SANDIA REPORT

SAND2011-7374
Unlimited Release
Printed October 2011

Refinery Burner Simulation Design Architecture Summary

G.M. Pollock, R. D. Halbgewachs, and M.J. McDonald

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.osti.gov/bridge>

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: <http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online>

Refinery Burner Simulation Design Architecture Summary

G.M. Pollock (Cyber Research and Education);
R. D. Halbgewachs (Effects-Based Studies); and M.J. McDonald (Effects-Based Studies)

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1914

Abstract

This report describes the architectural design for a high fidelity simulation of a refinery and refinery burner, including demonstrations of impacts to the refinery if errors occur during the refinery process. The refinery burner model and simulation are a part of the capabilities within the Sandia National Laboratories' Virtual Control System Environment (VCSE). Three components comprise the simulation: HMIs developed with commercial SCADA software, a PLC controller, and visualization software. All of these components run on different machines. This design, documented after the simulation development, incorporates aspects not traditionally seen in an architectural design, but that were utilized in this particular demonstration development. Key to the success of this model development and presented in this report are the concepts of the multiple aspects of model design and development that must be considered to capture the necessary model representation fidelity of the physical systems.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the National SCADA Test Bed (NSTB) program at the United States Department of Energy's Office of Electricity Delivery and Energy Reliability, which provided the funding for this research. We further acknowledge Brian Nelson, Sandia National Laboratories, who developed the Reboiler and Detailed Reboiler HMI.

Contents

Abstract	1
Executive Summary.....	7
INTRODUCTION	9
SIMULATION OVERVIEW	10
System Component View	12
Umbra Simulation Model	12
HMI VMWare	12
PLC-SCADA Controller	15
Network View	19
Establishing Desired Network	19
IP Addresses.....	20
Network Protocol	20
Event View.....	20
HMI – Burner States.....	22
HMI – Reboiler Info.....	24
HMI – Reboiler Detailed Info	26
Data Flow View	26
Main System Components	26
Burner System	28
Refinery System	30
Single Data Flow Trace.....	33
Visualization Data Flow	34
Mechanical View.....	34
Burner.....	34
Controller	37
HMI.....	38
Refinery	38
Physics View.....	45
Sensors	45
Flame	45
Fluid Flow	45
Heat	46
System Safety View.....	46
Gas Build-up	46
Presence of Flame	47
Gas Pressure Interlocks	47
Heat Calculation	47
Ladder Logic View	47
Timers.....	47
Switches	48
Lock-On	48
B10s -- Input	48
B11s -- Output.....	48
B20s -- Output.....	49
B21s -- Input	49
Buggy Ladder Logic Program	49
HMI Views	54
Refinery Reboiler HMIs	54
Refinery Burner HMI	72
Software Component View.....	75
Burner System Component	75
Storage Tank Controller	76

Cooking Tank Controller	76
Components Logical View	76
 Burner Component Logic.....	76
 Storage Tank Logic.....	76
 Cooking Tank Logic	76
Conclusions	77
Distribution:	79

FIGURES

Figure 1: Refinery Burner System Component / Data Flow Overview.	10
Figure 2: Umbra Visualization of Refinery and Burner.	13
Figure 3: Reboiler System HMI.	13
Figure 4: Refiner Burner HMI – Model States / Actions	14
Figure 5: HMI Reboiler Design 2 – Expanded Details; Running.....	14
Figure 6: PLC Ladder Logic Program State 0 – Waiting for “Power On” to be Pressed.	15
Figure 7: PLC Ladder Program States 1 and 2 – Burner Lit.	16
Figure 8: PLC Ladder Program States 3, 4, and 5 – Pilot Lit, Burner Off.	17
Figure 9: PLC Ladder Program States 4, 5, and 6 -- Burner Lit.....	18
Figure 10: Network Configuration.	19
Figure 11: Burner State Diagram	21
Figure 12: Refinery HMI -- Preparing to Light Burner Automatically Upon Power On.	22
Figure 13: RefineryBurnerMonitor Definitions – HMI Pilot Lit.....	23
Figure 14: Refinery Burner HMI -- Burner Lit.	23
Figure 15: Refinery Burner View -- Gas Dispersing.	24
Figure 16: Refinery / Reboiler Demo HMI.	25
Figure 17: Refinery Reboiler HMI – Design Mode.....	25
Figure 18: System Component Data Flow.	27
Figure 19: Umbra Refinery Burner System Data Flow.	29
Figure 20: Refinery Cooking Tank Data Flow Diagram.	31
Figure 21: Refinery RTUs Data Flow.	32
Figure 22: Abstract Burner.....	35
Figure 23: Burner Visual.....	36
Figure 24: Refinery Burner View 1.	36
Figure 25: Refinery Burner View 2.	37
Figure 26: Real-World Refinery.	38
Figure 27: Distilling Tower at Hydrocracking unit DSCO3097.	39
Figure 28: Refinery Burner Simulation View 3.....	40
Figure 29: Refinery Burner Simulation View 4.....	40
Figure 30: Refinery Burner Simulation Headers View 1.....	41
Figure 31: Refinery Burner Simulation Headers View 2.....	42
Figure 32: Refinery Burner Simulation View 5.....	43
Figure 33: Refinery Burner Simulation View 6.....	43
Figure 34: Refinery Burner Simulation View 7.....	44
Figure 35: Refinery Burner Simulation View 8.....	44
Figure 36: Buggy Ladder Logic - View 1.....	49
Figure 37: Buggy Ladder Logic - View 2.....	50
Figure 38: Buggy Ladder Logic - View 3.....	51
Figure 39: Buggy Ladder Logic - View 4.....	52
Figure 40: Refinery Burner Simulation View 9.....	53
Figure 41: Refinery Burner Simulation View 10.....	53
Figure 42: C301 RTU Values.	55
Figure 43: C301 HMI.....	55
Figure 44: C301 RTU Analog Point Specification 1.	56
Figure 45: C301 RTU Analog Point Specification 2.	56
Figure 46: C301 RTU Analog Point Specification 3.	57
Figure 47: C301 RTU Analog Point Specification 4.	57
Figure 48: C301 RTU Analog Point Modbus Specification.	58
Figure 49: C301 RTU Digital Point Specification 1.	58
Figure 50: C301 RTU Digital Point Specification 2.	59
Figure 51: C301 RTU Digital Point Modbus Specification.	59
Figure 52: C301 RTU Digital Point Control Specification.	60
Figure 53: C301 RTU Channel Definition.	61
Figure 54: C301 RTU Channel Primary Port.....	61

Figure 55: C301 RTU Holding Registers.	62
Figure 56: C301 RTU Input Registers.....	62
Figure 57: C301 RTU Output Coils.	63
Figure 58: C301 RTU Scanner Output Coils.	63
Figure 59: Heater RTU Interface Values.....	64
Figure 60: Heater RTU HMI.....	64
Figure 61: Heater RTU Channel Definition.	65
Figure 62: Heater RTU Channel Primary Port.	65
Figure 63: Heater RTU Output Coils Primary Port.	66
Figure 64: Heater RTU Holding Registers.	66
Figure 65: Heater RTU Input Registers.	67
Figure 66: Heater RTU Output Coils.....	67
Figure 67: Heater RTU Scanner Input Registers.....	68
Figure 68: Supply RTU Values.....	68
Figure 69: Supply RTU Definition.....	69
Figure 70: Supply RTU Channel Definition.....	69
Figure 71: Supply RTU Holding Registers Scanner Definition.....	70
Figure 72: Supply RTU Holding Registers Primary Port.....	70
Figure 73: Supply RTU Scanner Input Registers.	71
Figure 74: Supply RTU Output Coils Scanner.....	71
Figure 75: Burner Gas On Button Definition.	72
Figure 76: Burner State Definition.....	72
Figure 77: Burner RTU Channel Definition.....	73
Figure 78: Burner Input Discretes Scanner Top.	73
Figure 79: Burner Input Discretes Scanner Bottom.	74
Figure 80: Burner Input Discretes Primary Port.	74
Figure 81: Burner Output Coils Scanner.	75

EXECUTIVE SUMMARY

A simulation model of a refinery burner achieves a high fidelity by incorporating real-world Supervisory Control and Data Acquisition (SCADA) software and a Programmable Logic Controller (PLC) to drive the simulation. The refinery burner model and simulation is one of the capabilities included within Sandia National Laboratories' Virtual Control System Environment (VCSE). This model provides a platform to address key security issues of interest to the Department of Energy and the Department of Homeland Security.

To disseminate this technology and approach more widely, this report documents the project in terms of its architectural design, enabling others to use and expand the model. Further, the initial illustrative example of a refinery burner merged with a previous abstract refinery model demonstrates the high fidelity achievable with this approach and further demonstrates the feasibility of this approach both economically, and technically, in advancing key interests in homeland security threat reduction related to activities and processes using the Internet. The model was developed for a broader community and not for specific stakeholders. The use of control systems simulations such as this are extremely important since the simulation capability permits a variety of tests to be performed that could not otherwise be conducted due to cost and safety issues.

Architectural views of the refinery burner simulation present the key information necessary for an in-depth understanding of the model. These views, developed after the fact, enhance understanding of this model by identifying specifics related to the various aspects incorporated within the model, and identifying where improvements may be made in the future.

A number of views, or aspects, are presented in this report because they are necessary to fully understand the demonstration of this simulation. These views are representative of the multiple aspects of model design that must be considered for model representation of the controls for physical systems. The views included are:

- a system component view,
- a network view,
- an event view,
- a data flow view,
- a mechanical view,
- a physics view,
- a system safety view,
- a ladder logic view,
- an HMI view,
- a software component view, and finally,
- a components view.

This tool provides an excellent approach to control testing costs in the SCADA environment and is an excellent training tool for researchers and developers to understand the impact of specific control exchanges on expensive, critical, physical systems.

(Intentionally left blank)

INTRODUCTION

Activities and processes utilizing computer networks accessed through the Internet have an exposure to cyber threats in addition to the standard threats inherent in the same standalone physical systems. Security support provided to the Oil and Gas Industry Sector requires certain of these activities to be analyzed for threat awareness, in order to minimize and eliminate as many threats as possible. In addition, problems occurring with certain processes may have a high cost related to several measures such as economics, potential loss of life, scheduling impacts, and service disruptions.

In particular, we focus on Supervisory Control and Data Acquisition (SCADA) systems and other similar cyber-physical systems. Such systems use local area nets to send control information, status information, and other necessary exchanges between the SCADA systems and the equipment or infrastructures that they control. The SCADA systems, in turn, often exchange high-level data with corporate databases that, in turn, are often indirectly connected to the Internet. These indirect paths between the Internet and infrastructures are of particular concern.

Testing the accuracy of these exchanges and what happens to the physical system when presented with errors in the control directives can be prohibitively expensive and potentially quite dangerous to associated personnel in regards to their safety. Our approach allows us to examine items of concern through a high fidelity simulation that uses the real physical components to drive the simulation, while we model the extremely expensive physical systems with enough fidelity to make whatever assessments we are focused on during threat analysis. Key to the success of this model development and presented in this report are the concepts of the multiple aspects of model design and development that must be considered to capture the necessary model representation fidelity of the physical systems.

In this document, we focus on specifics related to aspects of the simulation's resulting architectural designs; understanding the utility and importance of this approach has been documented more extensively in another of our reports. (See reference 7: "Modeling and Simulation for Cyber-Physical System Security Research, Development and Applications.")

This document presents the Refinery Burner Simulation Design Architecture. This system represents a typical oil refinery and burner with an actual Programmable Logic Controller (PLC) directing the refinery processes. The refinery burner model and simulation is one of the capabilities included within Sandia National Laboratories' Virtual Control System Environment (VCSE). [7]

The architectural design views include:

- a system component view,
- a network view,
- an event view,
- a data flow view,
- a mechanical view,
- a physics view,
- a system safety view,
- a ladder logic view,
- an HMI view,
- a software component view, and finally,
- a components view.

These views document an example system developed to show proof of concept of our approach. The initial example focused on a simple event state process, and the subsequent design evolved from that example. In this document, we focus on specifics related to aspects of the simulation's resulting architectural design. The

utility and importance of this approach is necessary to determine the components that must be considered in modeling controls for complex, physical systems.

The first section of this document provides a high level overview of the simulation we developed, while the subsequent sections address the various architectural aspects of the overall model design. Finally, we present our conclusions following the presentation of the various architectural views. We begin with the simulation overview.

SIMULATION OVERVIEW

The refinery burner simulation combines two different models. The first models a simple refinery process that takes a fluid from a product tank, processes it by heating it to a specified temperature, siphons off the resulting product, and stores it in a product tank. The second models a real burner that an actual PLC controller turns on and off to maintain the fluid in the reboiler C301 tank within a specified range dictated by the Heater Remote Terminal Unit (RTU). In their integrated form, the PLC-controlled heater responds to control commands in the processing tank to cycle the burner's flame. A commercial SCADA system interacts with and depicts the current states of the simulation and the PLC.

Figure 1: Refinery Burner System Component / Data Flow Overview.

Figure 1 depicts the simulation's architecture. The simulation runs within three separate operating systems that interact via an Ethernet network. Umbra is a modeling & simulation framework utilized with the VCSE and is used extensively in this project. The SCADA HMIs run within a VMware virtual machine; the PLC Controller is an industrial control computer, and the Umbra model executes on a third computer. The SCADA software includes three HMI (Human Machine Interfaces) screens for interacting with different parts of the plant model. The first is an abstraction that is used to interact with the PLC. The second is a plan that allows the operator to interact with high-level plant functions. The third is a more detailed description of the reboiler that allows the operator to control specific aspects of the reboiler circuit. The SCADA communicates with the Umbra model and the PLC using Modbus, a communications protocol that is often used in real industrial control systems. The Umbra model is configured to emulate three different RTUs (Remote Terminal

Units) that have three separate network addresses. The PLC also interacts with Umbra through a Modbus interface. (This later interface was chosen for convenience.)

The system's architecture and tools can be readily extended to represent many simulated RTUs and incorporate many PLCs. It has also been tested with emulated PLCs that use the same operating system as the real PLCs and otherwise act just like real PLCs with the main difference being that no industrial hardware is used to contain the operating systems.

A commercial PLC from SoftPLC interfaces with the simulation via the network to drive the simulation's burner model. This PLC also interfaces with the SCADA system to allow operators to control high-level functions in the burner system through the SCADA HMI. Finally, it also interfaces with a developer host computer used both to program the PLC and to monitor its internal state while developing and demonstrating the software. The SoftPLC is programmed in a ladder logic language.

The demonstration system includes two ladder logic programs. The first controls the burner in a safe and efficient way. A second includes a software flaw that causes the burner system to fail catastrophically. We discuss the correct and flawed versions of the ladder logic program in the section expounding on the ladder logic view.

The umbra model merges two different models: a refinery process model and a burner model.

The refinery model is representative of a one of the more basic operations in oil refining. The example is based upon training materials used to train plant operators to control refinery processes. (A Sandia industry partner recommended this process to Sandia for a first proof-of-principal application of this VCSE model.) The system being modeled pumps oil from a storage tank to a processing tank. Oil from the processing tank is pumped through a heater circuit back to the processing tank. Processed oil can then be re-directed to a product tank.

The refinery model is based upon first-order physics equations and implemented using standard modular modeling techniques developed for Umbra. Fluid accumulation within the tanks and heater is computed by integrating flow rates over time. Temperature is computed by executing heat balance equations. The heated element includes a damage model, which is activated when the temperature within the heated element exceeds a given threshold. The model includes a 3D graphical representation that includes a 3D model of a refinery and animation models to represent fluid volumes in tanks and accident scenarios. The model also displays key process parameters, including temperatures and fluid volumes, above the model of each relevant piece of equipment.

The burner model is representative of typical industrial burners. The physical model is based upon a burner sold by Lesman. [3] The PLC program and its architectural location in the system is representative of safety interlock systems found in safety-sensitive burners used at oil refineries. A Sandia industry partner had highlighted this as a best-practice architecture used throughout the refining industry. Sandia choose to model it as an example of how high-fidelity VCSE models could be used to verify control software functionality and assurance in safety-critical applications.

The burner model is also based upon first-order physics equations and implemented using standard modular modeling techniques developed for Umbra. The model includes physics models of each valve and pipe in the burner. Burner ignition is modeled by modeling the accumulation, consumption and diffusion of gas in the region around the ignition sources. The model allows gas to enter the burning region through either a pilot light or the main burner gas plumbing. Prior to ignition, the model represents how gas can accumulate in the region and diffuses slowly. Post-ignition, it represents how the flame consumes the gas and produces heat.

The model represents how in normal operation, the pilot gas is introduced then an igniter is activated. This starts a pilot flame that, in turn, heats a temperature sensor. The burner gas is then introduced which lights and produces heat for the refinery process. It also represents how in abnormal operation, the igniter is not activated before gas is introduced to the burner element and gas accumulates in the region around the burner. (The gas dissipates slowly if the gas supply is turned off.) The model indicates that an explosion will occur if the igniter is activated when an over-abundance of gas is in the region.

The visual model of the burner presents an abstract representation of the components described as generally used by an industrial burner. We documented our assumptions regarding the functioning of the various components comprising the burner, in order to facilitate model validation at the appropriate time. The PLC controller turns the burner on and off, depending upon whether the fluid being heated exceeds the temperature ranges specified by the user through the HMIs. If the fluid becomes too hot, the burner turns off; if the fluid becomes too cold, the burner turns on and the flame reappears in the visual graphic. The "On – Off" cycling of the burner occurs approximately two minutes into the simulation.

SYSTEM COMPONENT VIEW

As noted previously, three system components comprise this simulation model: the HMI refinery controller interface, the PLC Controller, and the Umbra model. The HMI views: 1) monitor the refinery and PLC states, and 2) allow the user to interact with the refinery to set desired input values and limits. The PLC Controller drives the simulation by turning various gates and switches on and off, as specified by an internal ladder logic program according to the current system state and the occurrence of system events, such as new inputs or outputs. The Umbra model simulates the physical systems of an actual refinery and burner. The following subsections provide a high level review of each of these components before proceeding to discussion of the various architectural views of the model design, as stated in the introduction.

Umbra Simulation Model

The Umbra Simulation Model runs on a separate computer. The Umbra simulation receives input from the PLC Controller, updates the visual graphic models of the refinery and refinery burner based upon the current values of the gates and switches output to Umbra from the PLC Controller. Changes determined by the Umbra processing component result in values being passed back to the PLC Controller, which results in possible changes in the settings for the various gates and switches controlled and monitored by the PLC controller. The Umbra Simulation Model does not interact directly with the HMI screens, only indirectly through the PLC Controller. Figure 2 illustrates the visual graphic model of the refinery and burner. The small, neon green object in the middle of the picture represents the "blower" for the refinery burner. We present more views in later sections.

HMI VMWare

The HMI represents the refinery control panel that portrays the current status of the refinery components in real time. This simulation used commercial software packages to develop multiple HMI's for the simulation. We used WonderWare [16] for the initial design, and then transitioned to ClearScada. One HMI view utilizes icons and images to portray the reboiler system interface in a format familiar to the refinery worker managing a typical refinery control panel (see Figure 3). Another, more abstract, HMI view portrays the simple event states associated with the refinery burner used by the reboiler system in maintaining the specified temperature for the material being processed (see Figure 4). Finally, a third HMI view provides greater detail in a schematic format for the reboiler process (see Figure 5). We discuss particulars for each of

these HMI screens/views in greater detail in the following sections, and specifically within the section devoted to the HMI views.

Figure 2: Umbra Visualization of Refinery and Burner.

Figure 3: Reboiler System HMI.

Figure 4: Refiner Burner HMI – Model States / Actions

Figure 5: HMI Reboiler Design 2 – Expanded Details; Running.

PLC-SCADA Controller

Real hardware in the form of an actual PLC controller is used to drive the refinery / burner simulation, resulting in a high fidelity simulation that allows real world control programs to be tested and evaluated in a laboratory setting. Consequently, ladder logic programs must be written to run the controllers. Figures 6 through 9 illustrate the various logic rungs within the ladder logic program at various states during the simulation. These figures are independent of each other in the sense that each figure depicts the system at different times. In other words, these figures and states are not contiguous in time relative to the simulation execution. However, they do show which gates and switches are on and off during particular situations that occur when the simulation runs. Further, these four figures collectively depict each rung within the ladder logic program. The ladder logic view section, which appears much later in this document, discusses the details for these figures, this current section simply presents the various system components at a high level and we proceed with the next section, which outlines the network view for our simulation.

Figure 6: "PLC Ladder Logic Program State 0 – Waiting for "Power on" to be Pressed", shows the first three rungs of the ladder logic program. All gates are off, depicting that the burner is turned off and will not be turned "On", until the associated HMI "Power On" button is pressed. Figure 7: PLC Ladder Program States 1 and 2 – Burner Lit shows the next two program rungs. Their current status, as depicted, indicates that the pilot light has been lit. Figure 8: PLC Ladder Program States 3, 4, and 5 – Pilot Lit, Burner Off shows the next three rungs; here, the pilot light has been lit, but the burner has not. Figure 9: PLC Ladder Program States 4, 5, and 6 -- Burner Lit shows the last few rungs; the burner is now lit. These figures come from screenshots of a tool, (SoftPLC TOPDOC) that allows one to view the executing PLC program.

Figure 6: PLC Ladder Logic Program State 0 – Waiting for "Power On" to be Pressed.

State 1: READY; This state is entered once the power has been turned on and waits for the START button to be pressed to initiate lighting the

B10/3: Start Pressed (Button);

B20/2 Turn on the Pilot Gas Cock;

B20/3: Turn on the Pilot Solenoid Valve;

B20/4: Turn on the Solid-State Ignition Transformer, unless a flame is already present;

T4:0: Timer to light Pilot light; T4:1: Timer to allow slight delay to initiate reset if needed.

State 2: Timer On; T4:0/TT: Timer to light Pilot light;

B21/4: Flame Detector Sensor; B20/4: SolidStateignitionTransformer

B20/6: Thermocoupler Leadwire A (detects heat of pilot light);

B20/7: Thermocoupler Leadwire B (detects temp of main burner);

B20/14: pilotLightHot enough to light the burner. (deleted at present, but defined in umbra interface) could be used for

Figure 7: PLC Ladder Program States 1 and 2 – Burner Lit.

State 3: Reset Clear State, used to allow accumulated gas to dissipate before continuing to the ready state to retry lighting the pilot light.

Turns off: All gas valves in previous rungs when B11/3 goes on: B20/2 (Pilot Gas Cock);
B20/5 (Gas Control Valve); and
B20/8 (Main Gas Valve Operator) ...etc, then

T4:2: Timer waits 40 seconds for the gas to dissipate

State 4 (B11/4 is on) : Pilot Lit: Waiting for the flame to become hot enough to light the pilot light. Turns on B11/5 (State 5: Burner Lit).

B21/5: Flame Hot Enough to Light Burner (Sensor);

B10/4: Light Burner Button Pressed (Button).

B20/5: Turn on the gasControlValve for main pipeline

B20/8: Turn on the burnerValvecontroller for main pipeline (final valve for gas to flow)

Figure 8: PLC Ladder Program States 3, 4, and 5 – Pilot Lit, Burner Off.

State 4 (B11/4 is on) : Pilot Lit: Waiting for the flame to become hot enough to light the pilot light. Turns on B11/5 (State 5: Burner Lit).

B21/5: Flame Hot Enough to Light Burner (Sensor);

B10/4: Light Burner Button Pressed (Button).

B20/5: Turn on the gasControlValve for main pipeline

B20/8: Turn on the burnerValvecontroller for main pipeline (final valve for gas to flow)

State 6 (B11/6 Disperse is on): Shutdown the system: turn off the burner and disperse the gas. Wait for power turn off, or an error condition (

!B10/0: Power turned off.

B21/7: Flame is too hot (sensor).

B11/6: Signal that you are in the Disperse state.

B20/13: Signal that fatalErrors has occurred, occurs when Umbra turns on B21/7 thermocoupleSensorBTemp.

Reset all Timers.

Figure 9: PLC Ladder Program States 4, 5, and 6 -- Burner Lit.

Figure 10: Network Configuration.

NETWORK VIEW

We configured the simulation components on a standalone network separate from our corporate network to facilitate presenting the demonstration at various locations. This provided us with an isolated sandbox for our tests. Figure 10 specifies the exact configuration used by the refinery burner simulation.

Establishing Desired Network

We programmed and executed a short script that established the desired network addresses:

```
netsh interface ip set address name="Local Area Connection" static addr=192.0.1.50
mask=255.255.255.0 gateway=192.0.1.50 1
netsh interface ip add address name="Local Area Connection" 192.0.1.100 255.255.255.0
netsh interface ip add address name="Local Area Connection" 192.0.1.101 255.255.255.0
netsh interface ip add address name="Local Area Connection" 192.0.1.102 255.255.255.0
ipconfig
pause
```

When we concluded our work with the simulation, we ran another .bat file to restore our network:

```
netsh interface ip set address name="Local Area Connection" dhcp
ipconfig
pause
```

IP Addresses

The main host computer ran the SOFTPLC TOPDOC program, which allows one to monitor the ladder program being executed on the PLC controller dynamically. The host had an IP address of 192.0.1.50. The Umbra model also ran on the host computer at 192.0.1.50, but the configuration within the Umbra model also specified three additional RTUs at 192.0.1.100, 192.0.1.101, and 192.0.1.102, for the Supply RTU, the C301 RTU, and the Heater RTU used by the refinery model.

The HMI views ran in a VMWare set to 192.0.1.107, and the PLC Controller had an IP address of 192.0.1.110. The Burner model ClearScada HMI running in VMWare at 192.0.1.107 provided output from the Abstract HMI to the PLC controller at 192.0.1.110. The SoftPLC Controller at 192.0.1.110 receives input from the VMWare running the HMI at 192.0.1.107 and from Umbra running at 192.0.1.50; then processes the input and outputs via the same IPs to cycle through the simulation. The simulation essentially processes the input, then generates the output, then repeats the cycle.

The Umbra model has four RTUs, one for each of the major components within the model. These four IPs are used to send and receive status and control information for each of those main components: the SupplyRTU, the C301RTU, the HeaterRTU, and the TrustAnchorRTU, which is the RTU used by the burner.

Network Protocol

This simulation used the MODBUS network protocol to exchange data as required between the IP addresses. The TOPDOC tool from SoftPLC allowed us to specify the network configuration, including the drivers for the client and server processes. The ClearSCADA tool, also allowed the necessary specifications to configure the network as designed.

EVENT VIEW

The event view depicts the initial state diagram developed for the refinery flame burner. We defined possible states for the burner; then, we identified what events would cause a transition from any one of the states, to any of the other states. Figure 11 shows the initial refinery flame burner state diagram. There were five states initially, and then the diagram was reworked to streamline the events. Figure 12 shows the final version and actual Abstract HMI design used with the refinery burner simulation demonstration.

Figure 11: Burner State Diagram

Figure 12: Refinery HMI -- Preparing to Light Burner Automatically Upon Power On.

HMI – Burner States

The system is initially in the "Off" state. One must turn the gas on to advance to the "Ready" state. Once there, a timer begins and the pilot light tries to light. If the pilot lights, state 4 is achieved. Pressing a button to light the burner causes advancement to state 5. If the burner pilot light does not light within a specified time, the burner system cycles to the "Clear" state to allow the gas that has built up to dissipate. Once a set amount of time has passed, the burner system again enters the "Ready" state and the timer countdown for the pilot light to light begins again. The system cycles amongst these three states until the pilot is lit. Once the pilot light is lit, the burner system is ready to turn on the main burner. Pressing the "Light Burner" button will cause the main burner to turn on.

Three buttons are used in this abstract HMI view, and each button has two functions. One function occurs when the button is pressed and is up, while another function occurs when the button is pressed and is down. Whichever state, up or down, is current when pressed, the button will then transition to the other state. If a button is up when pressed, it will enter the down state; if it is down when pressed, it will enter the up state. The power on button located under the "Off" state, is used to both "Power On" and "Shutdown" the burner system. When the main burner is lit and the system is shutdown, the burner system enters the "Disperse" state, which enforces a delay before the burner can be relit in order to allow any gas that has built up to disperse as a safety measure. Figure 13, Figure 14, and Figure 15 show what the HMI looks like in these various states. The state indicators are either red or green. Green indicators indicate that a state has

been reached; a red indicator means the state has not been reached. Reset and start comprise one button; and a single button is used to turn the gas either "On" or "Off".

Figure 13: RefineryBurnerMonitor Definitions – HMI Pilot Lit.

Figure 14: Refinery Burner HMI -- Burner Lit.

Figure 15: Refinery Burner View -- Gas Dispersing.

HMI – Reboiler Info

The reboiler HMI view (Figure 16) running in VMWare, provides a more realistic control panel for viewing and assessing the status of the refinery. The Supply Tank, the C301 Tank, the Heater, and even the Product tank are clearly identifiable. Pipes show how the fluid flows amongst these components. The status, as to whether the refinery processing is running or not, is clear; and the associated temperatures of the fluid at key points in the refinery process are tracked and displayed. A switch allows the user, or refinery controller, to switch between manual and automatic control. Feedback mechanisms are clearly indicated through the pipe structures. This HMI utilizes IP addresses 192.0.1.100, 192.0.1.101, and 192.0.1.102 to exchange data with the Umbra simulation and visualization system component running on 192.0.1.50. If the fluid temperature drops below the minimum temperature, or exceeds the maximum temperature, the umbra component sends information to the PLC controller on IP address 192.0.1.110 to either turn the main burner on to heat up the fluid, or to turn it off to allow the fluid to cool off for a while.

Figure 16: Refinery / Reboiler Demo HMI.

Figure 17: Refinery Reboiler HMI – Design Mode.

HMI – Reboiler Detailed Info

The detailed reboiler HMI screen entitled "ClearSCADA Reboiler HMI" (Figure 17) provides an even more realistic representation of a typical control panel one might see at a real-world refinery. This HMI view can be used to turn valves and switches on and off in the Umbra refinery simulation and to track the current status of those same valves, as well as the temperature of the fluid being processed.

DATA FLOW VIEW

The Refinery Burner Simulation incorporates strong data abstraction principles. Each component has an invariable data flow boundary. This approach mirrors that of the Umbra environment. The Umbra simulation process uses data abstraction as a keystone principle. Once a simulation has been initialized, the simulation just loops over and over. During each loop, the system passes data between the components in a prescribed order: the components complete their calculations using their current/updated input values, then they update their own output values (including any feedback loops), and then the simulation cycle is ready to repeat. The order of updates follows the specified dataflow sequences. Therefore, understanding how the data flows between each of the components provides a strong foundation for understanding the overall architectural design for this simulation. In addition, the user needs to understand that numerous aspects of this simulation are non-deterministic and they cannot expect statements to always be executed in the order they are presented.

We consider multiple data flow aspects: 1) the data flow between the three main system components including the ClearSCADA HMIs running on a VMWare, the PLC controller running on a SoftPLC controller, and the graphic visualization of the physical refinery and burner being generated by the Umbra simulation code running on the host computer; and 2) the data flow between software components in the Umbra simulation code modeling the burner system and the refinery system. We consider several viewpoints for each of those aspects: first, we review the overall flow, and then explain how to read the data flow schematics developed for this simulation, and finally we trace a single data flow through the system.

Main System Components

The three main system components, the HMIs, the PLC Controller, and the Umbra visual simulation, exchange data via the MODBUS TCP network protocol. Figure 18 identifies the bit values passed between these components. Bit values pass from the Abstract Burner State HMI to the PLC controller and map to the B10 gates/switches within the PLC Controller. Data values within the PLC Controller are, in general, bit values that indicate whether a gate is "On" or "Off".

The only way to reference data values within ladder logic programs is to use a reference such as "B10/0". "B10/0", means bit zero starting at Block 0. Block B10 is used within our PLC ladder logic program to contain the bit values coming as input from the abstract event HMI. Block B11 is used to transfer output values from the PLC Controller to the HMI.

Bits in Block 21 are used to receive the bit values coming as input from the Umbra simulation and block 20 bits are used to output values from the PLC Controller to the Umbra simulation to direct the simulation as to which gates are "On" or "Off". The Umbra simulation uses those bit values to dynamically change the physical system visualization, such as turning the flame "On" or "Off".

From the data in Figure 18, one may determine that the functions to either "Power on" or "Shutdown" the refinery burner are merged into a single button, whose value is represented by the bit B10/0. The PLC

Refinery Burner Documentation

HMI			PLC					Umbra
WonderWare	State Diagram Name		PLC	PLC	PLC	PLC	PLC	trustAnchorRTU trustAnchorRTU
(output)	HMI (output)	HMI (output)	(input)	PLC (input)	PLC (input)	PLC (input)	PLC (input)	(output) Umbra (output) Umbra (output)
100001	Power On Button	===>	B10/0	B21/0	B21/0	B21/0	B21/0	<=== Z0 (offset filler)
100002	Shutdown Button	==>	B10/1 (merged with B10/0)	B21/1	B21/1	B21/1	B21/1	<=== Z1 (offset filler)
100003	Reset Button	==>	B10/2 (merged with B10/3)	B21/2	B21/2	B21/2	B21/2	<=== Z2 (offset filler)
100004	Start Button	===>	B10/3	B21/3	B21/3	B21/3	B21/3	<=== burnerNotStandAlone (Model Switch)
100005	Light Burner Button	===>	B10/4	B21/4	B21/4	B21/4	B21/4	<=== flameDetectorSensor
100006	Turn Off Button	==>	B10/5 (merged with B10/4)	B21/5	B21/5	B21/5	B21/5	<=== thermocoupleSensorAPIlot
100007	Gas High Button	===>	B10/6	B21/6	B21/6	B21/6	B21/6	<=== heatSwitchOnOff (Input from Reboiler)
100008	Gas Medium Button	==>	B10/7 (expansion)	B21/7	B21/7	B21/7	B21/7	<=== thermocoupleSensorBTemp
100009	Gas Low Button	==>	B10/8 (expansion)	B21/8	B21/8	B21/8	B21/8	<=== gasPressureTooHigh
				B21/9	B21/9	B21/9	B21/9	<=== gasPressureTooLow (in place, ignore)
(input)	HMI (input)	HMI (input)	(output)	PLC (output)	PLC (output)	PLC (output)	PLC (output)	(input) Umbra (input) Umbra (input)
000001	State 0	Power On	<=== B11/0	B20/0	B20/0	B20/0	B20/0	gasPressureRegulator (security function)
000002	State 1	Ready	<=== B11/1	B20/1	B20/1	B20/1	B20/1	pilotPressureRegulator (security function)
000003	State 2	Timer	<=== B11/2	B20/2	B20/2	B20/2	B20/2	pilotGasCock (under sink for pilot)
000004	State 3	Clear (for reset)	<=== B11/3	B20/3	B20/3	B20/3	B20/3	pilotSolenoidValve (faucet for pilot)
000005	State 4	Pilot Lit	<=== B11/4	B20/4	B20/4	B20/4	B20/4	solidStateIgnitionTransformer (ignites pilot)
000006	State 5	Burner Lit	<=== B11/5	B20/5	B20/5	B20/5	B20/5	gasControlValve (under sink for main)
000007	State 6	Disperse	<=== B11/6	B20/6	B20/6	B20/6	B20/6	thermocoupleLeadWireA (pilot hot enough)
				B20/7	B20/7	B20/7	B20/7	thermocoupleLeadWireB (flame too hot)
			(B11/7 end of Shutdown)	B20/8	B20/8	B20/8	B20/8	burnerValveController (faucet for main)
			(B11/8 end of Reset)	B20/9	B20/9	B20/9	B20/9	mainGasValveOperator (gas into house)
				B20/10	B20/10	B20/10	B20/10	ventValve (always open)
				B20/11	B20/11	B20/11	B20/11	lowPressureInterlock (mechanical?)
				B20/12	B20/12	B20/12	B20/12	highPressureInterlock (mechanical?)
				B20/13	B20/13	B20/13	B20/13	fatalErrors (signal to Umbra if needed)
				B20/14	B20/14	B20/14	B20/14	pilotLightHot (DELETED, use for another need)
				B20/15	B20/15	B20/15	B20/15	gasSetHigh (signal to Umbra if needed)

(Transported to ClearScada HMI also)

Figure 18: System Component Data Flow.

repeatedly cycles through the ladder logic program. During each cycle, the system rechecks the values of all of the input and output coils—input coils contain the B10 bits and output coils contain the B11 bits, according to the way we configured and designed this simulation. If B10/0 has a value of “1”, the sequence to turn the system “On” initiates; if it has a value of “0”, the shutdown sequence begins. The start and reset button are merged together in a similar fashion, and the button to light the burner has the alternate function of turning the main burner flame off.

Accordingly, B10/1, B10/2, B10/5, B10/7 and B10/8, although initially designed to have specific functions, currently are available for expansion. This compression occurred in order to limit the input to the PLC controller to 8 bits or fewer from the main components. The HMI transmits 4 bit values, and the Umbra Burner RTU (trust Anchor) transmits back 3 bits, leaving one bit for expansion. The yellow highlighting identifies the actual bits being used during one validation project. Because the B10s are input coils, the PLC controller cannot change any B10 bit value, it can only read or determine the current B10 bit values. The merged simulation of the refinery and burner together also uses B21/3 and B21/6.

The B11 bit values have a value of 0 or 1, depending upon whether the associated gates within the PLC ladder logic program are “On” (value of 1) or “Off” (value of 0). These output gate values are “On”, if current is able to flow through the ladder logic program rungs to that point in the program. “AND” and “OR” gates are used within the ladder logic program to direct, and/or block, the current flow as needed to achieve the desired logical functioning of the ladder logic program.

The HMI uses the B11 bit values to determine whether to set the indicator lights located in the middle of each oval event state (refer back to Figure 15) to red or green. A green indicator indicates a current state has been reached, or is enabled; a red state means the state has not been reached, or has been disabled for the moment. The B11 bits are “read only input” to the HMI view, whereas the PLC controller sets the values of the B11 bits. “Read only input” means the HMI cannot change these values -- it can only read or determine their values. Therefore, the B11 bits are input to the HMI from the PLC controller.

Based on the directives from the HMI view, and the coded logic, the PLC controller sets the appropriate bits for the B20s to be output to the Umbra simulation code. The Umbra simulation code reads these values as input and processes the visualization/simulation accordingly. The Umbra code cannot change the B20 values, it can only read them. As Umbra processes its’ internal loop, it will in turn set the appropriate values of the variables linked as output to the B21 bit values read by the PLC controller. The PLC controller can only read those values. Any change in those values by Umbra precipitates changes in the switches controlled by the PLC and the processing loop continues.

The reboiler HMI and the detailed Reboiler HMI views pass additional bit values through similar input and output coils that are exchanged with the Umbra refinery simulation. Those bits do not pass through the PLC Controller. These HMIs and the refinery were initially designed as a separate project, and then were merged with the burner system. A close examination of the ClearScada configuration and the Umbra software will identify the functioning and meaning of each bit value. We will leave that as an exercise for the developer with access to this simulation who wishes to either expand this simulation or delve more deeply into the nuts and bolts of the coding.

Burner System

As indicated above, a similar data abstraction, or encapsulation, occurs within the Umbra simulation code. The names of the B20 and B21 bit values identified in Figure 18 are the same names used within the Umbra code. Figure 19 illustrates the data flow throughout the burner system that is embedded within the larger refinery simulation. (You may need a magnifying glass to read this figure, we do. However, this figure clearly shows the data encapsulation and overall data flow concepts.)

Figure 19: Umbra Refinery Burner System Data Flow.

The orange rectangles that extend outside of the main greenish tinted box are the data values coming as input into the burner system code from the PLC controller. These bindings have the same names as those in Figure 18. Also, the associated B20 bit value is marked for each variable. The orange rectangles to the far right are the output values that progress back as input into the PLC controller. Again, the names are the same and the B21 bit values are noted.

The blue rectangles are variables coming as input into the burner system from the encompassing refinery system. These variables are set by, and are outputs from, other software components comprising the refinery simulation. The blue rectangles to the far right are values that feed as output from the burner system to become input into other components in the refinery code.

Each grey box with a green "lid" is a software component within the burner system. The green lid contains text specifying the type of the component (we use six component types within the Umbra Refinery Burner: ANDgates, ORgates, Switches, StorageTanks, Burners, and one Manifold); and a text entry specifying the name for that particular instantiation of the stated component type. The bright yellow rectangles outlined in grey to the left of each grey component specify the required input values for the component execution. Grey lines indicate the sources of these required input values.

The lighter yellow rectangles outlined in grey on the right edges of the grey components indicate the output values. These values may be bit values, string values, integer values, real values, *etc.* The output data type may be indicated in the text, or the specific name for that output value may be listed to avoid naming conflicts when there are multiple outputs of the same data types. Solid grey lines indicate where the data values flow, whether they proceed onwards to another component within the simulation, or are passed back to the PLC controller via the controller's input coils. Some data values flow forward to two locations, as indicated by the multiple lines proceeding forth from the variable specification.

The clear grey rectangles inside the grey components specify static data variables that are initialized within those components. The system must be recompiled to change those values.

The bright yellow rectangles outlined in orange on the left and right sides of the grey components are feedback variables. The leftmost variable receives its input value from subsequent iterations as the simulation progresses. The dotted grey lines indicate the data flow for these values in terms of origin and destination. Since these are feedbacks, the data value flows forward from the right most component linked by the dotted line back to the leftmost component, as indicated in the diagram.

Refinery System

The refinery system essentially has the same coding data flow structure as the refinery burner; however, the reboiler HMI view and the detailed reboiler HMI view also pass data values via input and output coils to the Umbra refinery system. The PLC controller does not process data from those two HMIs, except for one switch; a single input and output bit variable linking the refinery system and the burner system that indicates when the burner flame needs to be switched on or off based on the specified temperatures. The PLC controller controls the burner flame switch based on input through the B21 bit value provided, and controlled, by the Umbra code.

Figure 20 shows the general data flow diagram for a basic cookingTankController component that the Umbra refinery simulation uses. The variables in the pink filled rectangles were added to link the refinery burner simulation with the refinery simulation. "Heat On" exiting the Cooking Tank diagram feeds back to bit B21/6 in the PLC controller input coils and to the variable "heatSwitchOnOff" used as input to the refinery burner system, as seen in Figure 19. The cooking Tank diagram presents the abstract flow for this type of component; an actual implementation may incorporate additional variables, either inputs or outputs as needed (See Figure 21, which has two instances of cookingTankController -- each with slight variations.)

Figure 21: Refinery RTUs Data Flow.

We see several instances of `cookingTankController`, as we look at the data flow for a larger group of components used within the refinery system in Figure 21. Figure 21 shows the data flow interactions between the refinery components that link to the reboiler and detailed reboiler HMIs. Yellow "RTU" tags indicate the links to the input and output coils transmitted to the HMIs running on VMWare at 192.0.1.107.

We do not include the complete set of data flow diagrams for the Umbra refinery model in this document as they constantly change due to evolving project requirements, and thus, would be out of data by the time this document reaches publication. However, the included diagrams were accurate at one time. (See reference 6 for the most current documentation.)

Single Data Flow Trace

We start with the HMIs to trace a single data flow. Looking at the abstract HMI, pressing the "Power On" button will set the appropriate bit in the HMI output coil to "1" (and set the "power on" button in the down position, which changes its text from "Power On" to "Power Off" with a magenta background). The HMI output coils are sent to and read by the PLC controller. The PLC controller, after reading the coils from the HMI accordingly sets bit B10/0 to "1". Refer back to We start with the HMIs to trace a single data flow. Looking at the abstract HMI, pressing the "Power On" button will set the appropriate bit in the HMI output coil to "1" (and set the "power on" button in the down position, which changes its text from "Power On" to "Power Off" with a magenta background). The HMI output coils are sent to and read by the PLC controller. The PLC controller, after reading the coils from the HMI accordingly sets bit B10/0 to "1". Refer back to Figure 6 and examine the first rung in the PLC controller ladder logic program. The gate B10/0 will now turn "On". In turn, the switches B11/0 and B20/10 go "On" simultaneously, as well.

The next time Umbra completes a processing loop, it will receive the new B20 output coils from the PLC Controller. The Umbra software is configured to link the bit B20/10 to the `ventValve` variable in the simulation, thus the external variable labeled "ventValve (B20/10)" in the left hand side of the Umbra Refinery Burner Schematic switches to "1" from a value of "0". ("0" is the initialization value for these variables.) Nothing further happens at this time in regards to the burner schematic; but, the burner visualization graphic will change the color of the ventValve in the 3d visualization of the burner a lighter, more vibrant and luminescent shade of green -- making it appear as though the ventValve turned on, similar to a neon sign turning on and exuding a brighter color. In this case, the large "blower" (the large lime green object looking like the letter "P" on its back) also turns on in the visualization.

When the switch B11/0 goes "On" the bit value also changes to "1" within the B11 output coils. The next iteration of the processing loop within the HMI will result in those coils being read and processed, thus the bit value of "1" for B11/0 results in the red indicator in the "Off" state, the oval at the top of the Refinery Burner Controls Abstract HMI, lighting up and turning green.

Nothing happens further until the "Gas On" button to the upper right is pressed. This button may be left on the same way a person may leave the radio on in their car, and have it turn on the next time the car is started. If the "Gas On" button (B10/6) is already on, then the current in the ladder logic program will progress further and additional actions will happen. We will allow the reader to trace through the code as an exercise to determine what will happen next. However, you can see the definition of this bit as the first bit in the Abstract HMI output coils, if you refer back to Figure 13. Look at the database panel to the left of the image. You can see the definition under the `RefineryBurnerMonitor` folder. Examine the third rung in the ladder logic program (Figure 6) to determine what happens next with the data flow.

The data flow processing for the Reboiler and Detailed Reboiler HMI progresses in exactly the same way. The user may change specific values for several variables, such as the requested fill rate or the requested temperature, or the heat rate, *etc.* These values, in turn, are sent via the output coils using MODBUS TCP to the Umbra code. The bits/values in the output coils feed into the components (see Figure 21) and on the

next processing loop, Umbra makes whatever changes must occur as the data values input from the HMI change. The resulting output values from the various components then filter back to the HMI via a similar process. In addition, the user may flip switches on the HMIs. Images of switches are used to present a more real-world feel to the design. Once clicked, the switch will move down (if up) and up (if down). The appropriate data value indicating the position of the switch progresses via the HMI output coils to Umbra and the associated actions and visualization changes occur as programmed within Umbra.

Visualization Data Flow

The visualization works in a slightly different manner. The Tool Command Language (Tcl), a string-based control language, and Tk (a toolkit for programming graphical user interfaces) are utilized by Umbra. [17] Tcl/Tk code is used to handle the graphics for these models. Essentially a number of visual states are defined based on the values of specified variables. The state definition specifies what is present or not in the visualization and what it looks like. This gives a semblance of dynamic reaction and action within the visualization. The burner simulation uses the same variables passed as B20 and B21 input and output coils from the PLC controller. If one of those bits is "On" with a value of 1, the color of that item (or related visual objects) changes colors.

Headers over the key refinery components also display their changing values, such as the temperature over the heater and the C-301 component. The supply tank and product tank may show fluctuating levels of fluid depending upon how long the simulation has run and what parameters have been set by the HMI.

If certain conditions occur, explosions and fires may erupt.

MECHANICAL VIEW

From a mechanical perspective, we gathered as much data as we could find to produce as accurate a model as our resources permitted. This included visiting actual refineries, and gathering what information we could from websites, and querying staff members who we knew had prior refinery experience. We primarily concerned ourselves with mechanical operations that could be operated via a cyber component. However, some components displayed in the visualization had/have a strictly mechanical operation with no cyber connection. We briefly discuss the burner components, relevant refinery components, the controller, and the HMIs to identify mechanical aspects incorporated into this simulation. The following subsections address each of these elements in turn.

Burner

We found a web brochure with a diagram of a burner, and we used that design as the basis for our abstract burner. First we recreated the diagram; Figure 22 shows the results of the recreation. There were twenty items identified as individual components on the burner, as can be seen in the list at the top right of Figure 22. A number of the items appear to be duplicated in both the pilot pipe line, and the main burner pipe line. Items that we considered to be the same type were coded in the same color. Some assumptions were made regarding the functions of these components. Figure 23 shows the abstract visual model constructed representing the components used for a functioning burner as depicted by the research diagram used as the basis for the burner design. Figure 24: Refinery Burner View 1." shows a close-up of the abstract burner embedded within the refinery simulation.

Figure 22: Abstract Burner.

Figure 23: Burner Visual.

Figure 24: Refinery Burner View 1.

Figure 25: Refinery Burner View 2.

The PLC Controller turns on and off most of these components. Purely mechanical components that the controller cannot manipulate include: the manual reset, the two manual shut off cocks, and the two test cocks. The manual shut-offs only appear in certain versions of the simulation since they cannot be manipulated. We do not have a separate component for the Gas Pilot Burner Assembly; we just assume one is there, supporting the burner. A similar assumption is made with the primary control relay mounting base. The high limit safety shutdown is incorporated within the processing for the thermocouple sensors. We made the assumption that the remaining items can be controlled via cyber commands issued by the PLC Controller.

In Figure 25, we can see the lighted burner. We assume pipes link from the tank being heated, to the other relevant tanks. Those pipes were not added because we had limited resources when the burner was embedded into the refinery simulation. Similar assumptions were made in regards to any structures supporting the burner assembly.

Multiple documents developed by the authors further identify the process flow for igniting the burner. (See references.) However, understanding how the valves operate is the most important information required in understanding the overall burner processing. We assumed they operate similar to water valves in a house. A main valve controls water flow from the city to the house. Other valves found under the sinks control whether water may progress to the particular sink. Finally, valves present at the sinks determine whether hot or cold water will flow through the choice of valve, and when the water flows is determined when the valve is turned "On".

Controller

Many controllers will have the "switches" that they control actually implemented mechanically. Our simple model defines a number of bits as input and output and sets bit values to send to the other two components within the simulation. (Please refer back to Figure 18.)

HMI

Similarly, older SCADA HMIs used by the Refinery controllers may have mechanical implementations, but ours utilize MODBUS TCP to specify which values / switches may have changed. Abstract icons or images may be utilized to represent the mechanical components, as in the Reboiler HMI view -- Figure 16.

Refinery

To construct visualization, we started with pictures of real refineries we had visited to use as the basis for our abstractions. Figure 26 and Figure 27 show pictures from those visits that we used to model abstract components within our simulated refinery. We did not include all of the relevant pipes etc, between our components, although we assumed they were there for computational purposes. This includes the mechanical valves and sensors used to turn flows on and off and to sense levels and other key aspects. Omitting such features allowed us to minimize clutter in the visualization, while presenting the true overall sense of a refinery.

There are four RTUs embedded within the visualization: the SupplyRTU, the HeaterRTU, the C-301RTU, and the BurnerRTU. With the exception of the BurnerRTU, the other RTUs have labels floating above them in the visualization space. We show a number of views to allow the reader to consider the simulation from different angles. Figure 28 and Figure 29 show how the burner is embedded within the overall model. Figure 30 and Figure 31 show enlarged views to more clearly see the labels for the RTUs. These labels show three RTUs and the product tank. We also see towers that could be raffinate splitters; however, they are not all used in this particular scenario. Other figures are abstract representations of refinery objects, such as blow down drums, storage tanks, *etc.* Figure 32 thru Figure 41 show the simulation from different angles.

Figure 26: Real-World Refinery.

Figure 27: Distilling Tower at Hydrocracking unit DSC03097.

Figure 28: Refinery Burner Simulation View 3.

Figure 29: Refinery Burner Simulation View 4.

Figure 30: Refinery Burner Simulation Headers View 1.

Figure 31: Refinery Burner Simulation Headers View 2.

Figure 32: Refinery Burner Simulation View 5.

Figure 33: Refinery Burner Simulation View 6.

Figure 34: Refinery Burner Simulation View 7.

Figure 35: Refinery Burner Simulation View 8.

PHYSICS VIEW

The support to incorporate true physics in calculations related to this simulation is one of the more unique and valued aspects of this particular simulation using the Umbra software package. We implemented simple physics calculations in various aspects and designed the code so we can easily increase the physical fidelity later, as resources become available for additional development/improvements. The simulation considered the physics of the sensors, the flames produced, the fluid flows (including the product being produced by the refinery and the fuel used during the refining process), and heat generation. We consider each of these briefly.

Sensors

All of the sensors in this project/demo are simulated; they are not actual physical components of the simulation. However, we do depict some (flame detector, heat thermocoupler, *etc.*) sensors as components in the visualization. The Umbra component codes provide timing calculations for variations in the sensors for now.

The flame detector sensor goes on within a few seconds when both: 1) the pilot light gas is flowing, and 2) the PLC Controller turns on the solid-state ignition transformer. Another placeholder in the code allows a calculation to determine when the flame is hot enough to actually light the burner. For now, it is a simple calculation of a few seconds in time, and then the Umbra code turns on ThermocouplerSensorAPilot.

Additional calculations allow the code to decide whether or not the heat being generated is too hot, which is related to gas volume being burned. When that occurs, the system turns on ThermocouplerSensorBTemp.

Currently, there is no implementation for the high and low pressure interlocks, which are presumed to be some type of mechanical interlocks. However, we have defined places for information related to these interlocks within the code and have defined rungs to be used in the future with the PLC ladder logic program if the interlocks are to be included in the simulation.

Flame

We did not do a lot of physics calculations with the flame in the current simulation. We used an existing flame system and adjusted the size and color and texture of the flame to what we considered reasonable and similar to what we expected based on some experience with pilot lights and flame burners. Additional work needs to be done in this area; accordingly, we implemented input variables into the software components that, in time, would allow the user to specify the exact type of fuel being used so that future calculations could be made to adjust the flame color, height, and ultimately heat calculations as noted below.

Further, we did not calculate the geographical dispersion of any gas buildup, which could be affected by wind flow, to determine exactly how big a ball of flame to produce, but we made a general assumption that the flame produced would be bigger initially, similar to what happens when turning on a gas stove burner.

Fluid Flow

In regards to the fuel fluid flow, we did include a simple formula to calculate how much fuel was being burned in comparison to how much fuel had flowed (was flowing) out of the pilot light pipe line and the main burner pipe line, and how much fuel may have dissipated on the wind. We did not actually calculate the diameter and actual flow rate from the fuel pipes based on outside air temperatures and altitude; although

that could affect the end results, we left that for a future improvement. Further we did not calculate or consider buildups within the pipes as causing any obstructions to affect the flow for either the burner fuel or the product being processed at the refinery. However, if too much gas has flowed, the system “explodes”.

In regards to the flow of the product being refined, we did not calculate how much fluid is actually in the pipes, rather considered the amount of fluid present to be a zero sum between the various tanks – that is, the amount taken from one tank equaled the amount placed in another tank. We do not take into consideration the pressure in the pipes for this simulation.

Heat

We included just enough physics to reach the fidelity required for our scenario regarding the heat produced. Additional opportunities exist to expand this aspect. As noted above, we did not adjust the color of the flame to indicate heat levels. Physically, in the real world, the color of the flame potentially provides information about the type of fuel being burned and how high a temperature the flame has reached. The size of the flame based on the amount of fuel burning, can also be used in calculating the heat produced. The outside temperature may also have an effect, as well as the amount of oxygen in the air, possibly related to altitude. We did not consider those aspects for this learning simulation.

In addition, we did not calculate heat flow patterns within the tank; thus, we did not calculate the actual heat differentials between the top and bottom of the tank, in those cases where thermal levels form.

We did provide a simple calculation to specify how much heat (in BTUs) was being generated.

The system also determines the temperature of the liquid being processed, and turns the burner either “On” or “Off” to maintain the temperature within a desired range.

SYSTEM SAFETY VIEW

From a system safety view, we are not familiar with all aspects of system safety at real oil refineries. Therefore, we only address a few issues that we assume are actually addressed in such refineries. We implement simple features to address: 1) the gas build-up; 2) the presence of a flame; 3) the gas pressure; and 4) the heat generated.

Gas Build-up

In our model, the gas has to be turned on before the pilot light or the main burner light can be turned on, however, once on, our model will respond if the gas is turned off. According to the ladder logic program, turning the gas off after it has been turned on will cause the pilot light or burner to go out, assuming they had been lit.

Our model causes forced dispersion of gas if too much gas builds up prior to the pilot light being lit. The system loops between a dispersal state and a ready state until the pilot light “lights”. The dispersal state is entered because the system has calculated that too much gas is building up, and therefore, it turns off the gas momentarily to allow some of the gas that has built up to disperse, then it turns the gas back on and waits for the flame detector sensor to indicate that a flame is present. A timer specifies how long to wait for the gas dispersal.

This same method occurs when turning off the burner. The system has a forced timing to allow gas dispersion. That is, the system enters and remains in a locked state for a pre-specified amount of time before the user can attempt to relight either the pilot light or the main burner.

If too much gas builds up, the system signals that an explosion occurs and the visualization changes appropriately.

We do not calculate the material in the pipes to determine if there are erosion problems within the pipes or that the pipes themselves may present a problem in any way.

Presence of Flame

This system requires that the Pilot light is burning before the main burner can light.

A simulated sensor determines that the flame present is hot enough before the main burner flame can light. This requirement was included based upon information from websites that indicated such a sensor was used in some systems.

Another simulated sensor turns the burner off if it senses that the flame has become too hot.

Gas Pressure Interlocks

While we do not currently utilize the gas pressure interlocks, there is a low and a high pressure interlock designed for the refinery burner. However, we do not currently have the fidelity to detect gas pressure, so implementations using the interlocks are left for future expansion.

Heat Calculation

We calculate the heat being generated (in BTUs) and if the calculation is too high compared to an internal value previously specified, an explosion occurs. This happens by signaling through a key variable that the visualization must change.

Further, if the fluid temperature becomes too hot based on this calculation, the system can shut down the burner.

LADDER LOGIC VIEW

We briefly describe the components used in developing the ladder logic program, such as timers, and the various types of gates and switches. Then we list and describe a "buggy" program we developed to test with our refinery simulation.

Timers

Timers are used to achieve several functions within the ladder logic program running on the PLC Controller. The first set of timers appears in the fourth rung of the program. Both timers initiate at the same time. Once they initiate, they send an electrical signal that remains on while the timer counts down the

specified time. Once that specified time is reached, the signal terminates. These two timers accomplish the gas dispersal delay between the dispersal state and the ready state when attempting to light the pilot light on the refiner burner. When both timers start, the next rung, rung five is evaluated. If the ending gate, B11/4 goes "On", then the pilot light has been lit and the system advances to rung seven in the ladder logic program. Two events must occur before B11/4 turns "On": 1) the first timer must still be on, or counting down its time; and 2) the flame detector sensor must go "On".

In other words, if the system detects a flame within the timer period, the pilot light is lit, and the burner advances to the next state; if not, then a third timer initiates in rung six, which waits a set period and then returns the burner to the ready state to attempt to light the pilot light again. The system cycles in this manner until the pilot light is lit or the system is turned off. The second timer is necessary to allow a short period of time to switch states if the pilot light is not lit; otherwise, with the simultaneous termination of all gates, the system cannot sense that it must enter the dispersal state. With the completion of the dispersal state (rung six) all three timers are reset. A similar process is used for the shutdown process.

Switches

Switches are turned on when they are reached at the end of the rungs in the ladder logic program. They may then subsequently be accessed via an "AND" or an "OR" gate. Two vertical parallel lines indicate a gate and a black oblong appears between the lines when the gate is "On". The same with a line crossing from top to bottom indicates a "NOT" and the black oblong appears when the gate is "Off" (or rather is NOT "On"). If the black oblong does not appear, the current is not flowing past that gate.

"AND" operations are achieved by placing gates in sequence, and "OR" functions occur when the rung branches into parallel lines and the gates appear on separate rungs vertically. One of the rungs must have all gates activated for the current to flow past the parallel branches, thus achieving the "OR" function.

Lock-On

"OR" functions are used throughout the ladder logic program to "lock-On" actions. Once a rung has been enabled, the switch at the end of the rung goes "On". If that same switch is used in an "OR" function at the beginning of the rung, the rung may effectively be "locked-On". Thus, a button can be pressed and released in the HMI and have the same effect as the power-on button for computers. The button does not have to stay pressed down for the desired action to initiate.

B10s -- Input

The B10 gates are input to the PLC Controller from the HMI. They indicate the status of the buttons on the HMI. Figure 18 documents which button associates with which gate.

B11s -- Output

The B11 gates provide information being outputted from the PLC Controller back to the HMI. They tell the HMI the particular states that the process has reached, or is in, during the process of lighting the main burner. Those states then appear "lit" in the abstract burner HMI.

B20s -- Output

The B20s are output from the PLC Controller to the Umbra software components. They signal which switches have been turned "On" and which are turned "Off".

B21s -- Input

The B21s are the sensor switches that are set by the Umbra software components. They pass as output from Umbra and input to the PLC Controller. Several additional filler switches have been defined for future developments.

Buggy Ladder Logic Program

The buggy ladder logic program we developed demonstrates the power of this high fidelity simulation approach. The program appears in Figure 36 thru Figure 39. This program is exactly the same as the program described earlier, but we added a few errors to see how the system would respond: we switched the first two timers and mistakenly turn on the valves for the main gas line instead of the pilot light gas line. Figure 40 and Figure 41 show the resulting explosions and fires from near and far views.

Figure 36: Buggy Ladder Logic - View 1.

State 1: READY; This state is entered once the power has been turned on and waits for the START button to be pressed to initiate lighting the refinery burner.

B10/3: Start Pressed (Button);

B20/5 Turn on the Pilot Gas Cock;

B20/8: Turn on the Pilot Solenoid Valve;

B20/4: Turn on the Solid-State Ignition Transformer, unless a flame is already present;

T4:1: Timer to light Pilot light; T4:0: Timer to allow slight delay to initiate reset if needed.

Turn off the ignitor when the flame has been detected

State 2: Timer On; T4:0/TT: Timer to light Pilot light;

B21/4: Flame Detector Sensor; B20/4: SolidStateIgnitionTransformer

B20/6: Thermocoupler Leadwire A (detects heat of pilot light);

B20/7: Thermocoupler Leadwire B (detects temp of main burner);

B20/14: pilotLightHot enough to light the burner. (deleted at present, but defined in umbra interface) could be used for something else.

Figure 37: Buggy Ladder Logic - View 2.

State 3: Reset Clear State, used to allow accumulated gas to dissipate before continuing to the ready state to retry lighting the pilot light.

Turns off: All gas valves in previous rungs when B11/3 goes on: B20/2 (Pilot Gas Cock);
B20/5 (Gas Control Valve); and
B20/9 (Main Gas Valve Operator) ...etc, then

T4:2: Timer waits 40 seconds for the gas to dissipate

State 4 (B11/4 is on) : Pilot Lit: Waiting for the flame to become hot enough to light the pilot light. Turns on B11/5 (State 5: Burner Lit).

B21/5: Flame Hot Enough to Light Burner (Sensor);
B10/4: Light Burner Button Pressed (Button).
B20/2: Turn on the gasControlValve for main pipeline
B20/3: Turn on the burnerValvecontroller for main pipeline (final valve for gas to flow)

Figure 38: Buggy Ladder Logic - View 3.

State 6 (B11/6 Disperse is on): Shutdown the system: turn off the burner and disperse the gas. Wait for power turn off, or an error condition (flame too hot) to occur.

!B10/0: Power turned off.

B21/7: Flame is too hot (sensor).

B11/6: Signal that you are in the Disperse state.

B20/13: Signal that fatalErrors has occurred, occurs when Umbra turns on B21/7 thermocoupleSensorBTemp.

Reset all Timers.

Future Processing: Turn on the highPressureInterlock if Umbra sets B21/8, gasPressureTooHigh

Future Processing: Turn on the lowPressureInterlock if Umbra sets B21/9, gasPressureTooLow

Figure 39: Buggy Ladder Logic - View 4.

Figure 40: Refinery Burner Simulation View 9.

Figure 41: Refinery Burner Simulation View 10.

HMI VIEWS

We created the multiple HMI views used with this simulation with ClearSCADA commercial software. We have created both abstract and more detailed HMI views, as seen in the earlier figures within this document. In this section, we discuss in detail how we specified each of these HMI screens. First we discuss those HMI views designed for the Refinery Reboiler process, and then the abstract HMI version for tracking the Refinery Burner states. We categorize the details by the RTUs defined within the project. The reboiler process used three RTUs: the C301 RTU; the Heater RTU, and the Supply RTU. The burner used a single RTU.

Although we have included some screenshots for consistency of data, we have not included all of the data for every object because of the duplication of the definitions. Therefore, we do not include all duplicative data, making note where relevant, so the reader can refer to the appropriate figures for any data perceived as missing. Further, we do not include data where we simply used the system defaults. Further details on all files can be obtained from the authors as needed.

Refinery Reboiler HMIs

C301 RTU:

We only display nine figures containing the specifics for defining the particulars related to the C301 RTU. Each type of component defined within the HMI Software has one or more specification pages associated with it. As many of the pages are not used with this particular simulation, or simply use the default values, we will not exhaustively present all of that data. We instead present a sampling of relevant data essential to our HMI specification.

Figure 42 displays the HMI view that allows the user to interface with the reboiler in specifying the input parameters driving the simulation. The values in pink text (pink indicating that the HMI is not currently running) specify the HMI default values. These values indicate the working parameters for the refinery simulation and transmit back and forth between the HMI and the UMBRA software through the C301 RTU using the MODBUS TCP/IP protocol. Each parameter entry specifies either an analog point, or a digital point. We will see specification examples of each later in this document. These definitions appear in the left panel of the figure.

Figure 43 shows the design for the C301 RTU portion of the Reboiler HMI. The dots in the background indicate design mode for the displayed data. The pink text indicates the default values. This figure merges with similar partial figures for both the Heater RTU and the Supply RTU to create the Detailed HMI figure viewed earlier in this paper. (See Figure 17)

Figure 44 displays the particulars for defining an analog point. The user may provide specification details on eight panels within the ClearSCADA software, including categories of: Identification, User Methods, Alarm, Redirection, Analog Point, Historic, Modbus and Control. These specifications define the particular details to associate with this particular analog point during the simulation execution. We either did not use, or simply used the default values on most of these panels. However, we did specify information on the Analog Point, and Modbus panels. Figure 44 thru Figure 47 illustrate the details provided on the Analog Point panel. Note that the database panel was closed in several of those figures. This information is duplicated for all of the analog points, so we only show the details for the analog point, "Actual Outflow Heat." Figure 48 shows the details for the Modbus specification panel for the same point. We did not use any of the other panels for the simple point we defined.

In contrast, Figure 49 and Figure 50 provide the details to define a digital point. Figure 51 specifies the information necessary to use the Modbus protocol and Figure 52 specifies the Control panel information. This

Figure 42: C301 RTU Values.

Figure 43: C301 HMI.

Figure 44: C301 RTU Analog Point Specification 1.

Figure 45: C301 RTU Analog Point Specification 2.

Figure 46: C301 RTU Analog Point Specification 3.

Figure 47: C301 RTU Analog Point Specification 4.

ViewX - Reboiler Experiment.C301 RTU.C301 RTU Analog Point - Actual Outflow Heat [Read Only]

File Edit View Tools Window Help

Identification User Methods Alarm Redirection Analog Point Historic Modbus Control

In Service ☒

Confirm Disable/Enable Per User

Scanner Reboiler Experiment.C301 RTU.C301 RTU Scann ... Go To

Word Offset 2

Data Type INT

Swap Registers ☐

Scaling

Enabled ☒

Raw Full Scale 65535

Raw Zero Scale 0

Significant Change

Type Percentage of Span

Deadband 1

Ready Guest

Figure 48: C301 RTU Analog Point Modbus Specification.

ViewX - Reboiler Experiment.C301 RTU.C301 RTU Digital Point - Fill Enable Recirculate [Read Only]

File Edit View Tools Window Help

Identification User Methods Alarm Redirection Digital Point Historic Modbus Control

Data... x

Number of Bits 1

Format

States

State	Severity	Description	Color
0	None	Stopped	
1	None	Running	
2	None	State 2	
3	None	State 3	
4	None	State 4	
5	None	State 5	
6	None	State 6	
7	None	State 7	

Ready Guest

Example Projects.\~Config.Reports.PC Performance, Generate

Figure 49: C301 RTU Digital Point Specification 1.

Figure 50: C301 RTU Digital Point Specification 2.

Figure 51: C301 RTU Digital Point Modbus Specification.

Figure 52: C301 RTU Digital Point Control Specification.

information allows us to identify what the significance of the resulting value is for the specified digital point; in this case, when it signifies either open or closed.

Figure 53 specifies the C301 RTU Channel definition. This same definition is used for all of the RTUs we used. This channel definition only used the Modbus and Control panels. Figure 54 specifies the connection protocol type for the channel primary port.

Figure 55, Figure 56, and Figure 57 specify the scanning details for the holding registers, the input registers, and the output coils. Figure 55, and Figure 57 show the top portion of the panel; Figure 56 shows the lower portion of the scanner panel. The lower portion of the scanner panel is the same for all scanners defined in our simulation.

Figure 58 shows the primary port used for scanning the output coils. This information is the same for the holding registers and the input registers. To determine whether an analog or digital point is specified through either the input registers or the output coils, double click the value on the HMI to select what information is wanted concerning that point.

For example, you can determine what references a point, or the status of the point. The status provides all information stored in the internal database concerning that point. The first four analog points are read from the input registers, the next three analog point values are kept in the holding registers. The last analog point is read from the input registers. The two digital points are processed through the output coils. All of this information, and more, can be obtained by highlighting each point respectively in the database panel, right clicking, and then selecting "view status". To provide all of that detail in this document is unnecessary to understand the overall architectural design and would be too exhaustive to include everything. If additional information is needed, please contact the authors.

Figure 53: C301 RTU Channel Definition.

Figure 54: C301 RTU Channel Primary Port.

Figure 55: C301 RTU Holding Registers.

Figure 56: C301 RTU Input Registers.

Figure 57: C301 RTU Output Coils.

Figure 58: C301 RTU Scanner Output Coils.

Heater RTU:

The heater RTU definition is very similar to the C301 RTU definition. We do show some duplication of the data in this section so the reader may understand the general specification process.

Figure 59 shows the interface values. Figure 60 shows the HMI for just the Heater RTU. The definitions for the analog points and digital points are essentially the same, so we do not duplicate that data for the remaining RTU definitions. The Heater RTU channel is defined in Figure 61 and Figure 62; it is a duplicate of the data used for the C301 RTU. However, the primary port, shown in Figure 63 for the Output Coils, is different. We show the upper portion of the scanner panel for the holding registers, the input registers, and the coils in Figure 64, Figure 65, and Figure 66. These specifications are essentially the same as those for the C301 heater, except for the number of data values being passed / held in the various registers. Figure 67 shows that the primary port for the input registers is the same as that previously shown for the output coils; this port is also the same for the holding registers.

Figure 59: Heater RTU Interface Values.

Figure 60: Heater RTU HMI.

Figure 61: Heater RTU Channel Definition.

Figure 62: Heater RTU Channel Primary Port.

Figure 63: Heater RTU Output Coils Primary Port.

Figure 64: Heater RTU Holding Registers.

Figure 65: Heater RTU Input Registers.

Figure 66: Heater RTU Output Coils.

Figure 67: Heater RTU Scanner Input Registers.

Supply RTU:

Again, the Supply RTU design is similar to the previous two RTUs specified above. Figure 68 thru Figure 74 show the some of the same panels for the Supply RTU as shown for the C301 RTU and the Heater RTU to illustrate the duplicative nature of the information.

Figure 68: Supply RTU Values.

Figure 69: Supply RTU Definition.

Figure 70: Supply RTU Channel Definition.

Figure 71: Supply RTU Holding Registers Scanner Definition.

Figure 72: Supply RTU Holding Registers Primary Port.

Figure 73: Supply RTU Scanner Input Registers.

Figure 74: Supply RTU Output Coils Scanner.

Refinery Burner HMI

The required data to specify the Burner RTU, is the same type of data. Refer to Figure 75 thru Figure 81 for the specific details.

Figure 75: Burner Gas On Button Definition.

Figure 76: Burner State Definition.

Figure 77: Burner RTU Channel Definition.

Figure 78: Burner Input Discretes Scanner Top.

Figure 79: Burner Input Discretes Scanner Bottom.

Figure 80: Burner Input Discretes Primary Port.

Figure 81: Burner Output Coils Scanner.

SOFTWARE COMPONENT VIEW

In general the Umbra simulation software links together various predefined software components to derive a new simulation. Wherever possible we used preexisting components to describe and process the necessary data flow. For the burner simulation, we defined a new burner system component. For the refinery simulation, components were defined for the CookingTankController, and the StorageTankController. Refer back to Figure 19, Figure 20, and Figure 21.

Burner System Component

The Burner System Component takes four external inputs: A bit value, indicating whether or not the solid state ignitor is turned "On"; a double value indicating the flow rate; another double value indicating the quantity of gas available to burn; and third double value indicating the simulation time.

There are three internal inputs that are pre-initialized: an integer value indicating the type of ignition fuel; a double value specifying the heatLimit that is set internally to 1000.0; and, another double value that is used as the explosionThreshold, which is also initialized to 1000.0.

There are three outputs, and one feedback variable. Two of the outputs are Boolean values: one indicating whether or not an explosion is underway, and another indicating whether or not the burner is lit. The feedback specifies the rate at which the available gas is being burnt.

Essentially, during the update process, the component completes the initializations based upon the input values, and determines the amount of heat (in BTUs) being generated. If the burner is not burning, the output heat is set to zero; otherwise, the burner is burning so the calculated heat is set. If the calculated rate is greater than the heat limit, then the explosion indicator is set to true.

If an explosion has not occurred, the system sets the burner either "On" or "Off", depending upon the input conditions. If gas is burning, the system calculates how much gas has either burned or dissipated due to environmental conditions.

Storage Tank Controller

The storage tank controller operates similar to a manifold process. Figure 21 indicates the requisite inputs and outputs for our simulation.

Cooking Tank Controller

The cookingTankController component is used for both the Heater and the product tank for this simulation. Figure 20 indicates particulars related to the internal working of the cookingTankController. The indicated double switches control the logic flow in processing the fluid in the tanks. For additional information contact the authors. Figure 21 provide further information about the inputs and outputs for this component when used for the Heater and for the product tank.

COMPONENTS LOGICAL VIEW

Burner Component Logic

Essentially, during the update process for the burner component, the component completes the initializations based upon the input values, and determines the amount of heat (in BTUs) being generated. If the burner is not burning, the output heat is set to zero; otherwise, the burner is burning so the calculated heat is set. If the calculated rate is greater than the heat limit, then the explosion indicator is set to true.

If an explosion has not occurred, the system sets the burner either "On" or "Off", depending upon the input conditions. If gas is burning, the system calculates how much gas has either burned or dissipated due to environmental conditions.

Storage Tank Logic

The storage tank, based on the input flow and the output flow, determines the net flow, the actual outflow, the quantity of fluid in the tank, and the overflow rate.

Cooking Tank Logic

The cooking tank essentially takes the heatRateIn, which is the amount of heat going into the cooking Tank from the burner; the heatOut, which is the heat lost from the system; and, the inflowTemp, which is the temperature of the material flowing into the tank. The component then determines the temperature of the

material in the tank. Thus, this component is essentially a storage tank that can be heated. The material comes into the tank at one temperature then mixes instantly. With no heat, the temperature goes toward the weighted average of the new material plus the old material.

CONCLUSIONS

We have created this architecture after the fact, thus it does not simply follow standard architectural descriptions. We have included views/aspects that are not typically included with other software applications, but that are necessary to understand this demonstration. Further, this test bed was developed for a broader community and not specific stakeholders. Key to the success of this model development and presented in this report are the concepts of the multiple aspects of model design and development that must be considered to capture the necessary model representation fidelity of the physical systems. The refinery burner model and simulation is one of the capabilities included within Sandia National Laboratories' Virtual Control System Environment (VCSE).

We have determined this is an excellent approach to control testing costs as we use real-world equipment linked with software simulations substituting for those real-world components that might otherwise be damaged or destroyed during processing with inherent errors. Thus, many scenarios may be tested in real-world settings with minimal cost. Consequently, our particular simulation makes an excellent training tool for understanding the impact of control exchanges during the refinery process.

For next steps, we recommend increasing the physical fidelity of those aspects of the simulation that are related to key operational concerns for this type of refinery. We need to further explore existing safety features and make sure those aspects are included in this test bed. We need to further explore aging aspects of the tanks to address how that would impact future functioning. We should calibrate our assumptions in regards to the refinery process with specific processes, such as isomerization units that are utilized at various refineries. We should further validate the model with known issues that have occurred and have been analyzed over the years related to refinery processes. We should explore new technologies to address any safety or other issues identified to improve these refinery processes in addressing the security of these control systems and facilities.

REFERENCES

1. Anderson, W.E.; Chavez, A.R.; Lee, E.; and Trevino, C.M.; Generalized Code Obfuscation. SAND2010-6673. Albuquerque, NM, USA: Sandia National Laboratories.
2. Bass, L., Clements, P., & Kazman, R. *Software Architecture in Practice, Second Edition*. Carnegie Mellon Software Institute.
3. Brochure, L. I. (2008). *It's On Fire Catalog 2008 Volume BCS-2008*. Retrieved February 2009, from www.lesman.com.
4. Cassidy, R.H., Chavez, A.R.; McDonald, M.J.; Pollock, G.M. and Richardson, B.T.; "Protecting SCADA Supply Chain Using Trust Anchors," under development, SAND2011-XXXX. Albuquerque, NM, USA: Sandia National Laboratories.
5. CBS Safety Video: Explosion at Bp Refinery, Texas City, Texas. March 05, 2007, <http://www.youtube.com/watch?v=c9JY3eT4cdM>.
6. ClearSCADA, www.clearscada.com, 2009.
7. McDonald, M. J., et.al. (2010, February). *Modeling and Simulation for Cyber-Physical System Security Research, Development and Applications*. SAND2010-0568. Albuquerque, NM, USA: Sandia National Laboratories.
8. McDonald, M. J., (2010), Personal Umbra Refinery Working Papers. Albuquerque, NM, USA: Sandia National Laboratories.
9. Pollock, G. M. (2009) BURNER OBJECTS-Definitions_v2.docx, personal working papers.
10. Pollock, G. M. (2009) BurnerObjectsFunctionsList_v2.docx, personal working papers.
11. Pollock, G. M. (2009) Burner Valves Definitions.docx, personal working papers.
12. Pollock, G. M. (2009) Initial_States_and_Transitions_v2.docx, personal working papers.
13. Pollock, G. M. (2010, July). *Umbra Refinery*. SAND2010-5254P. Albuquerque, NM, USA: Sandia National Laboratories.
14. Pollock, G. M. (2010, July). *Directions to Conduct the Refinery Burner Simulation*. SAND2010-6259P. Albuquerque, NM, USA: Sandia National Laboratories.
15. TOPDOC NexGen User's Guide, SoftPLC Corporation, Spicewood, TX, www.softplc.com, 2008.
16. Wonderware, <http://global.wonderware.com/EN/Pages/default.aspx>, 2008.
17. Welch, Brent B. and Jones, Ken, *Practical Programming in Tcl and Tk*, <http://www.beedub.com/book/>.

DISTRIBUTION:

(Electronic Copies)

1	MS 1248	B.P. Clifford, 5623
1	MS 1073	B.K. Cook, 5624
1	MS 0671	J.M. Depoy, 5628
1	MS 1248	S.M. Rinaldi, 5643
1	MS 1027	G.M. Pollock, 5624
1	MS 1248	R.D. Halbgewachs, 5623
1	MS 1027	M.J. McDonald, 5623
1	MS 0671	B.T. Richardson, 5628
1	MS 0671	J. Mulder, 5628
1	MS 0671	A.C. Riehm, 5628
1	MS 0671	W.D. Atkins, 5628
1	MS 0899	Technical Library, 9536