closed-loop control which is to be accomplished by the system.

Now you must recognize how devices are logically extracted or selected from the plant P&IDs and placed on the Device Diagram. This is done in terms of sub-Plants, Groups, and Units as described previously.

The highest level of device design in UCOS should be intuitive. Again, referring to Figure 3, it should be obvious that the Pump Arbitrator decides if Pump 1 and Pump 2 should be run or stopped, the Pump Sequencers at the Group level coordinate operation of the two pumps and the valves associated with each, and so forth. This functional relationship is graphically represented on UCOS Device Diagrams.

A Device Diagram's major functionality is two fold. First it gives the plant engineer or technician a graphical guide to his plant logic that he is already familiar with. The vast majority of plant personnel have been using P&IDs to troubleshoot their plants for years, and Device Diagrams are an intuitive way to locate a device that he needs to troubleshoot or modify. The second major benefit of Device Diagrams is to give the project development team an easy way to find devices during project implementation. It is much easier to find a graphical representation of a valve than to try to remember the individual names of each device in a project.

Once functional relationships are established, it is time to design UCOS logic that will accomplish the desired real-world control.

Think of Device Diagrams as an organizational tool. Trying to modify a UCOS project from only the device list would be like trying to work in a PLC program that was organized with all of the rungs entered in alphabetical order.

Step 9: Build Graphics

After all of the devices have been completed, we can now start to assemble the graphic screens. It is recommended that you start out with a screen standard. What will the background color be? How will screen navigation be accomplished? After these types of questions have been answered, a screen template should be built and copied for every screen. Bitmaps can then be placed on these screen templates, the graphic symbols can be dropped, and any static lines can be drawn manually.

Step 10: Generate Project Documentation

After the graphics have been built, it is time to produce the UCOS-specific project documentation. This is accomplished by exporting all devices and hardware to csv files and importing the csv files into an Access database where pre-configured reports can be generated for all devices and the hardware configuration. Printouts of all screens, device diagrams and schema prints should also be included in the project documentation.

Appendix A: Example Device List

Device	Location	Description
PMP-G102A	Truck Loading	Diesel Loading Pump
PMP-G102B	Truck Loading	Diesel Loading Pump
PMP-G102C	Truck Loading	Diesel Loading Pump
PMP-G102D	Truck Loading	Diesel Loading Pump
TSHH-1257	Truck Loading	Temperature Switch Hi Hi
TSHH-1258	Truck Loading	Temperature Switch Hi Hi
TSHH-1259	Truck Loading	Temperature Switch Hi Hi
TSHH-1260	Truck Loading	Temperature Switch Hi Hi
PSLL-1262	Truck Loading	LoLo Suction Pressure Switch
PSLL-1263	Truck Loading	LoLo Suction Pressure Switch
PSLL-1264	Truck Loading	LoLo Suction Pressure Switch
PSLL-1265	Truck Loading	LoLo Suction Pressure Switch
FSLL-1267	Truck Loading	LoLo Flow Switch
FSLL-1268	Truck Loading	LoLo Flow Switch
FSLL-1269	Truck Loading	LoLo Flow Switch
FSLL-1270	Truck Loading	LoLo Flow Switch
TK-100A	Tank Farm	Diesl Storage Tank (14 Meters)
TK-100B	Tank Farm	Diesl Storage Tank (14 Meters)
TK-100C	Tank Farm	Diesl Storage Tank (14 Meters)
TK-100D	Tank Farm	Diesl Storage Tank (14 Meters)
LSHH-101	Tank Farm	HiHi Level Switch (Set at 13.5 Meters)
LSHH-102	Tank Farm	HiHi Level Switch (Set at 13.5 Meters)
LSHH-103	Tank Farm	HiHi Level Switch (Set at 13.5 Meters)
LSHH-104	Tank Farm	HiHi Level Switch (Set at 13.5 Meters)
LSLL-105	Tank Farm	LoLo Level Switch (Set at 1 Meter)
LSLL-106	Tank Farm	LoLo Level Switch (Set at 1 Meter)
LSLL-107	Tank Farm	LoLo Level Switch (Set at 1 Meter)
LSLL-108	Tank Farm	LoLo Level Switch (Set at 1 Meter)

Appendix B: Example Device Associations Document

Device	Interlock	Description
PMP-G102A	TSHH-1257	Temperature Switch Hi Hi
	PSLL-1262	LoLo Suction Pressure Switch
	FSSL-1267	LoLo Flow Switch
PMP-G102B	TSHH-1258	Temperature Switch Hi Hi
	PSLL-1263	LoLo Suction Pressure Switch
	FSSL-1268	LoLo Flow Switch
PMP-G102C	TSHH-1259	Temperature Switch Hi Hi
	PSLL-1264	LoLo Suction Pressure Switch
	FSSL-1269	LoLo Flow Switch
PMP-G102D	TSHH-1260	Temperature Switch Hi Hi
	PSLL-1265	LoLo Suction Pressure Switch
	FSSL-1270	LoLo Flow Switch

Appendix C: Example Template Documentation

Pump – Shipping, Stop Pulse High

Name

PUMP-200SHIP

Typical Use

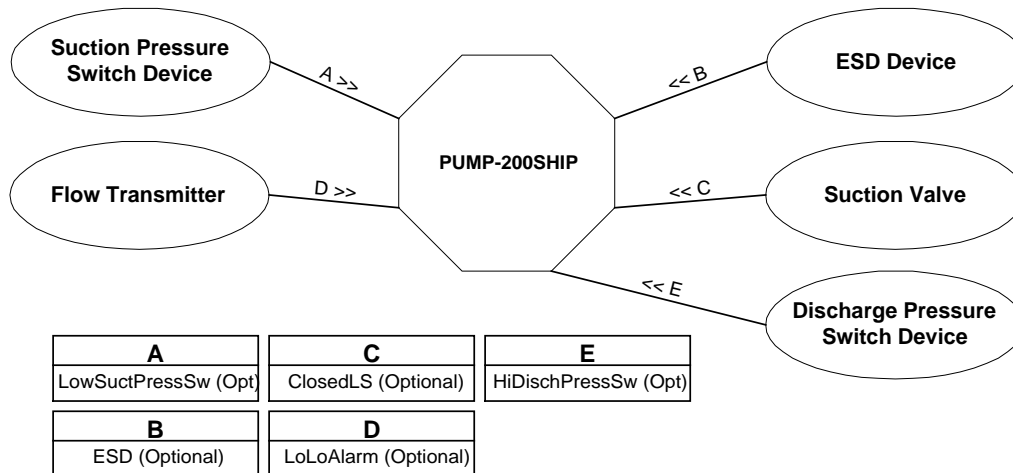
This template is designed to represent a dual output centrifugal petroleum shipping pump. It should be used for pump controllers that are expecting two pulsed outputs which are both pulsed high to start and stop the pump.

Overview

This pump template has two pulsed outputs for starting and stopping the pump. The start and stop outputs are pulsed high to start and stop the pump. If a fault is detected the system will attempt to stop the pump.

The pumps runtime hours and cycles are calculated for display to the operator. The runtime hours can be reset by operator command.

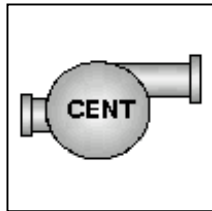
Data Flow Diagram



Template Graphic



Device Diagram Symbol



(CPump.usy)

Command Window Layout

The screenshot shows a software window titled "PUMP_" with a menu bar containing "File", "Commands", and "Show". The main interface is divided into several sections:

- Control Buttons:** A vertical column of buttons on the left side, including "START", "STOP", "AUTO", "MANUAL", "OFFLINE", "SET HOURS", and "RESET FLT".
- Loading Pump:** A section containing several data fields:
 - State:** A field displaying "STOPPED".
 - Modes:** A field displaying "REMOTE" and "AUTO".
 - Alarms:** An empty field.
 - Faults:** An empty field.
- Analog Values:** A table with two columns: "Analog Values" and "Value".

Analog Values	Value
Cycles	0.0000000
RunHours	0.0000000
- Setpoints:** A table with two columns: "Setpoints" and "Value".

Setpoints	Value
RUN HOURS	0.0000000
START TIME	5.0000000
STOP TIME	5.0000000

Logic Descriptions

Modes

Mode Group 1 – Auto/Manual mode group

This mode group is changed from one mode to another by operator commands. If the pump is in Automatic mode the *AutoStart* and *AutoStop* tags are enabled to start and stop the pump. If the valve is in manual mode the *StartCMD* and *StopCMD* tags are enabled to start and stop the pump. In off-line mode the pump cannot be controlled by UCOS either automatically or manually. Null1 mode is used if there is no automatic/manual modes for the pump and will not display a mode for this group in the command window. The pump is placed in Null1 mode only by configuring this mode to be the default mode for the group.

Mode Group 2 – Remote/Local mode group

This mode group is changed from one mode to another by a digital input *RemoteSwitch*. When the pump is in remote mode the pump can be controlled by UCOS according to the Auto/Manual mode group. In local mode the pump cannot be controlled by UCOS. Null2 mode is used if there is no *RemoteSwitch* input for the pump and will not display a mode for this group in the command window. In this case the pump operates as though it is in remote mode. The pump is placed in Null2 mode only by configuring this mode to be the default mode for the group.

States

The states for the template indicate the status of the pump. The pump can be Running, Stopped, Starting, Stopping, or LocalRun. The pump is in LocalRun state when the pump runs without being commanded to by UCOS.

Alarms

The *FalseRun* alarm is generated when the pump runs without a command from UCOS and the pump is not in local mode.

The *StartDisable* alarm is generated if the pump is not allowed to start and a start signal is sent.

The temperature alarms compare the real world inputs from thermocouples or RTDs with setpoints for the temperature. If these temperature signals are 4-20mA signals a transmitter should be used and the SCALE_VALUE from the transmitter should replace the real world analog input tag.

Faults

The *FailToStart* and *FailToStop* faults are generated when the pump has been in starting or stopping state for the preset time limit.

The *RunInterlock* fault is generated based on the associated permissives required for running the pump.

The *LowSuctionPress*, *HiDischargePress*, *LowFlow*, and *SealLeak* faults are generated based on input signals or tags from other devices.

The *FalseStop* fault is generated if the pump is stopped while in automatic mode without a command from UCOS.

The *ESD* fault is generated by a signal from another device and is used to stop the pump on a plant ESD.

Control Philosophy and Standards for UCOS Project Design and Development

The *CycleTrip* fault is set if the number of cycles is greater than the *MaxCycles* setpoint

The *Temperature* fault is set if any of the temperature inputs are greater than the setpoint for the temperature inputs. This fault is different from the temperature alarms in that it is a single fault for all of the temperature inputs. If these temperature signals are 4-20mA signals a transmitter should be used and the *SCALE_VALUE* from the transmitter should replace the real world analog input tag.

With the exception of the *ESD* and *CycleTrip* faults, all faults are latched and must be reset by operator command.

Control

Page 1

The *RunOutput* and *StopOutput* tags are pulsed on for five seconds when the *Starting* or *Stopping* state becomes high.

The *RunMinutes* tag is calculated by adding one to itself every time the *RunReadback* tag is on for an accumulated time of one minute. The *RunMinutes* tag is a retentive tag and is divided by 60 to determine *RunHours*.

The *AutoOpEnabled* tag is used by other devices to determine whether an auto open command to the valve will be acted upon or not.

Page 2

The *Cycles* tag indicates the total number of pump starts within the preset time period.

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.

The information contained in this document is subject to change, without notice, at any time and should not be used as a specification for any UCOS product nor for any particular UCOS project.

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.

VXL, FUEL-FACS, UCOS, and the CSI logo are registered marks of Control Systems International, Inc. Control Systems International and CSI are trademarks of Control Systems International, Inc. UCOS U.S. Pat. 5,812,394.

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

Copyright © Control Systems International, Inc. 1999. All Rights Reserved.
8040 Nieman Road, Lenexa, KS 66214-1523
Telephone: 1-913-599-5010 Facsimile: 1-913-599-5013
<http://www.ucos.com>

CSI Document Number: K-9206-APP-2-MKTG-06032002-RL