TULSA Water and Sewer Department

SCADA System Improvements

PLC Software Standard

FINAL

PRESENTED TO

Cindy Cantero
City of Tulsa
Water Pollution Control
175 E 2nd Street, Suite 1400, Tulsa, OK 74103

PREPARED BY

Tetra Tech
7645 E. 63rd St.,
Suite 301
Tulsa, Ok 74133

P: *(918) 249-3909*
www.tetratech.com

TETRA TECH

200-11383-19001
April 22, 2024

CONTENTS

List of Tables

List of Figures

Revision History

After the Supervisory Control and Data Acquisition (SCADA) system has been modified or updated, this document should be revised to reflect the changes. Changes may include software configuration or upgrades, equipment functionality, and Add-On Instruction restructuring.

The version is broken into two parts: major (**X**.0) and minor (1.**X**). A major version is reserved for adding or removing sections of this document. A minor version is reserved for modifications to existing sections.

Version	Date	Description
A	July 9, 2021	Draft submitted to the client.
B	April 4, 2022	Final submitted to the client.

1 INTRODUCTION

The purpose of this document is to provide standard methods and guidelines for the configuration and development of the Programmable Logic Controller (PLC) software component of the SCADA system. The topics covered in this document include, but are not limited to program structure, programming language, program documentation, best practices, and standard code templates.

This document is meant to provide a sustainable standard for training and future development. These standards apply to external programmers, engineers, consultants, and internal developers who create or edit PLC programs in the system.

This PLC software standard has been developed to provide internal and external programming entities a source for standardization across all components of the City of Tulsa Wastewater system. Consistency between projects and across different facilities increases the development efficiency because logic can be re-used. In addition, modifications are easier to troubleshoot and less likely to contain errors, thus improving confidence in automatic control. With a standard in place, multiple integrators can use the system while enforcing the quality of their work.

2 PROGRAMMER REQUIREMENTS

It is the responsibility of the programmer to follow the software standards. Failure to do so will result in the City rejecting the submitted program and requiring resubmission per the standards at no additional cost to the City.

Programming modules that do not prescribe to the standard require approval from the City. Any new or modified modules shall be documented in a manual and submitted to the City for incorporation into this standard.

3 DEFINITIONS

Table 3-1 Definitions

Term	Definition
Add-On Instruction (AOI)	A standardized module of code that can be reused multiple times in the same program.
Alarm	An audible and/or visible means of indicating to the operator an equipment malfunction, process deviation, or abnormal condition requiring a response.
Analog Input	A numerical current or voltage value read by the controller from a field device that represents a process value.
Analog Output	A numerical current or voltage value written by the controller to a field device.
Calculations	The formulations necessary to perform the automatic controls that have been defined.
Commissioning	Procedures prior, or related, to handing over a system for placing into service. These procedures often include acceptance testing (FAT, SAT, and SIT); handing over of drawings and documentation; delivering instructions for operation, maintenance, and repair; and providing training to personnel.
Console	The hardware, software, and furniture or enclosure at which users monitor and/or control the process, which may include multiple stations, communication devices, and other devices (e.g. cameras, barcode devices, pushbutton stations).

Control Room	A room with at least one HMI console from which a process is monitored and/or controlled and possibly containing other control system equipment and/or other facilities for operators.
Control System	A system that responds to input signals from the equipment under control and/or from an operator and generates output signals that cause the equipment under control to operate in the desired manner.
Controller	The hardware which executes functions for monitoring and control of one or more process variables.
Digital Input	An on/off value read by the controller from a field device.
Digital Output	An on/off value written by the controller to a field device.
Faceplate	A display, part of a display, or pop-up used for monitoring and/or direct operation of a single control loop, device, sequence, or other entity.
Function Block Diagram	A programming language consisting of functional blocks arranged on a grid. Network lines are used to connect the blocks to each other and to input and output variables.
Human Factors Engineering	A scientific discipline concerned with the understanding of interactions between human and other elements of a system that applies theory, principles, data, and methods to design in order to optimize human well-being and overall system performance.
Human Machine Interface	The collection of hardware and software used by the operator and other users to monitor and interact with the control system and with the process via the control system.
Ladder diagram	A programming language consisting of rungs, contacts, coils, and functions, arranged in a ladder structure.
Monitor (verb)	To maintain awareness of the state of a process, by observing variables or the change of variables against limits or other variables, to keep track of operations and enable timely and appropriate response to abnormal conditions.
Operator	The primary user of the HMI, the person who monitors and makes changes to the process.
Process and Instrumentation Drawings (P&ID)	Drawings that depict all field devices in a given system and detail which signals are connected to the controller.
Process Control Narrative (PCN)	A document describing how the system should be configured and controlled.
Programmable Logic Controller (PLC)	An industrial computer control system that continuously monitors the state of input devices and makes decisions based upon a custom program to control the state of output devices.
Remote Terminal Unit (RTU)	One or more monitoring and/or control devices at a location geographically separate from, but communicating with, the control center.
Supervisory Control and Data Acquisition (SCADA)	A system for monitoring and control of processes which are geographically widespread. This includes all equipment and functions for acquiring, processing, transmitting, and displaying the necessary process information.
Tag	The unique identifier assigned to a process measurement, state, calculation, device, or other entity within the HMI or controller.
Task Analysis	A method of extracting a user's requirements based on a review of tasks performed by the user.
Usability	The extent to which a system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.
User-Defined Data Type (UDT)	A custom set of tags grouped together under a parent tag.

Validation	Process of demonstrating by examination, testing, or other objective evidence that the HMI, as installed, meets applicable requirements and specifications.
Verification	Process of demonstrating by examination, review, testing, or other objective evidence that the outputs of an HMI lifecycle activity meet the objective and requirements defined for the activity.

4 CONTROLLER PROPERTIES

These standards have been developed for use with Rockwell Automation's Logix family of hardware and Studio 5000 software. The PLCs should meet the requirements specified in the PLC Hardware Standard.

The controller should be named in accordance with the plant, process area, and building ID's defined in the Tag Naming Standard. The controller properties should have a description that defines the PLC name.

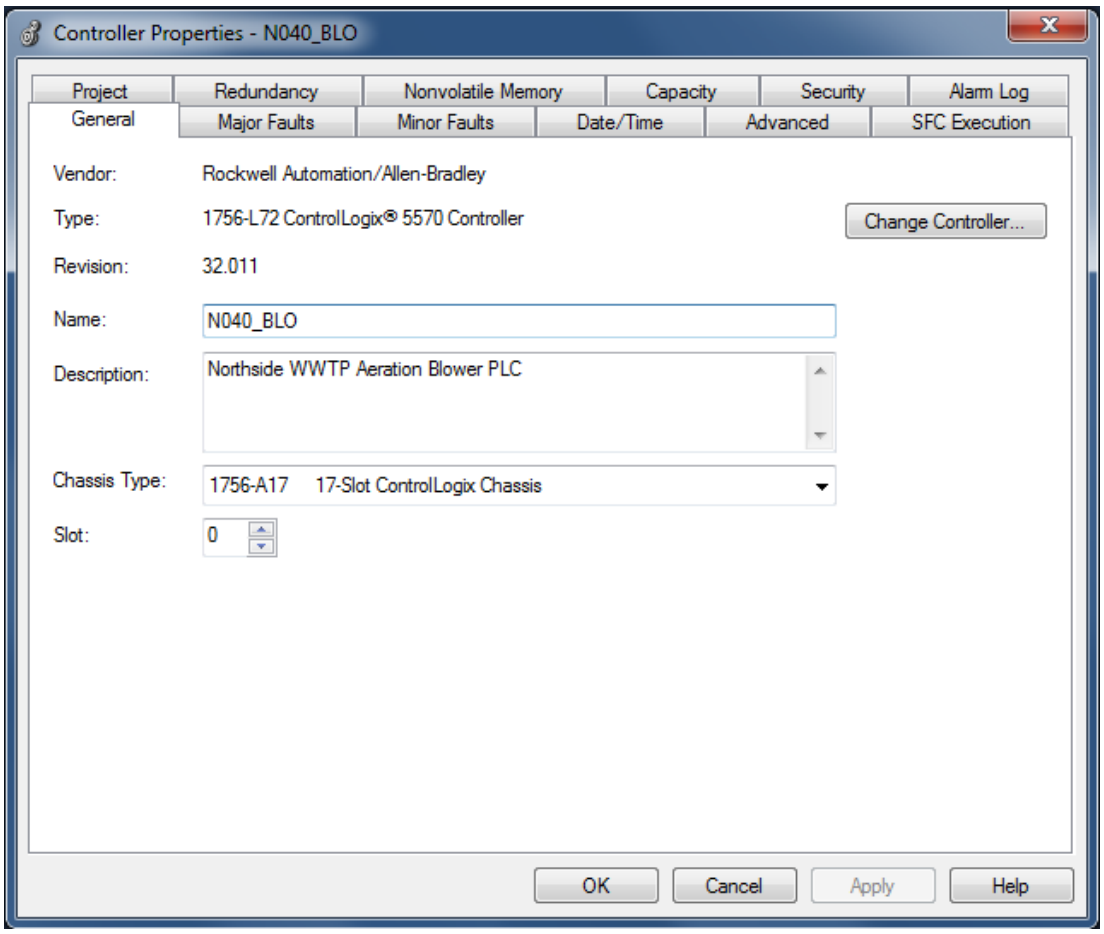

Figure 4-1 Controller Properties

5 VERSION CONTROL

5.1 Firmware

All PLCs should use firmware revision 33 to maintain consistency throughout the facilities. If the City decides to upgrade the firmware, it can be upgraded using ControlFLASH software, and should be implemented on all PLCs throughout all facilities.

5.2 Controller CPU Utilization

The Rockwell Automation PlantPAx Distributed Control System guide has published recommendations for monitoring CPU utilization within its family of controllers. Free process controller CPU time is required to handle communication, abnormal conditions, and other transient loads. When defining the application code, make sure the CPU utilization of the process controller can accommodate these values:

- In the development environment, CPU utilization is recommended to be less than 50% to allow for the additional CPU load that is experienced in the production environment.
- During the operation of the system, monitor the CPU utilization, especially after a change to the application code. It cannot exceed 75% for a Simplex controller or 50% for redundant controllers.
- During the design of the application code, it is important to account for software components, such as Wonderware InTouch or Historian. The software is actively collecting data from the controller, so be sure that CPU utilization is less than stated limits to allow for communication with the SCADA system elements.

5.3 File Naming and Save Location

As with the controller naming, PLC file naming should be based on the tag naming standard. The file name should be the controller name plus the date that the file was saved. For example, the Northside WWTP Aeration Blower PLC program would be saved to the network as N040_BLO_yyyymmdd, where “yyyy” represents the year, “mm” the month, and “dd” the day of when the PLC program was saved with data uploaded from the controller.

PLC files should be saved to a centralized network shared folder, as appropriate to the plant:

\\WS204607\NorthsidePLC
\\WS204607\SouthsidePLC
\\WS204607\HaikeyCreekPLC
\\WS204607\LowerBirdCreekPLC

Every backup of a PLC file should be saved to its corresponding folder.

5.4 Add-On Instructions

In the Add-On Instruction (AOI) definition, there is an option to set the major and minor revision number. The major revision should be incremented when there is a change to the code structure or tag names. The minor revision should be incremented when there is a change to the internal code that does not affect the external structure of the AOI. The revision notes should document each time the major and minor revisions change and include the programmer name and date. Any changes to an AOI should be implemented to all PLCs in the system.

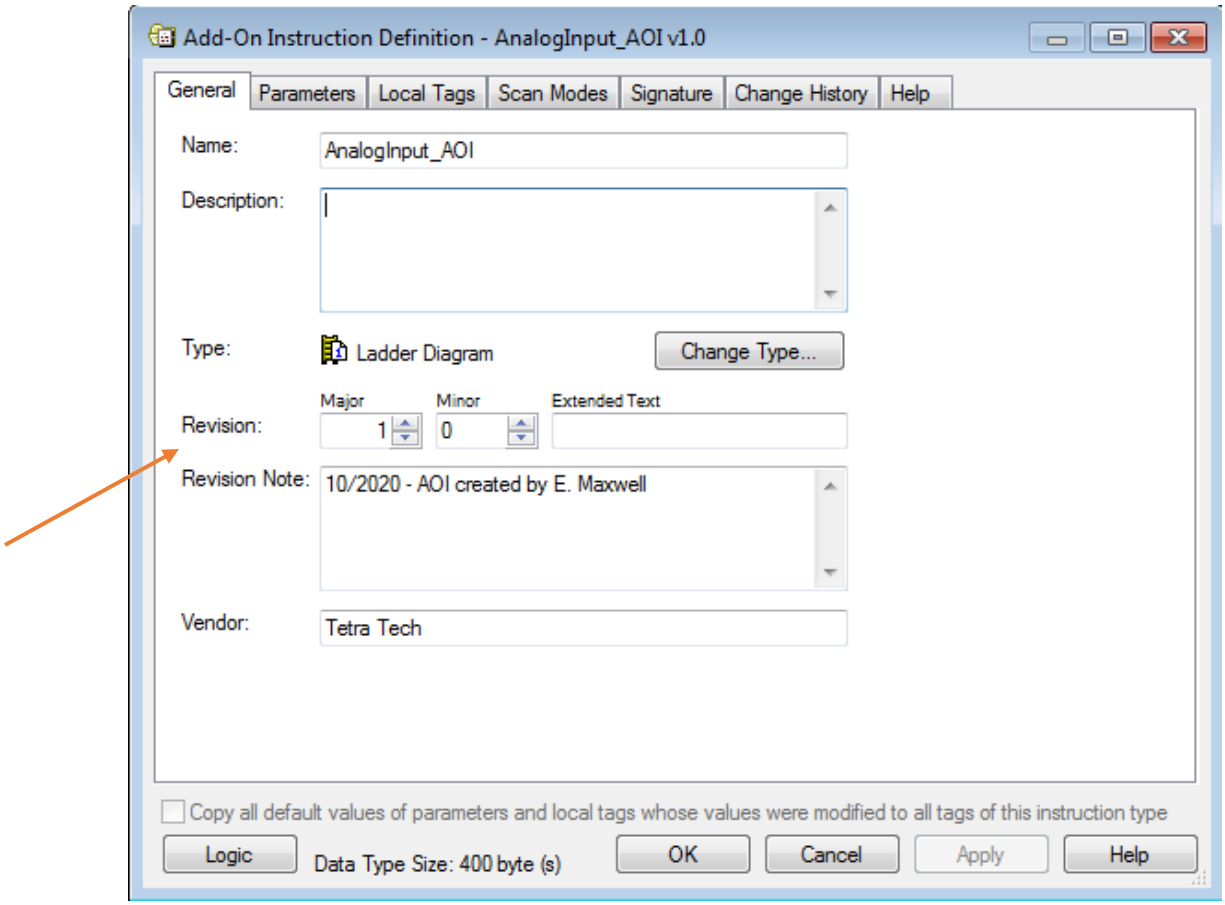

Figure 5-1 AOI Revision Settings

6 PROGRAM LAYOUT

6.1 Tasks

The default project configuration contains one continuous task for all logic, MainTask. This is sufficient for most applications, although some situations may require more than one task.

There are three types of tasks available:

- Continuous – where most of the code is expected to reside. There can only be one continuous task.
- Periodic – used for slower processes or when time-based operation is critical.
- Event – used for operations that required synchronization to a specific event.

Each non-continuous task is assigned a priority, with the continuous task having the lowest priority. If there are too many tasks, the continuous task can take too long to execute, tasks can experience overlaps in execution, and controller communication can be slower.

6.2 Routines

Tasks host routines, which contain the executable code. Each task has a MainRoutine that calls the subroutines to execute using Jump to Subroutine (JSR) commands. Routines should be organized based on equipment or module type. For example, all analog inputs should be contained in an “Analog” routine,

all motors in a “motor” routine, etc. Specific functions such as automatic process control or PLC-to-PLC communication should be organized into separate routines.

Some PLCs may be large with multiple remote I/O racks and should therefore be further organized by rack number. For example, all analog inputs for Rack No. 1 should be contained in an “Analog_Rack1” routine, all analog input for Rack No. 2 should be contained in an “Analog_Rack2” routine, and so on. Automatic process control should be split into separate routines based on process.

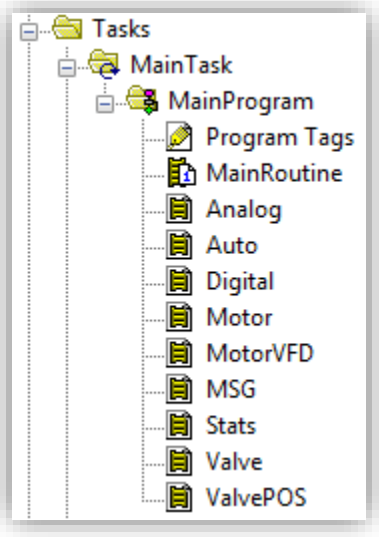

Figure 6-1 PLC Routine Structure for a Small System

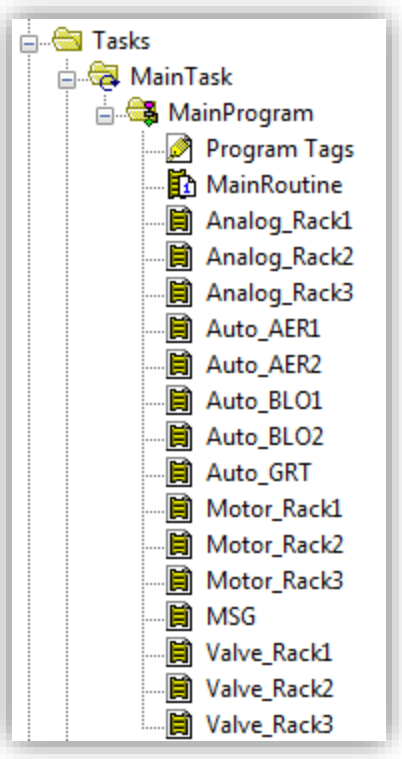

Figure 6-2 PLC Routine Structure for a Large System

6.3 I/O Configuration

The I/O Configuration of a PLC program should exactly match the card configuration of the PLC hardware in the field. The cards should be named according to the following format: Rack#_Slot#_CardType, where <CardType> corresponds to Table 6-1. An example card configuration is shown in Figure 6-3.

Table 6-1 I/O Card Types

Card Type	Description
AI	Analog Input
AO	Analog Output
DI	Digital Input
DO	Digital Output
ENET	Ethernet

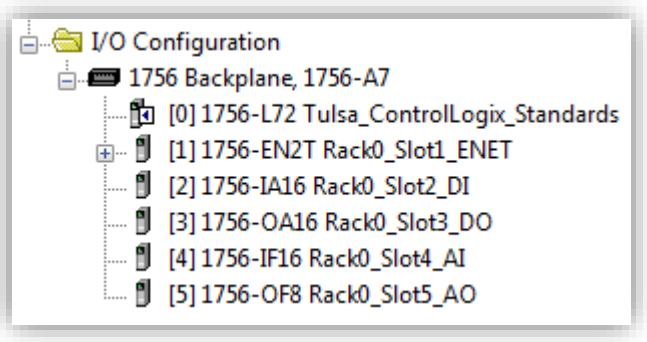

Figure 6-3 I/O Card Naming Convention

Analog cards must be configured to ensure proper scaling of raw data. The scaling from raw value to engineering units takes place within the Analog Input AOI. Similarly, the scaling from engineering units to raw value takes place within the Analog Output AOI. Because of this, both the analog input and analog output type I/O cards should be configured so there is no scaling at the card level. In the card properties, the scaling for each channel should be set to 4-20 mA for the signal and engineering units, as shown in Figure 6-4.

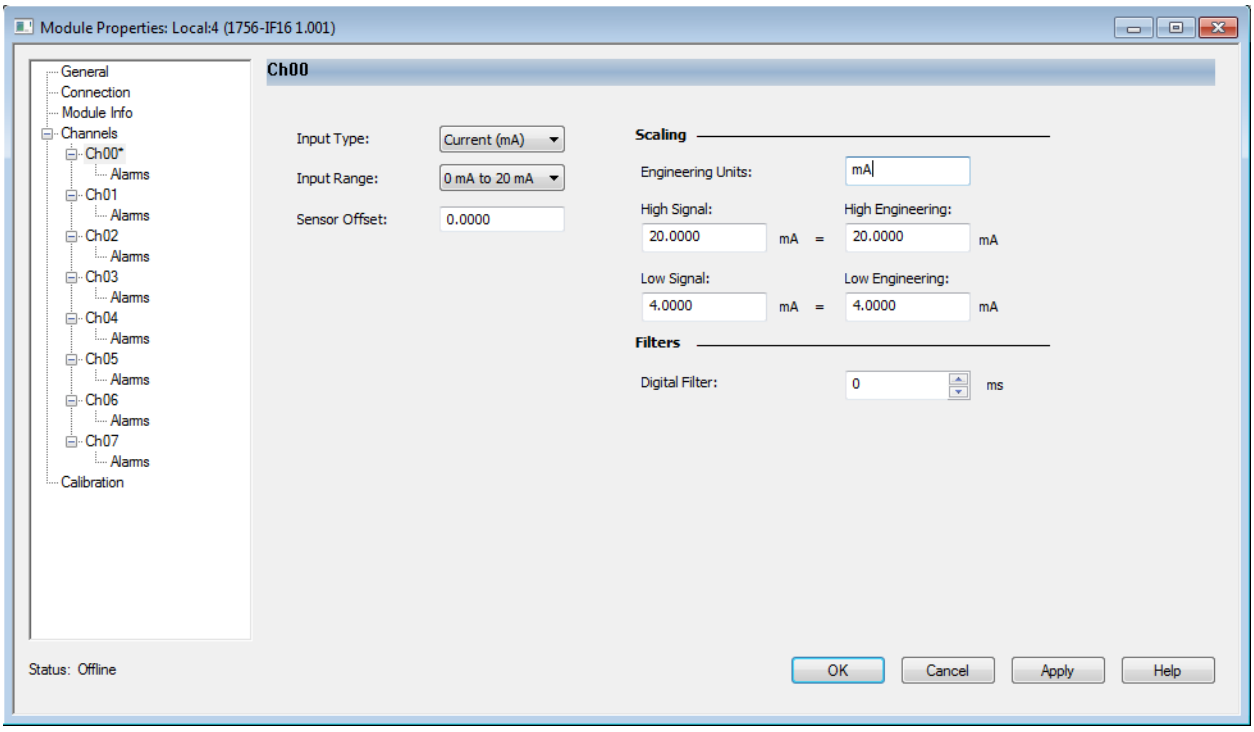

Figure 6-4 Analog Card Configuration – Scaling

Analog cards also include the option to process alarms. Alarms are generated from the Analog Input AOI, so any alarm options within the card properties should be disabled.

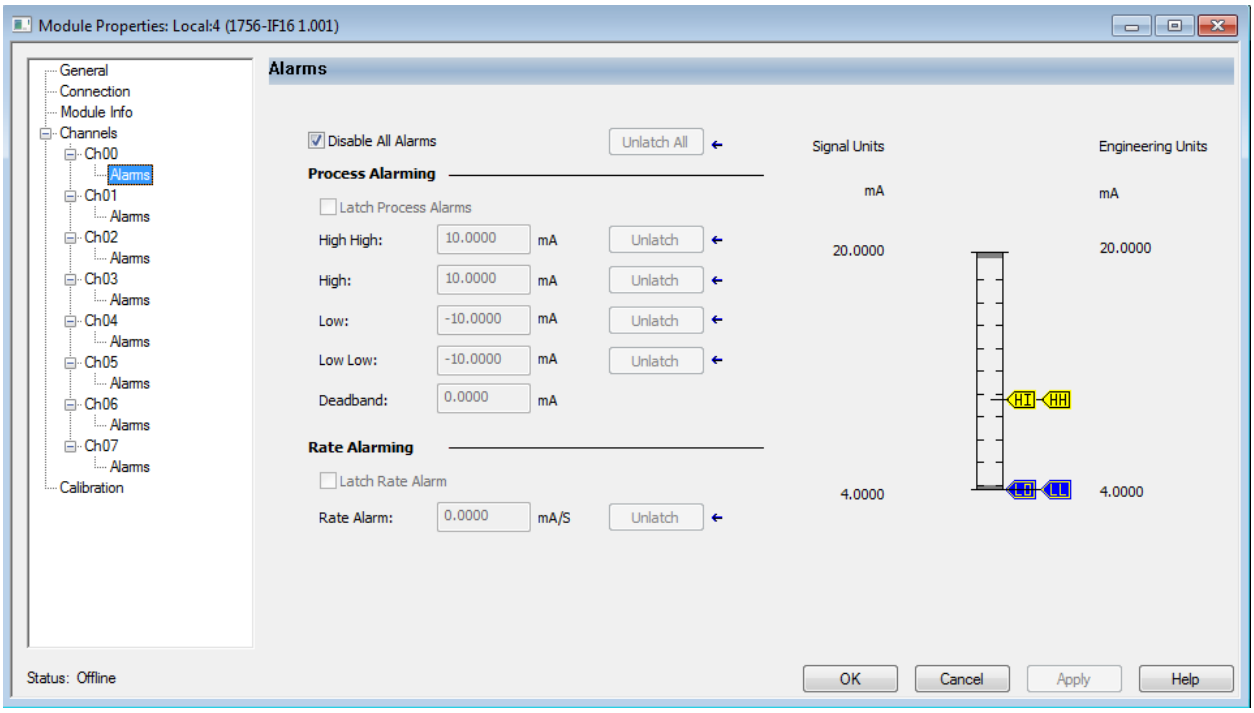

Figure 6-5 Analog Card Configuration – Alarms

7 PROGRAMMING LANGUAGE

International Standard IEC 61131-3 defines five programming languages for use within a control system: ladder diagram, function block diagram, structured text, instruction list, and sequential function chart. Ladder diagram should be used whenever possible, with the exception of specific functions that are not available in ladder, e.g. totalization, MAVE, or PIDE. If a programmer wants to use a different language for a special case, they can request permission from the City and submit their reasoning.

8 STANDARD LOGIC TEMPLATES

When creating a new PLC program, the programmer shall use the Tulsa ControlLogix Standard PLC file as a base. To ensure the latest standards are followed the Integrator/Programmer shall request from the City the Tulsa Standard PLC file for each new project. This file contains the standard Add-On Instructions as well as templates that should be followed in order to standardize the PLC logic. Each AOI has a corresponding template, located in the “Unscheduled Programs / Phases” task, that contains the proper logic structure required for using the AOI. For additional information regarding the setup and configuration of each AOI, listed in the table below, reference the corresponding AOI documentation, which should come with the Tulsa Standard PLC file.

Table 8-1 Documented Standard AOI Templates

Standard AOIs
Analog Feedback Fail
Analog Flow
Analog Input
Analog Output

Analog Trip Reset
CommFail
Discrete Alarm
Flow Totalizer
Heartbeat
Linear Scale
Module
Motor
Motor Meter
Motor Stats
Motor VFD
PLC
Valve
Valve POS
Wall Clock Data

9 RUNG DOCUMENTATION

Rung comments should be utilized to explain the logic. This aids in troubleshooting and editing the logic in the future, particularly as it relates to automatic process control. There should be a minimum of one rung comment for every 5 rungs of logic. Additionally, tag descriptions should be used for every tag in the system, including I/O.

10 TAG NAMING STANDARD

The full Tulsa Tag Naming Standards are described in a separate document. This section provides a summary of the standard with regards to PLC tag names.

Standardizing the tag naming convention makes identification of instruments, and their use within the control system, consistent throughout the facilities. In using a consistent tag structure, the engineering effort is reduced, and it is easier to maintain the control system.

The PLC tag format closely follows TMUA's Equipment ID Scheme, with additional nomenclature to categorize the equipment, PLC, and HMI functions required for monitoring and operating. Tag names should adhere to the following format:

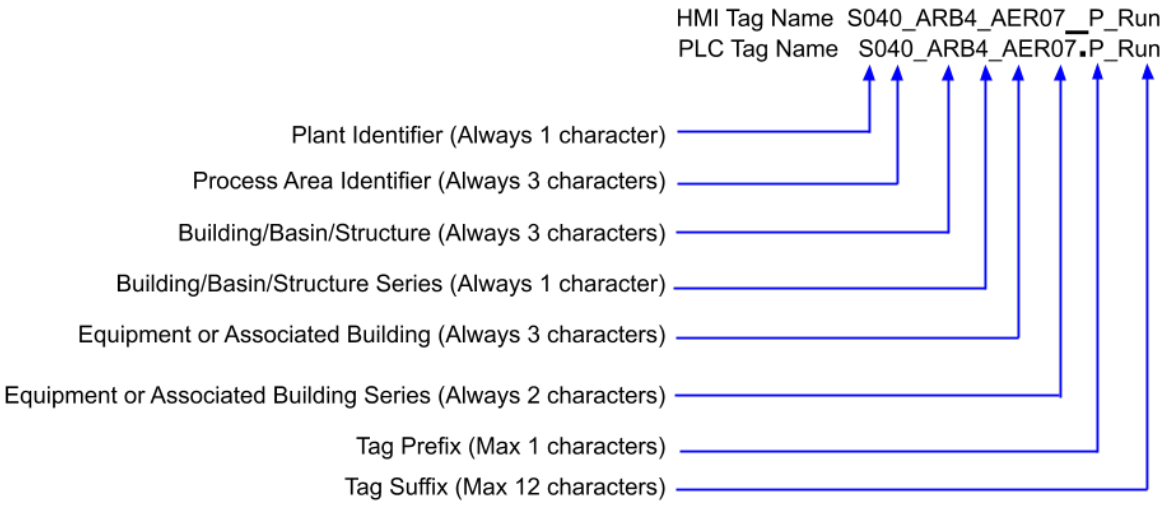

Figure 10-1 Tag Naming Format

Each tag should begin with a Plant ID, followed by a three-digit Process Area number. The Building/Basin/Structure ID and the Equipment/Building ID can be found in TMUA's All Plant Naming Conventions list. The tag prefix indicates the primary function of the tag within the context of the PLC. The tag suffix indicates the specific function of the tag within the context of the device or equipment. To maximize space and retain clarity, Camel Case should be used in the tag suffix. Camel Case uses capital letters to delimit the words in the tag function. The list of tag suffixes can be found in Appendix A.

Table 10-1 Tag Naming Standard Plant Identifier

Identifier	Plant
N	Northside
L	Lower Bird Creek
S	Southside
H	Haikey Creek

Table 10-2 Tag Naming Standard Process Area Identifier

Identifier	Process Area
001	Operation
002	Maintenance
003	Administration
004	Underground Piping
010	Bar Screen/HDW/DVS
020	Grit
030	Primary Clarifier
040	Aerator/BIO/BLO
050	Final Clarifier
060	Disinfection/Effluent
070	THK Process (DAF/THK/RDT)

080	Digester/Pasteurization
090	Belt Press/Lagoon
100	Lift Station
130	FEB
140	Waste Transfer Station
151	Stormwater/TFC
152	Oil Containment Pit
153	East Bank Junction
200-220	Electric Primary Feed

Table 10-3 Tag Naming Standard Tag Prefixes

Prefix	Description
C	Configuration – Indicates the tag modifies the configuration of an equipment or instrument.
O	Output – Indicates the tag controls a real-world output within the PLC.
H	HMI – Indicates the tag modifies a PLC register from the operator interface.
P	PLC Logic – Indicates a read only tag that is modified or calculated within the PLC.
I	Input – Indicates the tag is displaying an equipment or instrument status.

11 SECURITY AND SOURCE CODE PROTECTION

Source code protection locks out routines and AOIs so they cannot be viewed or edited unless the programmer provides a username and password. This feature should not be used for PLCs in the City of Tulsa SCADA system. If an integrator wishes to use source code protection, they must request and obtain written permission from the City. If permission has been granted, the integrator must provide the City with the credentials to access the source code.

12 DIGITAL INPUT CONDITIONING

Digital inputs should be mapped from the input address to a buffer tag. Then, if the digital input point changes, it only needs to be corrected once in the logic. Buffer tags also enable the use of debounce timers.

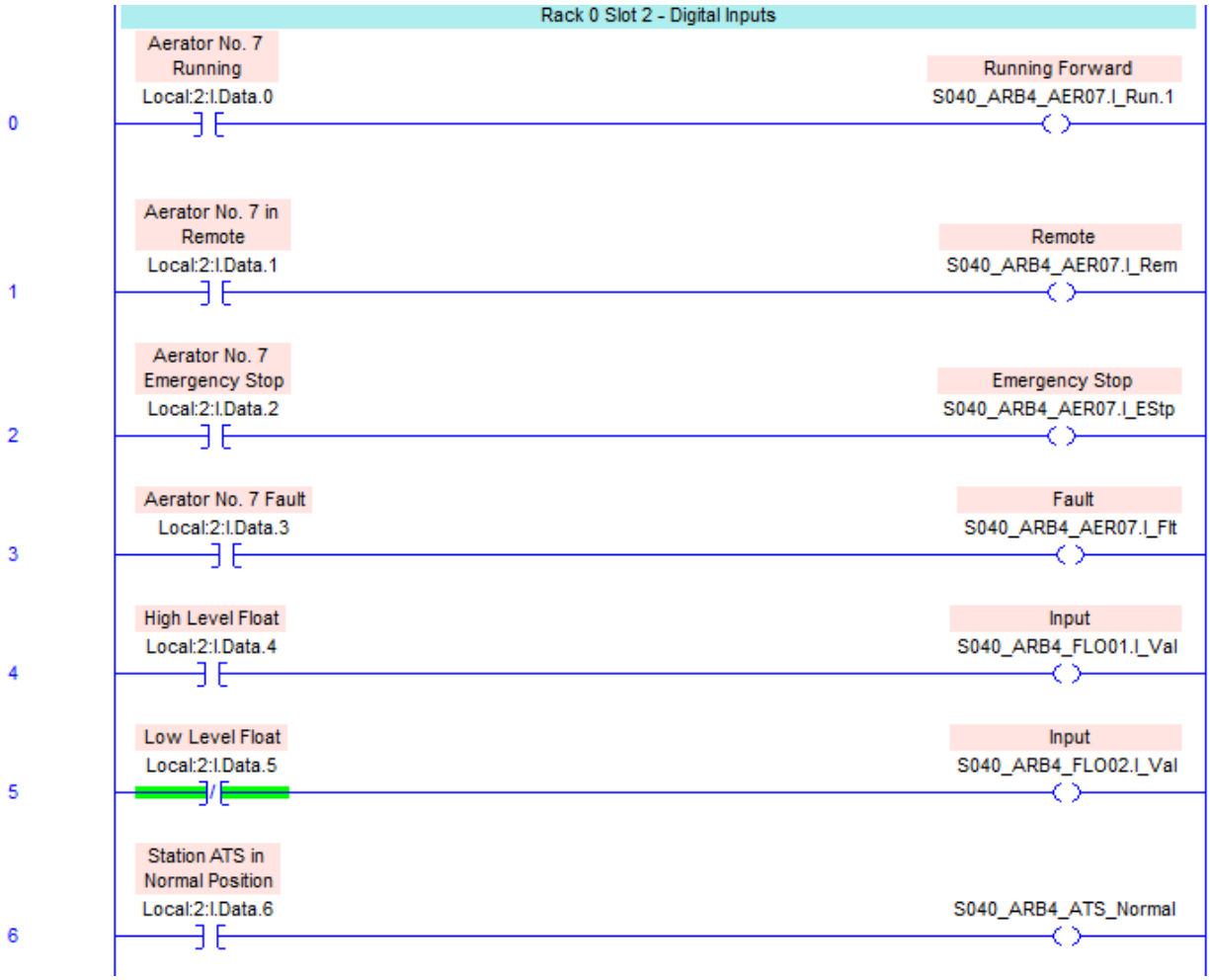

Figure 12-1 Digital Input Conditioning

13 PLC TO PLC COMMUNICATION

To maintain seamless process control, it may be necessary to transmit data from one PLC to another. This information could relate to alarms, interlocks, analog data, or equipment status. In these instances, READ type message blocks should be used whenever possible to transmit the data. WRITE type message blocks are discouraged but can be used under certain circumstances if required. The City must give permission for each case where a WRITE type message is used. The message blocks must be configured with the correct type, source, destination, length, and communication path in order to properly connect to the remote PLC.

Data should be organized into arrays to minimize the number of message blocks required. All BOOL and DINT type data should be organized into a DINT array, and all REAL type data should be organized into a REAL array. This ensures that each PLC-to-PLC communication only requires a maximum of two messages blocks. A communication mapping file should be created for each instance of PLC-to-PLC communication that outlines the tag arrays and the data that will be transmitted.

All message blocks should reside in a “MSG” routine. Message blocks need to be enabled using a self-resetting timer, which will result in the message block executing every time the timer is done. Each message block should have its own timer to avoid overlaps in execution. Additionally, the enabling of a

message block should not rely on the completion of another message block, in case the first one results in an error.

Each PLC-to-PLC message connection should include a communication fail alarm. The first bit of the DINT array that is read should be a heartbeat signal from the remote PLC. This heartbeat signal should be mapped to the C_Hb tag of a CommFail instruction. If the heartbeat does not change before the delay timer expires, a communication fail alarm is triggered.

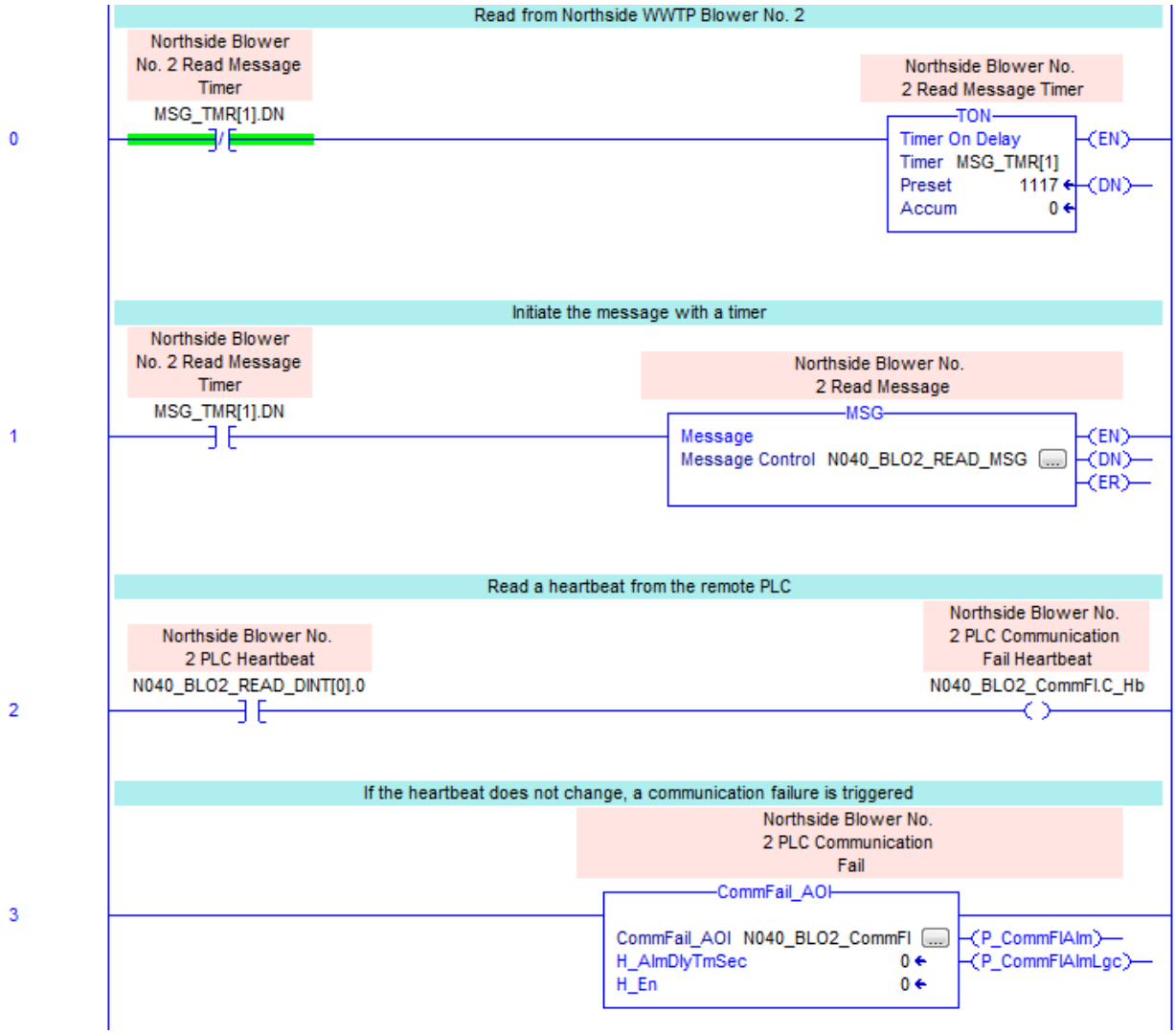

Figure 13-1 Communication Logic Example

14 POWER FAILURE HANDLING

If a PLC receives a power failure alarm, this indicates that the control panel is running on UPS power and has a short window of time before it shuts off. If a PLC shuts off due to loss of power, it can lead to numerous alarms coming into the SCADA system. When the power is resumed, equipment would return to the operation mode prior to power loss, which may not be desirable. The integrator should discuss with the City how to handle power failures on a per-site basis. Options include inhibiting equipment control

and/or inhibiting alarms when the power failure alarm is activated and before the UPS battery backup drains.

15 FIRST SCAN INITIALIZATION

When tags are created, they have an initial value of zero (for DINT and REAL data types) or false (for BOOL data types). There are certain tags that should not have a value of zero, such has setpoints, pump sequences, timer presets, and software switches. These tags need to be initialized with a default value. During the first scan of the PLC, the logic should check if these tags are set to zero, and if so, input the default value.

16 MEMORY MODULE

The program should be stored to the PLC memory module any time a change has been made. Storing to the memory module temporarily halts any outputs from the PLC, so equipment needs to be turned off or operated in hand. The integrator should coordinate with operations prior to storing the program.

The memory module can be configured to load on corrupt memory, power up, or user initiated, and can be set to place the PLC in Remote Run or Remote Program mode after it has loaded. For PLCs in this system, the memory modules should be configured to load in Remote Run mode on corrupt memory, unless otherwise specified by the City.

17 BEST PRACTICE

The following “best practices” for PLC programming have been identified and shall be followed whenever possible:

- Do not download old copies of the program to the PLC unless the PLC memory has been corrupted or erased. Changes should be made online or after the latest version has been uploaded.
- Ensure that the logic will not allow any calculations to result in an undefined value, which will fault the processor, e.g. dividing by zero or a negative timer preset.
- Minimize the use of latching and unlatching unless it is for an auto sequence.
- Do not use shortened branches.
- Do not duplicate destructive bits.
- Do not use alias tags.

APPENDIX A – TAG NAMING STANDARD SUFFIXES

Table A-1 Tag Naming Suffixes

Abbreviation	Full Description
Alm	Alarm
Avg	Average
Bat	Battery
Brng	Bearing
Byp	Bypass
Calc	Calculated, Calculation
Cap	Capacity
Cl	Close, Closed
Clmp	Clamp
Cmd	Command
Cmtv	Cumulative
Crnt	Current
Cstm	Custom
Diff	Difference
Dis	Disable
Dly	Delay
En	Enable
Eng	Engineering
Excd	Exceed, Exceeded
Fdbk	Feedback
Fl	Fail
Flt	Fault, Faulted
Flw	Flow
Freq	Frequency
Gal	Gallon
Hb	Heartbeat
Hdsh	Handshake
Hi	High
Hr	Hour
In	Input
Inh	Inhibit
Inv	Invert
Ivld	Invalid
Kgal	Thousand Gallons
L	Last
Lgc	Logic
Lim	Limit

Lo	Low
Ls	Loss
Man	Manual
Max	Maximum
Mgal	Million Gallons
Min	Minimum
Mmtry	Momentary
Mnt	Minute
Mot	Motor
Msg	Message
Mth	Month
Oor	Out of Range
Oos	Out of Service
Op	Open, Opened
Out	Output
Ovrld	Overload
Pb	Pushbutton
Pct	Percentage
Per	Period
Permsv	Permissive
Ph	Phase
Pos	Position
Pri	Priority
Pwr	Power
Raw	Raw
Rem	Remote
Req	Request
Rst	Reset
RTm	Runtime
Run	Running
Sd	Shutdown
Sec	Second
Sp	Setpoint
Spd	Speed
Stat	Statistic
Strk	Stroke
Strt	Start, Starter
Sts	Status
Tday	Today
Tm	Time
Tmr	Timer

Trbl	Trouble
Trp	Trip
Trq	Torque
Ttl	Total
Ultch	Unlatch
Val	Value
Vib	Vibration
Warn	Warning
Wk	Week
Yday	Yesterday
Yr	Year