PLC PROGRAMMING
AND
CONFIGURATION STANDARD

TMS1229

Standard Technical Specification

REVISION HISTORY

Revision Number	Date	Amendment Details	Responsible Officer
0.0	26/07/2004	Original Drafts	Geoff Timms
0.2	26/07/2004	Original Drafts – Comments included from Peter Sherriff	Geoff Timms
0.4	26/07/2004	Original Drafts – Comments included from Paul Daley	Geoff Timms
1.0	18/08/2004	Issued and Authorised	Geoff Timms
1.1	19/07/2006	Appendix A – PLC Standard Parts List Added	Rahim Janfada
2.0	14/02/2007	Standard and Advanced Functionality and Points Added	Tim Waggett
2.1	1/03/2007	Review changes added	Tim Waggett
2.1	6/06/2009	Updated Tagname convention to reflect BWEA	Tim Waggett
2.3	14/12/2009	Mods for Fairfield / Gibson Island Upgrades	Jim Stark
2.5	21/12/2009	Renamed to bring inline with the Citect HMI Document	Jim Stark
2.9	17/04/2014	Improved QUU template formatting	David Michalzik
2.10	22/04/2014	Extensively restructured and reworded to increase focus on role as standard	David Michalzik
2.12	29/10/2014	Revived merged PLC and SCADA Tag Naming Standard v0.3	Kanchana Hattotuwegama
2.13	10/11/2014	Incorporated Sections relating to SCADA from"TMS61 Electrical, Instrument, PLC and SCADA Installation V18 DRAFT"	Kanchana Hattotuwegama
3.0	Jan 2016	Issued for Use – Stakeholder comments updated	Steve Bourke
4	Apr 2019	General review	Gavin Davidson

DOCUMENT CONSULTATION

Revision Number	Date Sent	Name	Comments	
			Received	Incorporated
4	March 2019	Gavin Davidson	Y	Y
4	March 2019	Steve Bourke	Y	Y
4	March 2019	Gerard Anderson	Y	Y
4	March 2019	Technical Engineering Group(TEG)	Y	Y

QUEENSLAND UrbanUtilities

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ACRONYMS AND ABBREVIATIONS

ACRONYM	DEFINITION
DNP3	Distributed Network Protocol
EWS	Engineering Work Station
EPC	Event-driven Process Chain
HSEQ	Health, Safety Environment & Quality
IDC	Internet Display Client
IIS	Internet Information Services
ITP	Implementation & Test Plan
JSA	Job Safety Analysis
PLC	Programmable Logic Controller
PROFIBUS DP	Process Field Bus Decentralised Peripherals
PROFIBUS PA	Process Field Bus Process Automation
PTW	Permit to Work
QUU	Queensland Urban Utilities
RA	Risk Assessment
RDP	Remote Desktop Protocol
RSM	Responsive Support Manual
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SOE	Standard Operating Environment
STP	Sewerage Treatment Plant
SWMS	Safe Work Method Statement
VM	Virtual Machine
VNC	Virtual Network Computing (graphical desktop sharing system)
VPN	Virtual Private Network
WRP	Water Reclamation Plant
EWS	Engineering Work Station

1. INTRODUCTION

1.1 General

QUU operate and maintain all the Sewerage Treatment Plants (STPs) and Network Assets within the five council regions that they operate. Network Assets and the STP's have their own control systems and different process requirements. This specification aims to ensure consistent PLC programming style and structure is implemented in all the STPs and Network Assets regardless of process or hardware used.

1.2 Purpose

This document outlines the technical layout and requirements for the implementation of a PLC control system conforming to Queensland Urban Utilities (QUU) standards.

This specification provides the scope for the implementation of the PLC control system including programming and project requirements.

This specification is intended for control system engineers and other technical persons to reference during PLC code development, testing and acceptance. This specification shall be read in conjunction with TMS1202 Control System Implementation Technical Specification.

1.3 Scope

This document applies to QUU's STPs and Network Assets, where an existing PLC control system is being modified or a new PLC is being provided. The specification excludes minor changes to PLC's where the existing programming of the PLC's does not conform to this specification. In these cases, the existing programming style and structure shall be maintained by the Contractor.

This specification gives general requirements that are to be adopted for all PLC systems. Specific vendor requirements are detailed in the respective library specification:

- TMS1707 Siemens S7 PLC Library Specification
- TMS1708 GE Fanuc PLC Library Specification
- TMS1709 Rockwell Logix PLC Library Specification
- TMS1710 Siemens TIA PLC Library Specification

1.4 Current Versions of PLC

Refer to TMS1151 Preferred Equipment – Control Systems for PLC hardware.

1.5 Definitions

In this document, the following definitions apply:

Project Documentation	Governing technical documents for the specific item(s) for the specific works included or referenced in the Contract
Contractor	The entity bound (including sub-contractors appointed by the contractor) to execute the work having responsibility for design, manufacture and supply, installation, delivery, documentation and other functions as further defined in the documents related to the work.
Contract:	The agreement between QUU and the Contractor to which this specification pertains.

1.6 Reference Documents

Document Number	Title
TMS849	CITECT SCADA Configuration Standard
TMS1151	Preferred Equipment List – Control Systems
TMS1201	Instrumentation Installation - Technical Specification
TMS1202	Control System Implementation Technical Specification
TMS1647	Equipment Tagging Technical Specification
TMS1707	Siemens S7 PLC Library Specification
TMS1708	GE Fanuc PLC Library Specification
TMS1709	Rockwell Logix PLC Library Specification
TMS1710	Siemens TIA PLC Library Specification

2. STANDARDS AND REGULATIONS

Refer TMS1202 for a list of all standards and regulations applicable to PLC programming.

2.1 Units and Language

AS/ISO 1000 (metric SI system) shall be used. All documentation and correspondence shall be in the English language.

2.2 Sub-CONTRACTORS

The Contractor shall disclose, at the tender stage, all sub-contractors they intend to use as part of the contract works. The Contractor shall not sub-contract any work to any party without the prior written consent of QUU. It shall remain the Contractors' responsibility to audit and co-ordinate the performance of their sub-contractor with the results to be disclosed to QUU. All requirements applicable to the Contractor are applicable to their Sub-contractors.

2.3 Contractor Exceptions

The Contractor shall be responsible to submit, together with the Tender, a list of deviations or exceptions to this Specification. In the absence of any exceptions, it will be construed that the Contractor fully complies with this Specification.

2.4 Order of Precedence

In the event of any conflict arising between this Specification and other documents listed herein, refer comments to QUU for clarification before the works commences. The order of precedence that applies is as follows:-

1. The Contract or Purchase Order Scope or Work
2. Project Data Sheets
3. This specification and other QUU technical specifications
4. Project Drawings
5. Standards, Codes and Regulations

3. PLC PROJECT IMPLEMENTATION REQUIREMENTS

3.1 PLC Programming Software Licenses

PLC programming software version requirements are detailed in the respective library specification. The Contractor shall confirm with QUU the current software licence version applicable to the project before commencing works. Refer TMS1202 for further details on this topic.

3.2 Project Team and Reporting Structure

Refer TMS1202

3.3 Project Schedule

Refer TMS1202

3.4 Contractor's Previous Experience

Refer TMS1202

3.5 Design Documentation

Refer TMS1202

3.6 Supply of Hardware

Refer TMS1202

3.7 PLC Software Development

Refer TMS1202 and TMS849

3.8 Equipment Installation

Refer TMS1200 and TMS1201

3.9 Testing and Commissioning

Refer TMS1202

3.10 Training – Operator and Technical

Refer TMS1202

3.11 Operation and Maintenance Manuals

Refer TMS1202

3.12 Defects Liability and Warranty

Refer TMS1202

4. PLC HARDWARE

The Contractor shall confirm the PLC hardware requirements that are specific to the site or project requirements.

Refer to TMS1151 Preferred Equipment List – Control Systems for specification of the hardware requirements.

For existing sites, new PLC hardware selection should be consistent with whichever manufacturer predominates at that site.

Refer to TMS1202 for spare I/O capacity for each I/O type used. In the case where multiple PLCs or remote racks are distributed across the functional areas of the plant, then each functional area shall provide 20% spare I/O capacity.

5. PROGRAM NAMING AND REVISION CONTROL

Program naming shall be according to the following format:

<Site_ID>PLCnn_YYYY-MM-DD

where:

<Site_ID>	The short representation of the Site ID, for example ST018 for Luggage Point
nn	The PLC number on site
YYYY-MM-DD	Date

Example:

ST018PLC01_2012-05-07 Luggage Point (ST018) PLC1 on 07/05/2012

The code revision history must be incorporated as a comment at the beginning of the program.

It must include:

- Revision (Major.Minor)
- Date
- Author
- Description of changes made

The major revision number must be incremented upon a PLC upgrade, or commissioning of a new section of plant.

Otherwise, a minor revision number must be incremented for each quantity of minor works, such as a single access to make amendments, or a single day's amendments.

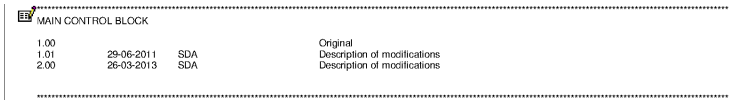

Figure 1 - Revison History on a GE Proficy PLC

```
This section of code provide the generic functions for the PLC code.

Version 1.0 10/01/2005 DJH Initial Release
Version 1.1 10/01/2006 DJH Included PLC Time-Of-Day Code
Version 1.2 22/03/2005 DJS Included STO_DATE for 2AM Time Synchronisation
Version 1.3 24/04/2005 NPG Modified driving of "Midnight" pulse
```

Figure 2 - Revison History on a Siemans PLC

6. PROGRAMMING LANGUAGES

6.1 General

PLC code shall be developed using QUU Libraries, which are based on function blocks called from ladder code subroutines. General, code shall only be written in Ladder Logic using object orientated formatting and data structures.

The use of Function Block Diagrams (FBD) is only allowed under special conditions and at the discretion of QUU. Structured Text (ST) and Structured Control Language (SCL) shall be restricted to function blocks. Function blocks are to be used for generic code only. Example code, used on QUU operational sites can be supplied on request. Sequential Function Chart may be used for sequencing purposes.

The allowable programming languages are defined in:

IEC 61131-3 "Programmable controllers - Part 3: Programming languages"

IEC 61131-3 Language	Type	Approval for use	Rockwell Logix Supported	GE Proficy Supported	Siemens Step 7 Supported	Siemens TIA Supported
Instruction List (IL)	Text	Approved	No	No	Yes (STL)	TBA
Structured Text (ST)	Text	Restricted	Yes	Yes	Yes (SCL add-on)	TBA
Ladder Diagram (LD)	Graphical	Approved	Yes	Yes	Yes	TBA
Sequential Function Chart (SFC)	Graphical	Approved	Yes	No	Yes (GRAPH) (add-on)	TBA
Function Block Diagram (FBD)	Graphical	Not Approved	Yes	Yes	Yes	TBA

Table 1 – PLC Programming languages

Notes:

- Functions are a feature of Rockwell logix, GE Proficy and Siemens Step 7, and are approved for use. Such functions may be written as a Ladder Diagram, or in Structured Text, and are then referred to as a Ladder Block or a Structured Text Block. Generically they may be referred to as a function block, but this is different from the "Function Block Diagram" graphical language.
- Structured Control Language (SCL) is the Siemens STEP 7 implementation of the Structured Text programming language as defined in the IEC 61131 standard.
- Statement List (STL) corresponds to the "Instruction List" language defined in IEC 61131-3.
- Sequential Function Chart (GRAPH) corresponds to "Sequential Function Chart (SFC)" language defined in IEC 61131-3.

6.2 Sequential Logic

A structured high-level control programming language such as Function Block Diagramming (FBD) and Sequential Flow Charts (SFCs) shall be used and shall provide the necessary facilities for real-time control of sequential processes. Simultaneous execution of multiple sequences shall be required. As part of SFCs (for some elements of SFCs), other complying programming languages may be used.

6.3 Drive Interlocking Logic

A structured high-level control programming language such as FBD shall be used and shall provide the necessary facilities for real-time interlocking of drives, on/off valves and process equipment. Generally, SFCs shall not be used for interlocking.

Interlocking is a crucial aspect of control systems implementation and needs to be defined thoroughly by the Contractor. The triggering of excessive alarms due to interlocks that are not visible on SCADA is not accepted. All the faults/interlocks shall be visible to the operator.

Drive blocks shall be programmed with separate block inputs between Interlocks, Remote-Manual Permissives and Remote-Automatic Permissives. All blocks provided shall not be write-protected and shall be fully annotated with source code provided.

6.4 Regulatory Logic

A structured high-level control programming language such as FBD shall be used and shall provide the necessary facilities for real-time regulatory control and monitoring of the process.

6.5 Calculations

Function Blocks (FB), Structured Text (ST) or Structured Control Language (SCL) shall be used for calculations, whichever is more appropriate. Simple calculations are easily expressed using Function Blocks. Calculations involving detailed formulae, array references or iterative processes are more efficiently expressed using Structured Text. The results of calculations are to be verified using sample values with the predicted results. Where complex calculations are to be reused, the calculation shall be encapsulated and then called in a subroutine or object orientated, modular block. All blocks shall not be write-protected and shall be fully annotated with source code provided.

7. PROGRAM DOCUMENTATION

Comments must be added to the program to segregate logic into functional areas of operation to enhance readability

7.1 Tags

All tags used within the PLC must be given useful tag names and comments that conform to the relevant standard.

7.1.1 Tag Names

Detail on the tag naming that should be implemented can be found in the PLC and HMI Tag Naming Standard which is currently under development. The following section specifies the different types of tags that exist as well as the rules and structures that shall be implemented by the Contractor.

The standard tag name format is as follows:

<EllipseID>TtXxxxx (Ellipse), or

DddddEeeeeTtXxxxx (non - Ellipse)

Where:

Field	Chars	Description	Example
<EllipseID>	18	Ellipse Equipment ID	Old Ellipse format: SP381-PUS-0111-002 SP381 Pump Set 2 SP381-RTU-1090-001 SP381 RTU 1 New Ellipse Format: SP381-0111-PUS-002 SP381 Pump Set 2 SP381-1090-RTU-001 SP381 RTU 1 Note: The '-' is removed to reduce the tag length.
Ddddd	5	Device/Area Identification	AER02 Aerator No.2 BFP01 Belt Filter Press No.1
Eeeee	5	Equipment Identification	BLO01 Blower No.1 CMP02 Compressor No.2
Tt	2	Tag Type	HMI and PLC examples: ds Digital HMI Status Bit dc Digital HMI Command Bit as Analog HMI Analog Status (Value) ac Analog HMI Analog Command (Set point) Additional PLC examples: di Digital Digital input gb Digital General bit gr INT General register tm INT Timer
Xxxxx	<= 15	Mnemonic	ThermalOLoad

Table 2 – PLC Tag Description Details

The use of upper and lower case to split up the tag name is permitted. eg.

ST032Tnk01Vlv01dsFault.

Device and equipment identification may be split up to include more information if required. eg.

BR3Z4 Bioreactor 3 Zone 4, or
AEREQ Aeration Equipment

Tag name and description for digital tags shall correspond to the 'ON' (1) energised state (eg. VLV01dsOpen).

Under no circumstance shall raw inputs on the PLC be referred to directly by the SCADA. All PLC I/O must be mapped to 'ds' or 'as' registers for use by the SCADA. The Contractor shall define all tag types used.

Table 3 – Sample Device ID's

Device ID	Description
AER02	Aerator No.2
BFP01	Belt Filter Press No.1
CF004	Clarifier No.4
DIG01	Digestor No. 1
FSP01	Final Settling Pond No. 1
FST01	Final Settling Tank No.1
INL02	Inlet No. 2
IRR01	Irrigation
RAS01	Return Activated Sludge No. 1
SLT01	Sludge Thickener No. 1
WAS02	Waste Activated Sludge System No.2

Table 4 – Sample Equipment ID's

Equipment ID	Description
BLO01	Blower No.1
CMP02	Compressor No.2
CN001	Conveyor No.1
FLW01	Flow meter No.1
MX003	Mixer No.3
PMP01	Pump No.5
TNK02	Tank No.1
VLV01	Valve No.1

Table 5 – Sample SCADA Tag Types

Tag Type	Data Type	Description
Ds	Digital	SCADA Status Bit
Dc	Digital	SCADA Command Bit
As	Analog	SCADA Analog Status (Value)
Ac	Analog	SCADA Analog Command (Set point)

7.1.2 Tag Comments

Tag comments must be created to detail the association or purpose of every tag. When a tag is closely associated with a field device, the tag comment must include:

Plant section name

Device name

If there is any ambiguity in the interpretation of the tag, the comment should include detail to clarify exactly what the tag represents.

Wherever possible, descriptors of variables that are part of a structure must include the description of the structure, not just the variable itself. For example, the description of the variable PU1234567.acAuto should be "RAS Pump 1 : Auto Mode", not just "Auto Mode". Hardware I/O tags shall include the physical Rack/Slot/Point in the tag comment.

7.2 Rung Comments

Comments must be added to individual rungs, networks or sections of code to enhance the readability of the code.

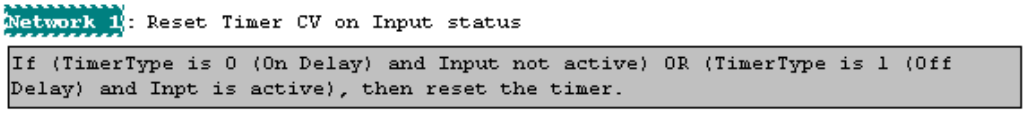

Figure 3 - PLC netwrok comments on a Siemans PLC

7.2.1 Sequence Steps

Comments must be inserted at every definable state and transition to provide a description of the purpose of the state and the conditions required to enable the transition.

7.2.2 Complex Logic

Comments must be inserted to provide information about and an explanation of complex logic.

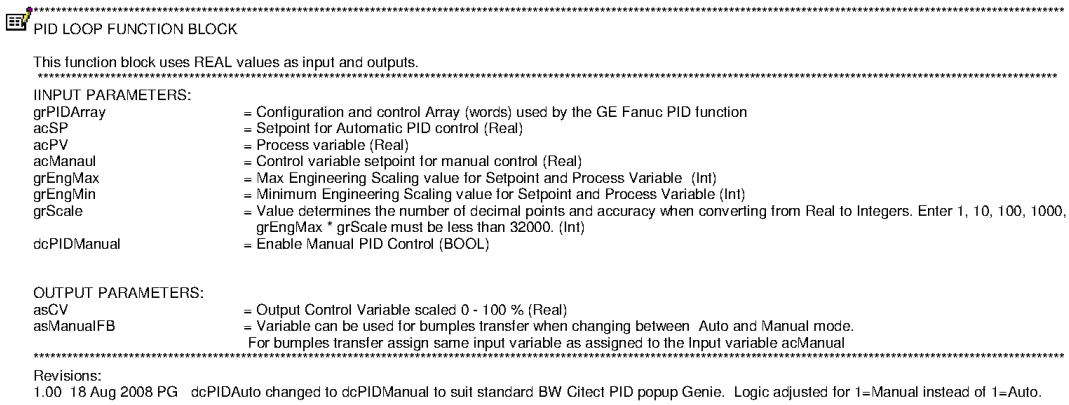

Figure 4 - Complex logic comments on a GE PLC

```
FC501 : Generic Timer Function

Generic timer function used when not wanting to use Siemens standard timers.

Input (BOOL)      - Timer input condition
Pulse (BOOL)      - Time base pulse input (typically 1 second pulse)
Timer Type (INT)  - 0 = On delay timer, 1 = Off Delay Timer
TimerSP (INT)     - Timer Setpoint as a multiple of the Time Base pulse period
TimerCV (INT)     - Timer Counter Variable for the timer counter storage
Output (BOOL)     - Timer Output as per functionality below:

On Delay:  If Input is on for the time period TimerSP, then the output will
           turn on.  If the Input is off, the output will turn off.

Off Delay: If the Input is on, the output will be turned on.  If the input is
           turned off, the output will remain on for the time period TimerSP.

Version 1.0 - 25/11/2004 DJH
Version 1.1 - 28/12/2004 DJH Modification to Off Delay Function
```

Figure 5 - A Complex logic comments on a Siemans PLC

7.2.3 Calculations

For all complex calculations, develop Add-on Instructions (Global function blocks). This type of add-on instruction can be developed using structured text programming.

All complex calculations done over multiple rungs must be preceded by comments providing a detailed overview of the following code.

```
POLY FLOW DETERMINATION.

The Poly Flow is determined according to the Following Equation

Feed Flow (kl/Hr) * Feed Concentration (mg/L) * Poly Dosage Rate (kg/Tonne)
_____________________________________________________
Poly Batch Concentration (mg / L)

@ a Typical Feed Flow 9 l/s
                                  = 9 L/s = 30kL/Hr
                                  = 30 kl/Hr @ a Feed Concentration of 3% = 30000mg / L  (30 g/L)
                                  = 970 kg /Hour of Solids

@ a Poly Dosing Rate of 3 kg/Tonne
                                  = 970/1000 * 3 = 2.9 kg/Hour of Poly Required

@ a Poly Concentration of 0.18%  1800 mg/Kg
                                  = 2.9 kg/Hour / (0.18/100)
                                  =1620 L / min
                                  = 27 L/min
```

Figure 6 - Complex Calcualation Description as comments on GE PLC

```
Network 5 : n-Length Filtering

The RawValue is filtered by performing an n-length filtering algorithm, with
the length in seconds being defined in the FilterLengthSP setpoint.   The
filtering is performed by the execution of the following equation every one
second:

Value = ((Value * (FilterLengthSP - 1)) + RawValue) / FilterLengthSP

If filtering is not required, then the FilterLengthSP should be set to 1.  This
will update Value every 1 second with the RawValue.

If filter length is zero, then the Raw Value is copied direct to the Value

DBD16 = acFilterLength
DBD20 = asRawValue
```

Figure 7 - Complex Calcualation Description as comments on Siemans PLC

All rungs that contain calculations must include a detailed comment detailing the formula being used.

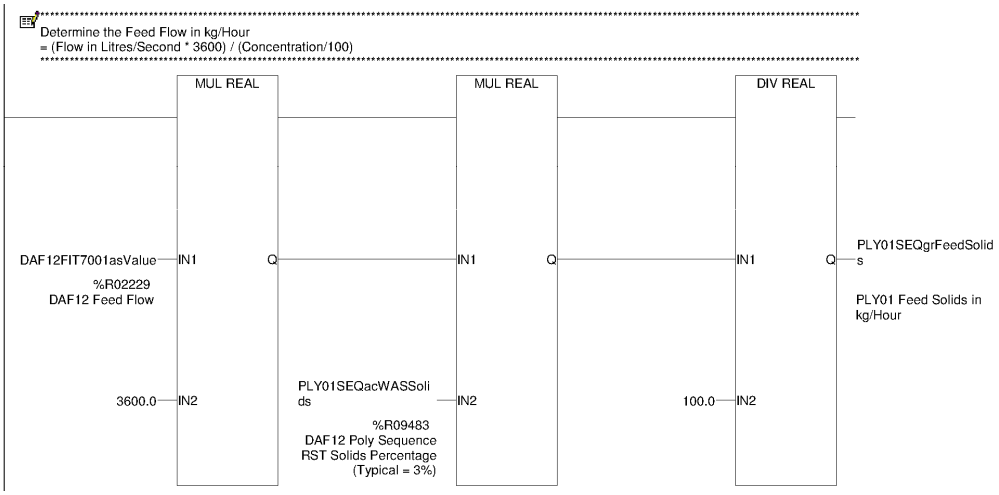

Figure 8 - Rung commenmts on a GE PLC

7.3 PROGRAMMING REQUIREMENTS

Multiple output control in one rung of logic is not to be used. Each output should have its own dedicated rung of logic.

Except for timer and counters, embedded constant values in ladder logic should be avoided. Program constants should be assigned to data block.

Bits which are set by any SCADA system, from outside the PLC software directly controlling the device, should be reset upon use by the PLC software concerned.

All Analogue values are to be scaled to engineering unit values (x1, x10 or x100 for the purposes of resolution) in the PLC using a scaling function block.

The program to be installed on site should take up no more than 80% of the capacity of the CPU Logic I/O or Variables). This is to allow for future modifications or additions that may be necessary on site.

All PLC physical inputs will be mapped to memory words & bits to use in PLC logic routines/networks. All PLC physical outputs should not be used in the logic routines. A logic routine should be created to map internal memory words & bits to PLC physical outputs.

For Siemens Step 7 PLCs Digital inputs will need to be mapped to "M" memory bit addresses, a typical memory address ("M0.3") a digital I/O will get mapped. Analogue inputs will need to be mapped to a "MD" memory double word addresses, a typical analogue memory address ("MD2164")an analogue point will get mapped.

.

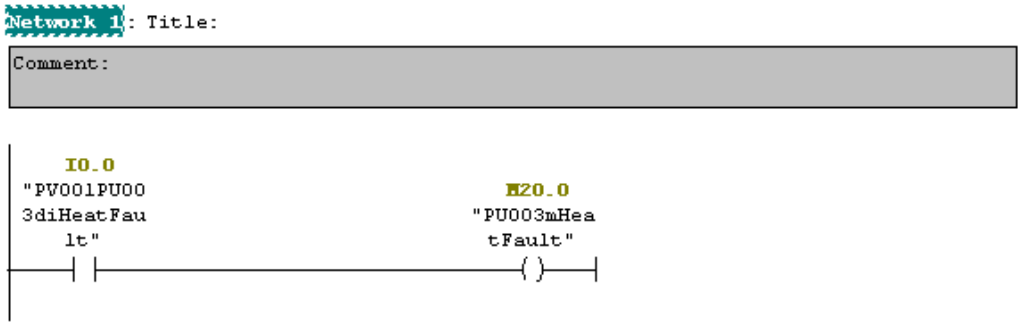

Figure 9 - Digital input mapping to a Memory Address in a Siemans PLC

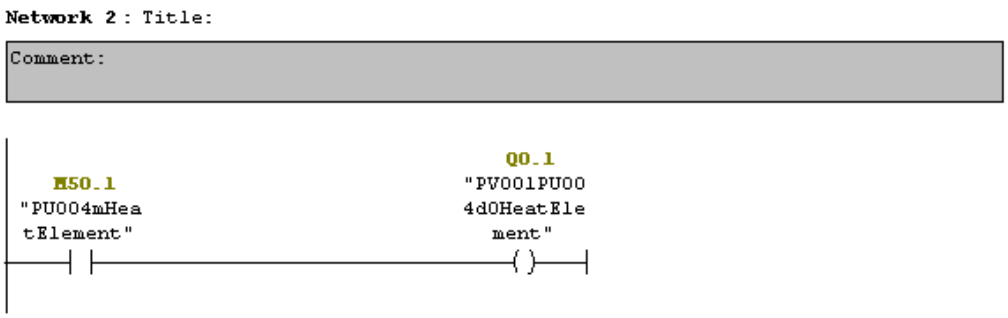

Figure 10 - Mapping a Memory Address to a digital output in a Siemans PLC

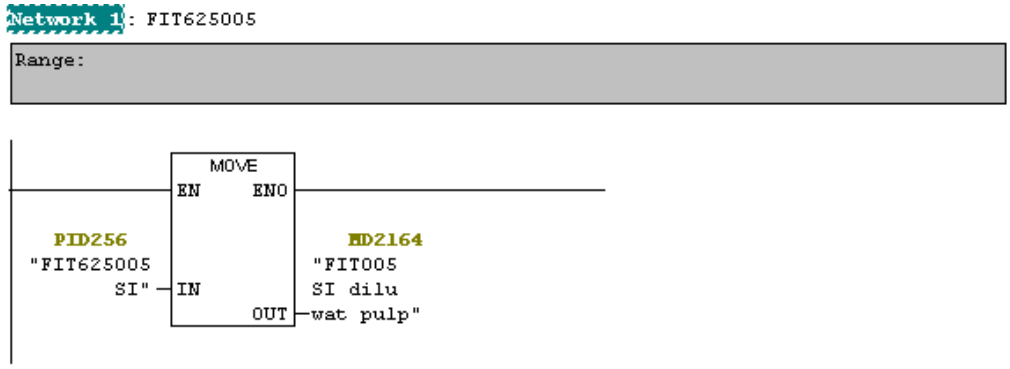

Figure 11 - Ananlogue input mapping to a Memory Address in a Siemans PLC

8. PROGRAMMING FOR FAILSAFE CONTROL

The control logic shall de-energise the relevant sections of the plant upon the following conditions:

Equipment failure, such as:

- PLC processor, rack or module failure
- PLC to PLC communication failure where this transfers equipment interlocks

Power failure, such as:

- Mains power failure
- Control power supply failure

The de-energised state shall be deemed the safe state.

The PLC must control the plant safely without reliance upon the SCADA/HMI.

Any critical safety inputs/signals like a high-level switch input need be 'on' or logical '1' state in healthy state. Any tag or symbol description state should be 'on' or logical '1' state like a tag for pump fault "dsPumpFault" will need to be 'on' during a fault.

9. LIBRARY USE

9.1 General

During the development of PLC code, priority must be given to the reuse of program blocks copied from the "PLC standard code library".

Prior to any program block modification or creation, authorisation must be gained from QUU. This is to ensure consistent and efficient support of the library from project to project, determining whether:

- A standard program block is reused (without change to the library)
- A standard program block is reused with the addition of logic surrounding it to modify the overall effect (without change to the library)
- A standard program block is modified (with update to the library)
- A new program block is created (with possible addition to the library)

All modified or new program blocks must be submitted to QUU for review, approval, and at the discretion of QUU, added to the PLC standard code library.

If a standard block is approved for modification, either:

- The minor version number shall be incremented.
    - backward compatibility must be retained

or

- A new block with a new name is created from the existing standard block.
    - The existing version history must be retained, but must be marked as historical and reference the existing standard block.
    - The version number shall recommence at 0.1.

Modifications of standard blocks shall conform as closely as possible to the existing code structure.

10. SEQUENCE PROGRAMMING

10.1 General

In the event that a number of control actions are necessary, in sequence, a state sequencer shall be implemented.

Each sequence must be implemented in its own program block.

A state sequencer may only be in one state at a time. Each state shall be defined by a unique integer number. Each possible sequence state must also activate an associated unique digital flag. These digital flags along with the conditions required must be tested to drive the transitions between states.

Transition between states must be controlled by conditions comprising device or process conditions, and / or timing.

A sequence always has an idle state and may have a completion state, though it is conceivable that there may be requirement for a sequence with no clearly definable completion state.

The idle state shall be numbered as the 0 state, and shall be the default state adopted upon PLC start-up, and must drive no device activations.

The sequence is considered to be running when the sequence is in a state other than the idle state.

The sequence shall commence running upon the activation of an explicitly defined sequence start interlock and transition conditions.

The explicitly defined sequence run interlock is required to enable the sequence to run following start-up. If the run interlock is lost, the sequence must transition to an appropriate shutdown state or the idle state.

Once a sequence has transitioned to the completion state, it must automatically transition back to the idle state. This transition shall be controlled by a timer of length adequate to support visual feedback of the completion state to the Operator.

It is usual practice to number the states in sequence, commencing from 0, then 1, 2, 3 etc. Though this is recommended, this is not mandatory, except for the 0 (idle) state. Provision must be made to allow insertion of additional steps in future if required.

Each sequence step should have a timeout alarm, if the step is taking more than the specified time a timeout alarm will get generated and if safe transition to the 0(idle) state.

Where a sequence state drives the plant, the state must drive clearly defined control actions and setpoints, which are then utilised by the device control program blocks. The sequence program block must not control devices directly.

Where more than one sequence must interact, a master sequence must be created to control multiple sub sequences. Only when all these sub sequences have completed, the master sequence may complete.

Upon Operator initiation, a sequence must fully automate the plant start-up. Plant shutdown must also be fully automated.

10.2 Sequence Example

This is an example of a simple sequence, of acceptable form, written in ladder logic.

Examples of acceptable associated HMI indications are also shown.

Each of the example conditions may be replaced by one or more conditions. If no real conditions are required then a system bit (*ALWAYS_ON* or *ALWAYS_OFF*) should be used.

10.2.1 Start Permissive

Figure 12 - Typical logic for Start permisives conditions of a GE PLC

All interlock conditions as well as the resultant sequence start permissive, must be shown on the HMI:

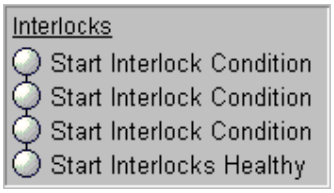

Figure 13 - Start permisives conditions displayed on SCADA

10.2.2 Running Permissive

The start and running permissive may be identical in which case dsCondition4 would be dsStartIlock.

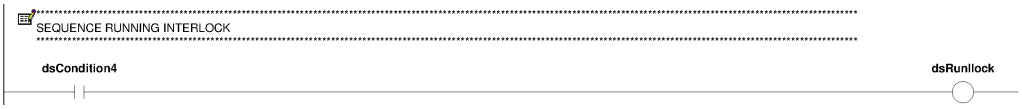

Figure 14 - Typical logic for Run permisives conditions of a GE PLC

All interlock conditions as well as the resultant sequence run permissive, must be shown on the SCADA:

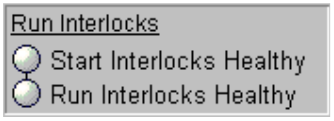

Figure 15 - Run permisives conditions displayed on SCADA

10.2.3 Sequence steps timers and fault handling

The sequence should contain timers with user defined pre-set time to determine if a step is taking too long and when a step doesn't transition to next step. When a step transition fails, the sequence should safely transition to the shutdown part of the sequence or an idle step if applicable.

10.2.4 Sequence stopping and pausing

There should be an option for the operator to pause the sequence. The operator should have the ability do an Emergency or controlled shutdown of the sequence from all the steps.

10.2.5 Sequence steps example

All sequence states as well as state permissive conditions must be shown on the HMI:

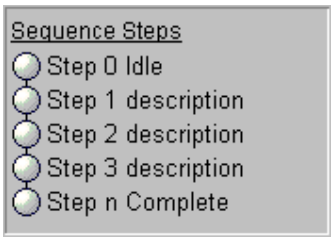

Figure 16 - Sequence step displayed on SCADA

10.2.6 Idle step example

The idle state is to be adopted upon PLC startup, or when the sequence is active and the run interlock is lost.

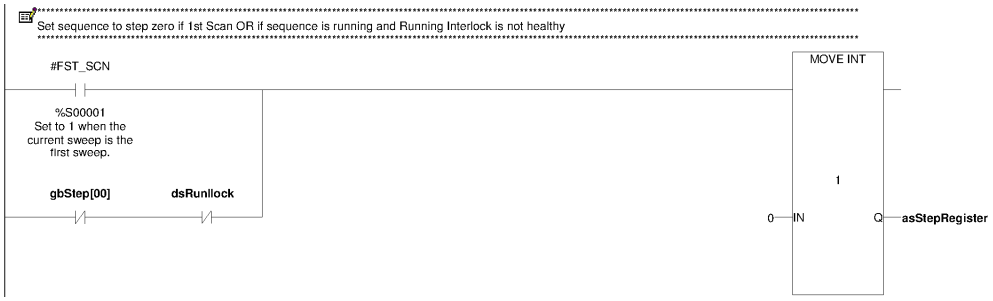

Figure 17 - Typical logic for sequence idle step of GE PLC

10.2.8 Branching logic example

This example shows a transition from step 0 to either step 1 or step 3.

The branching logic must be implemented separately, to drive a state which is then tested by each branch.

Figure 18 - Typical branching logic of GE PLC

Figure 19 - Typical logic for serquence stop step of GE PLC

10.2.9 State transition examples

This example shows a transition from step 1 to step 2 based upon a condition.

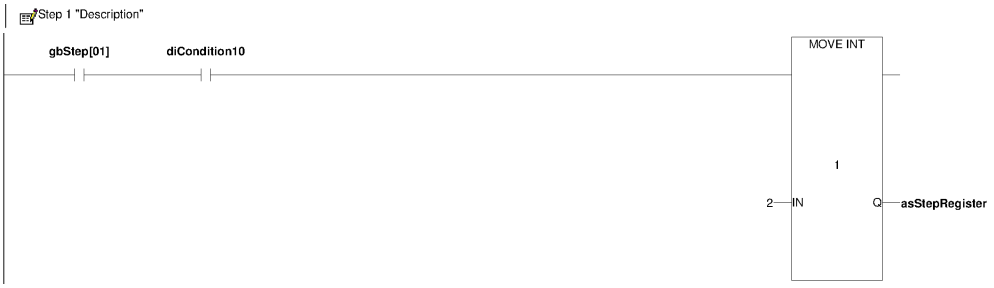

Figure 20 - Typical logic for sequence step 1 for GE PLC

The following example shows a timed transition from step 2 to step 3.

All transition timers (except for those which are only for visual feedback) are critical process parameters, and must therefore be stored in non-volatile memory. These must also be settable by the operators.

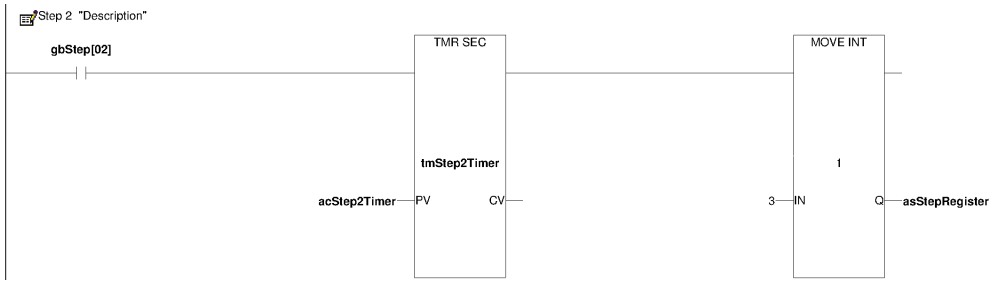

Figure 21 - Typical logic for Sequence step 2 activetransition timer on GE PLC

10.2.10 Last step example

Step n to step 0. Once the sequence has transitioned to the completion state, it must transition back to the idle state. This example uses a 2 second timer before transitioning to the idle state.

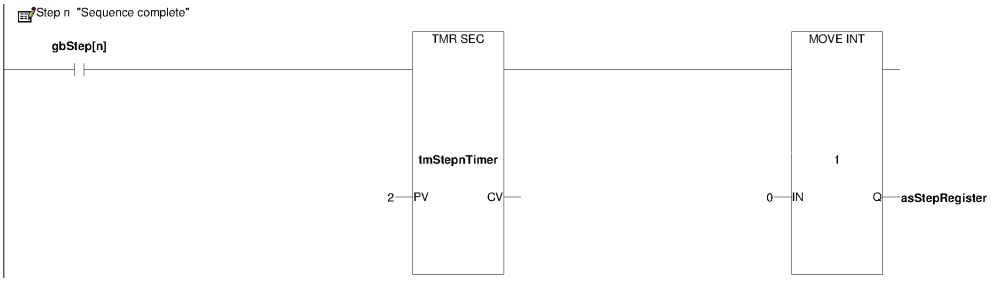

Figure 22 - Typical logic for Sequence step n activetransition timer on GE PLC

10.2.11 Sequence state flags

Each possible state number must also activate an associated unique digital flags. This example sets bits within a word to represent the current state.

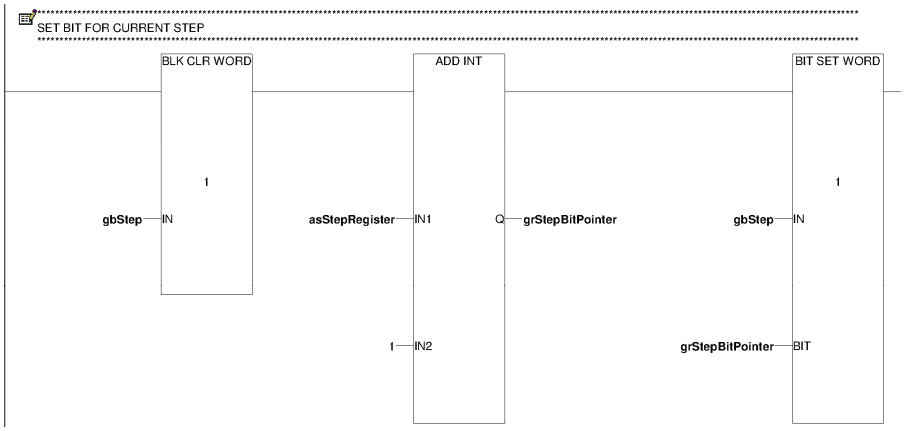

Figure 23 - Logic sequence step active indication Bit set

10.2.12 Activations example

Where a sequence state drives the plant, the state flag must be used to drive clearly defined control activations, which must be utilised by the device control program blocks. These activations should provide separation from between the sequence and the device control meaning that changes required on either side of the activation would only require changes to the code in a single location.

Step 1 Activations

Figure 24 - Sequence Step 1 indication to SCADA

10.2.13 Sequence running indication example

The Sequence Running indication shall be activated when the sequence is not in the idle state.

Figure 25 - Sequence Running Rung

The Sequence Fault indication could be driven by step duration timeouts or other relevant conditions.

11. GENERAL REQUIREMENTS

11.1 Parameter Initialisation

Process parameters which are critical for plant operation must be:

- stored in non-volatile memory, and
- initialised with hard coded values on the first scan if those value are zero

These include, as a minimum, the following:

- Analog scaling ranges and filtering parameters
- Alarm set points
- Process set points
- PID parameters
- Mode selection

The hard coded values should be determined during commissioning and updated as required or upon recommissioning of the plant.

11.2 PLC Start-up

To ensure plant stability at PLC start-up (eg. after recovery from power failure), timers must be implemented to delay the commencement of the process to allow plant stabilisation, and to inhibit nuisance alarms until such stabilisation.

A staged start must be implemented for equipment (eg. motors) which consume a significant peak inrush current during start, and which may trip High Voltage feeds. A suitable delay must be introduced between the start of each such item of equipment to minimize the plant's peak power consumption.

11.3 Load Shedding

If applicable, load shedding must be implemented to inhibit running of non-critical motors while the site is running on generator power.

11.4 Digital[GD1] Alarm Processing

All digital alarms must be processed in a consistent manner prior to their use elsewhere in the program.

11.5 Analog Input Processing

All analogue inputs must be processed in a consistent manner prior to their use elsewhere in the program.

11.6 Analog Output Processing

All analogue outputs must be processed in a consistent manner prior to their passing to the field.

11.7 Drive Standard Logic

For each type of drive, standard drive logic for repeat use throughout the project shall be developed where not provided by QUU. The use of standardised logic embedded into standard user defined function blocks is preferred, providing this is supported by the selected control system.

Typically the drive function block shall contain the following logic:-

- Mapping of Inputs
- Drive Availability logic
- Interlocks
- Auto Run Command
- Specific device trip/alarm logic

11.8 Drive Modes of Operation

Drives shall generally have three modes of operation including LOCAL, OFF and REMOTE. The site specific functional specification shall outline the modes of operation. A consistent method of display and changing the drive mode on plant graphics shall be implemented. The following control modes shall be supported for controlled devices.

MCC/LCP selector	Local	Off	Remote		
HMI selection	Irrelevant	Irrelevant	Manual	Auto	Out of Service
Control	Controlled by physical pushbuttons	Hardwired de-energised	According to HMI control buttons	According to plant sequences	Device unavailable for remote operation

Table 6 – Control Modes Information

Emergency Stop pushbuttons de-energise the device in all modes.

11.8.1 Local Mode

The operator can select this mode by operating the "Local/Off/Remote" selector switch at the equipment site to "Local". The selection of this mode is indicated on SCADA. The operator can start and stop (or open and close) the equipment at the equipment site independently of the SCADA system while in this mode.

The drive will stop if the operating mode is switched to Remote while running in Local. Upon return to Remote, the drive will resume the mode it was in before it was switched to Local. This means that a remote manual start will be required if the mode is Remote/Manual, or if in Remote/Auto, the drive will start if the automatic logic requires it to.

11.8.2 Off Mode

This mode may be selected at the equipment site by selection the "Local/Off/Remote" selector switch to "Off". When equipment is placed in the Off

Mode, the equipment cannot be operated either manually or automatically by the control system, and no alarms are generated for this equipment.

Controlled equipment is defined as being available if the equipment is not in the Off Mode and it is not in a Fault state. This mode is generally used for maintenance purposes.

11.8.3 Remote Manual Mode

The Remote Manual Mode can be selected on SCADA when the equipment is in remote by selecting the "Manual" button. When equipment is in Manual Mode, it is remotely controlled from SCADA by selecting Start and Stop (or Open and Close) buttons.

The remote manual control of the equipment is only possible if the following conditions are met:

- Manual Interlock Condition – the equipment can be started if the defined manual interlock conditions are met
- Fault Condition – The equipment can be started if the fault condition is not active

If the equipment fails to start or stop (or open or close) within a specific time after the command is issued, then the equipment will be put into a Fault state, and an alarm will be raised on SCADA.

If equipment is running in Remote Automatic Mode, and Remote Manual Mode is selected, then the equipment will continue to run and it will adjust its speed or position (if applicable) to the Manual Setpoint set on SCADA.

11.8.4 Remote Automatic Mode

The Remote Automatic Mode can be selected on SCADA by selecting the "Automatic" button. This mode is the "normal" mode of control for all devices within the plant. When equipment is placed in the Automatic Mode, the control of the equipment is defined by a series of conditions, set points, and delay times as follows:

- Automatic Start/Open Condition – this condition is logical expression controlled by the equipment's automatic sequence, that when true starts/opens the equipment
- Automatic Interlock Condition – the equipment can only be started if the defined automatic interlock conditions are met.
- Fault Condition – The equipment can only be started if the fault condition is not active.

If the equipment fails to start or stop (or open or close) within a specific time of the automatic commands, then the equipment will be put into a Fault state, and an alarm will be raised on SCADA.

Automatic sequences are groups of equipment that run together under automatic control, according to a defined series of steps, loops, times, and interlocks. The normal operation of these sequences is that they are constantly in running mode, but the facility exists for the operator to start and stop the operation of the sequence. Each sequence has "Start" and "Stop" control, and a Running/Stopped feedback to show the status of the sequence.

All setpoints and times required for the logical equations for automatic sequence control are adjustable from SCADA. The operator input for setpoints is limited to a safe operating range defined for each setpoint.

If equipment is running in Remote Manual Mode, and Remote Automatic Mode is selected, then the equipment will run as per automatic conditions defined above.

11.8.5 Remote Out of Service Mode

The Remote Out of Service Mode can be selected on SCADA by selecting the "Out of Service" button. This mode is used to indicate that the equipment is out of service on the SCADA system.

11.9 Drive Duty Operation

11.9.1 Duty Assist Control

Equipment can be in the Duty control configuration if the process requirement is at times not able to be provided by a single item of equipment operating, (i.e. requires the operation of 2 or more pieces of equipment). Individual Start and Stop conditions are defined for each piece of equipment required to operate.

If the Duty(n) equipment is unavailable (Faulted, Disabled, or Manual Stop), then the Duty(n+1) equipment is to continue operating with the duty equipment parameters. The duty changeover is operator-selectable remotely from the SCADA as:

- 1 = Duty A, 2 = Duty B, etc.
- 2 = Duty A, 1 = Duty B, etc.
- Auto changeover

In Auto changeover mode, for equipment that stops and starts, the duty will change every time the Duty A equipment stopped stops, and for equipment that runs continuously, the duty will change when the duty equipment has been running continuously for 24 hours.

If the Duty A equipment fails, then the Duty B equipment shall immediately become the new Duty A equipment if available.

11.9.2 Duty/ Standby Control

Pumps may be in the Duty/Standby Control configuration if the process requirement can be provided by only one pump operating. Start and Stop conditions are defined for the operation of the Duty Pump.

If the Duty Pump is unavailable (Faulted, Disabled, or Manual Stop), then the Standby Pump is to operate in place of the Duty Pump. Pump Duty is operator-selectable remotely from the SCADA as Duty, Standby, and Alternate.

- 1=Duty, 2=Standby
- 2=Duty, 1=Standby
- Auto changeover

In Auto changeover mode, for equipment that stops and starts, the duty will change every time the duty equipment is stopped, and for equipment that runs continuously, the duty will change when the duty equipment has been running continuously for 24 hours.

If the Duty equipment fails, then duty shall failover to the Standby equipment and it shall become the new Duty equipment.

11.10 Drive Alarm Management

Unless otherwise stated, the occurrence of any drive alarms will cause the control system to remove the run signal to the drive.

11.10.1 Drive Status Indication

For each drive, at least the following states shall be provided for operator display purposes and for other logic:

- Auto Selected
- Remote Manual Selected
- Drive Running
- General Fault (NB. Could be Thermal Overload or other fault)
- Stopped, Not Ready to Start
- Interlocked
- Stopped, Ready to Start.

For each drive, at least the following states shall be displayed on the custom plant graphics.

Status	Functionality
Tripped	Latched on by drive in the run state being tripped by a safety or equipment interlock. Reset by a restart of the drive.
Interlocked	Indicated when drive is prevented from running in Remote mode by a process interlock. Different types of interlocks shall be displayed including the individual interlocks. Permissive and interlocks shall be clearly displayed and differences able to be readily distinguished. The Contractor shall document thoroughly the development of hierarchical logic for device and system interlocks.

11.10.2 Drive Interlock/Permissive

Drive interlock logic shall reflect the specific tripping requirements of the particular drive as well as a 'failed to run' trip and shall be programmed in the standard drive function block.

Drive interlocks shall not cause an alarm to be generated if the drive was not running or starting.

There are three types of drive interlocks and each shall have different functionality as described in the following table.

Interlock	Example	Functionality
Safety Interlock	Stop Button, Thermal Overload, Pull Wire	Hard wired, cannot be over-ridden. Inputs to control system configured for monitoring, alarming, tripping of drive logic.
Equipment Interlock	Seal Failure, Belt Drift, Blocked Chute, Failed to Run, Trip	Inputs to control system configured for monitoring, alarming, tripping of drive logic. Able to be bypassed by configured logic.
Process Interlocks (Auto/Manual Permissives)	Downstream equipment status, Process levels, flows and pressures.	Inputs to control system configured for monitoring, alarming, stopping of drive logic. The drive is stopped until the process condition returns back within its operational range. The drive will then be enabled to operate as required. Able to be bypassed by configured logic. Bypassed in 'MAINTENANCE' Mode. No alarm for drive, only for the process interlock.

11.10.3 Process Interlock/Permissive

A process interlock is also commonly referred to as permissive. A process interlock may cause a drive to stop and an event shall be recorded in the Event Log, however will not generate a drive trip alarm for that drive. This is because the stopping of the drive is a consequence of another event that is subject to the alarming defined for that event.

All new device function blocks shall provide for Remote Manual and Remote Automatic permissive/interlock inputs. The function block inputs are to be ON when the permissive / interlock is healthy.

For example, if drive A is interlocked to stop when drive B is stopped then:

- If drives A is running and drive B stops then no drive trip alarm is generated for drive A.
- If the cause of drive B stopping is a trip then a drive trip alarm is generated for drive B only.

- If the cause of drive B stopping is that the operator stopped it from the drive faceplate, then no alarms are generated for drive A or for drive B.
- If the cause of drive B stopping is an interlock from another drive then a drive trip alarm is generated for that other drive only.

11.11 Drive Run Time Indication

Drive run indication shall be provided for each drive. The duration of run time logged before rollover shall be a minimum of 10 years. The run time shall accumulate with a resolution of one second.

If time accumulators with the above characteristics are not supplied as a standard part of the control system, then a standard function block with a retentive timer and one or more counters shall be implemented.

11.12 Valve Standard Logic

The Contractor shall implement QUU standard valve software in PLC and SCADA where applicable.

For each type of valve controlled by the PLC, standard valve logic for repeat use throughout the project shall be developed where not provided by QUU.

Typically, standard valve control logic types shall include On-Off valves (with various forms of feedback) and motorised valves.

11.12.1 Valve Common Requirements

Logic execution speed shall be appropriately selected for the service.

11.12.2 Valve Mode Selection

Consistent methods of display and changing of control mode similar to drive control shall be developed for all types of valves to be used within a project.

In mode transition if the valve is operating in Remote Manual or Automatic mode, and Local Manual mode is selected, the valve will remain in its current position.

11.13 Valve Operator

The control logic for a valve shall provide the necessary flags for faceplates and pop-up graphic windows that shall be provided for the operator. These displays shall include the following:

- Position Status Indication—Open/Closed/Indeterminate
- Analogue Position Indication—% Open and % Feedback
- Mode Indication—Maintenance/Manual/Auto/(Cascade, Ratio, Computer)
- Fault Control Status Indication—Interlocked, Tripped, Ready for Sequence Control

- Valve Icon
- An Alarms Section shall display the possible alarm conditions for the valve. Existing alarms shall be differentiated from dormant alarms by colour change or highlighting. Alarms to include 'Failed to Open' and 'Failed to Close' indications
- An Interlocks Section shall display the permissive conditions for valve operation
- The current state of each permissive shall be differentiated by colour change or highlighting
- Control buttons for mode control
- Control buttons for manual control.

11.14 Valve Alarm Management

11.14.1 General Valve Alarm Logic

Unless otherwise stated, the control system must attempt to maintain the desired position of the valve regardless of the occurrence of any alarms generated by that valve.

11.14.2 Valves with Open/Close Limit Switches

Valve position feedback (Open and Closed limit switches) shall both be installed and connected into the PLC on every installation unless otherwise specified. The Contractor shall not simulate a valve's position feedback using the valves command signal.

Valves with limit switches shall generate an alarm if they do not achieve their target state within a pre-set individually adjustable time. When a 'failed to open' or 'failed to close' alarm is generated, the target will be maintained unless altered by operator action or other logic that has been specified.

Valves with limit switches shall generate a discrepancy alarm if the limit switches indicate a mutually exclusive state (such as both open and closed indicated).

11.14.3 Valve Device Interlocking

Valve interlock logic shall reflect the specific tripping requirements of the particular valve.

11.14.4 Valve Process Interlocks

A process interlock may cause a valve to change state but will not generate a valve trip alarm for that valve.

This is because the operation of the valve is a consequence of another event that is subject to the alarming defined for that event. This is a similar concept as described in Drive Interlocking—Process Interlocks.

11.15 PID Control

The Contractor shall implement QUU standard PID software blocks in PLC and SCADA configuration where applicable.

Where not provided by QUU the standard logic for PID control shall be developed by the Contractor and shall include the following:-

- Remote Manual control of the final element (eg. pump speed, valve position) shall be through the PID controller
- Setpoint limit clamping
- Output limit configured within the PID controller
- Setpoint Ramp control (only if required)
- Operator selection on SCADA between Cascade and Single Loop control modes
- Bumpless transfer for;
    - Cascade to Single Loop
    - Single to Cascade Loop
    - PID Manual to Auto mode
    - PID Auto to Manual mode
    - System initialisation
- Bumpless transfer during mode selection between two separate PID controllers eg. The operator selects valve control between Flow PID or Pressure PID. The transition between Flow control and Pressure control shall be bumpless
- Prevention of integral windup
- SCADA PID popup windows shall allow for direct engineering parameter manipulation, setpoint by control systems engineers
- Retention of all PID tuning parameters on cold restart of CPU.
- All tuning parameters shall be trended.

11.15.1 Control Loop Diagrams

The Contractor shall provide Control Loop Diagrams for complex control systems. The Control Loop Diagram shall be referred to in the site specific functional specification.

11.15.2 Loop Tuning

Loop tuning results and trends shall be submitted after commissioning for;

- Setpoint Step-Response
- Process Load Disturbance

11.16 Calculations

All calculations shall be carried out in engineering values, not raw values.

Ensure adequate error trapping is used, for example, check for zero value to avoid divide by zero errors before all divisions.

11.17 Reporting

The program shall accumulate certain values and make it available to the HMI for daily reporting. Current day's data shall be accumulated and made available to the HMI as it is being accumulated. At a pre-set time each day, the HMI activates the End of Day (dcEOD) tag in the PLC. Upon the End of Day tag activation, the PLC shall move all current day values into the associated yesterday tags, and shall then reset the End of Day tag.

The following data, as a minimum, shall be accumulated:

Description	Current day tag suffix	Yesterday tag suffix
Drive Run Hours	asHours	asHoursYDay
Drive Run Minutes	asMinutes	asMinutesYDay
Drive Starts	asStarts	asStartsYDay
Valve Operations	asOperations	asOperationsYDay
Flow Totals	asTotal	asTotalYDay
Power Totals	asTotal	asTotalYDay

Table 7 – Run hours & number of starts Information

11.18 PLC Status

All standard PLC status bits and the health of all communications links shall be made available to the HMI.

For example:

- Hardware faults such as PLC Running/Stopped Status and Configuration Errors
- PLC rack faults
- IO card errors
- Communications errors

12. PROGRAM FILE STRUCTURE

12.1 General

At the beginning of the PLC program, it shall contain comment text with the following detail:

- program revision number,
- date of the revision,
- PLC programmer's full name,
- PLC programmers company name and
- summary of major modifications to current revision
- functional specification document number

12.2 Logic Structure

A hierarchical structure that links higher-level modules (such as sequence logic modules) to lower level logic modules (such as those controlling individual drives) shall be defined. This structure will group logic modules for the various plant areas and subsystems in a manner that simplifies the navigation through the configuration logic.

The program shall be structured as a calling hierarchy, "GOTO" and "JUMP" commands must not be used. The following table details the various blocks that should be implemented and how they relate to each other.

Name	Example	Contents	Calls	Note
Main program block		Calls	Administrative program blocks or Section program blocks	First block executed
Administrative program blocks	Generic PLC functions PLC status and faults PLC to PLC comms PLC to device communications Test code	Logic	Library program blocks	
Section program blocks		Calls	Section sequence program block or Device program blocks	First block dedicated to a plant section
Section sequence program block		Logic		
Device program blocks	Control program blocks Status program blocks Sequence program blocks	Logic	Library program blocks	
Library program blocks	bwALARM bwANALOG bwPID	Logic		Copied from "PLC standard code library"

Table 8 – PLC programming file structure

The following is an example implementation (GE) of a calling hierarchy:

- *_MAIN* is the first program block, which is run whenever the PLC is in run mode, and which calls the section control block:
- *a_PreTreatment_Main*, which calls the device control block:
- *FCV2000_PostGritAerationValve, which calls the standard control block e.g. bwALARM*

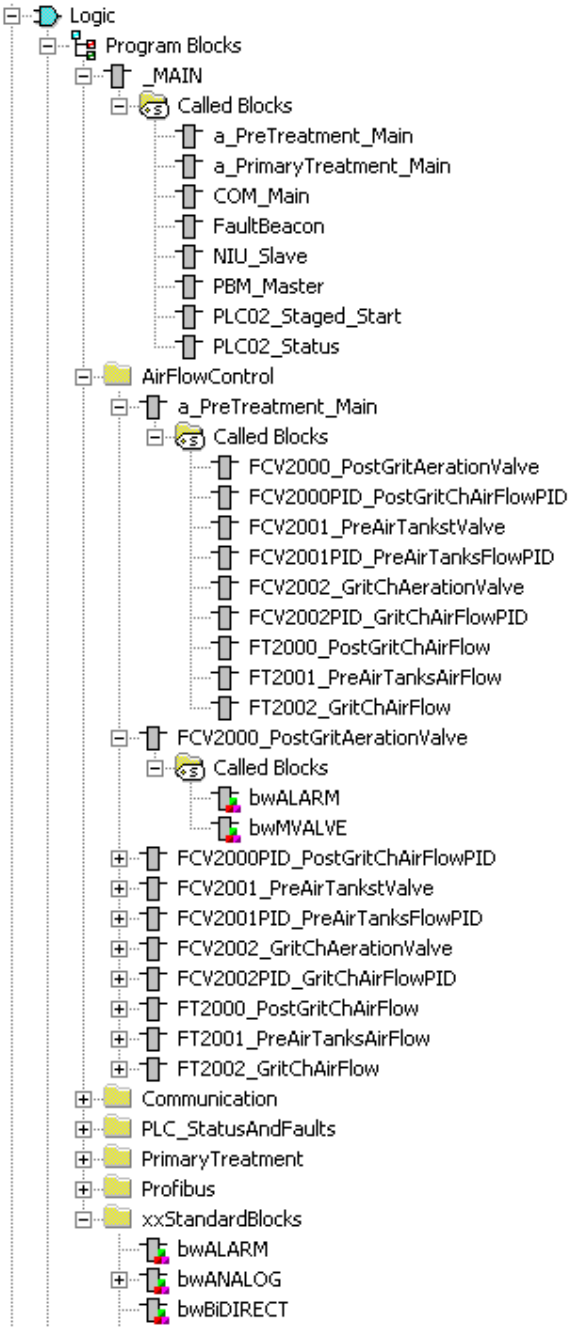

Figure 26 - PLC program sytructre for the GE PLC

12.3 Logic Modules

For PLC programs that are repeated across multiple sites under the same Contract scope or works a standard program shall be developed with site specific setup parameters defined. The compilation of standard programs shall include QUU stakeholders detail review and acceptance of the software.

PLC software code shall be designed as simple, formally structured logic modules, each carrying out clearly defined tasks. Examples of a logic module are the control and interlocking logic for a drive or a PID control loop. A general rule of thumb shall be that any logic module shall contain only one coherent plant system from a commissioning perspective.

Wherever possible, all regulatory and sequence control for a specific system should be grouped together.

Readability of the logic has priority over compactness of coding. An over emphasis on minimisation can result in configuration which is poorly structured and unreadable, and hence difficult to modify. This does not mean that the limits of the configurable resources available should not be considered.

Each module shall minimise the interlocks or signals required from other modules.

For applications that are repeated throughout the project, modules shall be designed to be standard repeatable implementations that have been proven by testing.

Within modules, standard repeatable subsections of code that has been proven by testing shall be utilised wherever possible.

Only one logic module or function block shall write to a particular output field such as a flag, output or other item.

All blocks shall not to be write-protected and shall be fully annotated with source code provided. Password protected or locked code is not accepted.

12.4 Logic Location

Logic shall be configured and executed in the same controller where the I/O for the logic is configured. Peer to peer controller communications shall be minimised. Any peer to peer communications shall be described and documented in the Control Systems Administration Manual. The documentation must detail the consequence of failure of the remote peer data to the local plant.

Where peer to peer communications are used, then logic shall detect that the communicated values are valid and current and shall perform appropriate interlocking, tripping and alarming functions. Analogue watchdog alarms shall be incorporated with all networked PLCs where robust hardware failure detection is not provided. Failure of the watchdog will trigger an invalid alarm. All critical signals shall be relocated to the PLC performing the control function where possible.

12.5 Order of Execution

Logic components shall be arranged into an execution order minimising the number of logic scans for a change in the system to propagate through the logic.

The main routine shall only be used to call individual subroutines. It shall not contain process functionality.

12.6 Order of Appearance

Whenever program logic modules are listed (such as in a hierarchical tree display) they shall be listed in a sensible order, grouped first by plant area, then by category of equipment and then in alphanumeric order.

12.7 Grouping of Modules in Folders

Where the programming system allows modules to be grouped into folders for programmer convenience, modules shall be grouped into folders by plant area. Folders shall be named by area, followed by a descriptive name. There must be a clear distinction between the devices and where they are used within plant control.

12.8 Segregation of Elements

A system of reserving various programming elements, flags and memory blocks based on a functional grouping criteria shall be defined in the revised specification for the selected process control system. For example, flags used for a particular drive or purpose may be grouped together. The objective is to provide structure by setting guidelines if the structure does not inherently exist.

12.9 Signal Monitoring

Whenever possible, signal monitoring functions for alarming and or tripping shall be implemented by any built-in signal monitoring capabilities of the function blocks that also used for monitoring and controlling.

Critical safety interlocks shall not share functionality with process control functions.

13. COMMUNICATIONS

13.1 Peer to Peer Communications

Where peer to peer communications are required, a heartbeat value must be included with each transfer. This value must change with each transfer. The communications may only be deemed to be healthy if the heartbeat value changes at an appropriate rate, and has done so for a predetermined time.

Upon loss of communications, control logic relying upon this communications must default to a safe state.

13.2 Device and Instrument Communications

A program block containing the management of communications to and from devices via instrumentation buses (e.g. Profibus, Modbus, Ethernet etc.) shall be developed for each communications channel used. It shall include all the required initialisation, monitoring, diagnostics and alarming.

13.3 HMI Communications

Communication between the PLC and the HMI is the responsibility of the HMI as the PLC does not rely on this link. If required, the HMI is responsible for monitoring the communications link to determine online / offline status.

14. DEVICE CONTROL

Device logic shall control the operation of drives, valves and other devices based on field inputs and outputs, as well as Operator interaction via the HMI.

The logic shall be structured as follows, as appropriate for each device's functional requirements.

14.1 Available Logic and Interlocks

Logic determining if the device is available to be operated in either local or remote, based on its faults and alarms. The Available bit is equivalent to a run Interlock. This section also contains interlock logic including start, run and auto control interlocks.

14.2 Sequence Activations

Activations from the state sequencers are translated into the device specific flags for automatic control of the device.

14.3 Device Control

The device's functional requirements shall be implemented in this section. Standard library blocks should be used where appropriate, such as those containing common control for a valve or drive. This section will also include any device specific calculations, conversions and accumulators. If real world inputs are required they will be used here including any pre-processing that is required.

14.4 Reset Command

Devices can often be reset from various sources or as part of a group of devices. The logic to map these conditions into the device reset flag is placed in this section.

14.5 Status, Alarms and Faults

The first action required in this section is to set a flag based on status of the alarms and faults relating to the device. Following this will be the logic to trigger the alarms and faults.

14.6 Outputs

Device specific digital and analogue outputs are mapped here including any final filtering or scaling.

14.7 Reset Command Bits

The final logic will clear the digital commands (dc tags) which have been acted upon by the device control logic.