TECHNICAL SPECIFICATION FOR
LCLS-II Cryoplant General Control Software Specification

Document Number: 79120-C6003

Revision History:

Revision	Description of Change	Date
-	Original release	1/31/19

Chris Scanlon
JLab Cryogenics Department
Electrical Engineer

Swapnil Shrishrimal
JLab Cryogenics Department
Electrical Engineer

Robert Norton
JLab Cryogenics Department
Electrical Engineering Manager

Ritendra Bhattacharya
JLab Cryogenics Department
LCLS-II Commissioning Lead

John Hogan
JLab LCLS-II
Cryoplant CAM

Dana Arenius
JLab Cryogenics Department
Principal Cryogenic Engineer

Approved: 3/1/2019; E-Sign ID : 383714; signed by: Ap. 1: R. Norton; Ap. 2: R. Bhattacharya; Ap. 3: D. Arenius; Ap. 4: J. Hogan; C.: S. Shrishrimal; D: A. Barron

Table of Contents

I. Definitions

LCLS-II	Linac Coherent Light Source – II
SLAC	SLAC National Accelerator Laboratory, the purchaser of the equipment mentioned herein
JLAB	Thomas Jefferson National Accelerator Facility
EPICS	Experimental Physics and Industrial Control System
PLC	Programmable Logic Controller, an electronic device used to control the equipment mentioned herein
VFD	Variable Frequency Drive, an electric motor drive used to control the speed of electric motors
MCC	Motor Control Center, an electronic device that provides supervisory control over electric motors which are not controlled by a VFD
DP	Differential pressure
PID	Proportional Integral Differential, a control loop which includes changes to the control signal based on proportional, integrated and differential error terms.
Tag	A variable in the PLC code, which references a piece of equipment, input or output signal and whose name is based on ANSI/ISA 5.1 standard

II. Purpose of this Document

The purpose of this document is to define the structure and functional requirements for the LCLS-II cryoplant PLC control software. The software shall perform real-time monitoring and engineering unit conversion of all digital and analog signals connected to the PLCs. The PLCs shall provide all alarms and shutdowns for each system's equipment protection. The alarm set points will be operator modifiable. The shutdown set points will not be operator modifiable. The system will provide information to the operator via a locally mounted touch panel display and through a remote communications connection to the main control system. The system will handle all operational functions including startup sequences and normal/emergency shutdown sequences.

III. Reference Documents

The cryogenic plants' control systems are based on the P&IDs and reference documents listed in each system's software specification in addition to the following general reference documents:

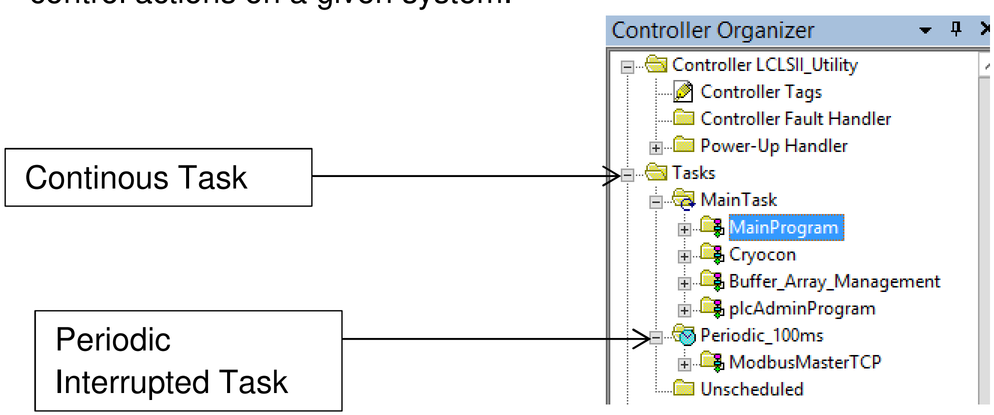
Document Number	Document Name	Revision
LCLSII-4.8-FR-0244	Cryogenics Plant Functional Requirements Specification	0
LCLSII-2.7-FR-0492	Cryogenic Controls Functional Requirements Specification	1
LCLSII-4.1-FR-0327	Cryogenic Systems Integration Functional Requirements Specification	2
LCLSII-4.9-FR-0057	Cryogenic Distribution System Functional Requirements Specification	2
LCLSII-2.5-IC-0056	Accelerator Systems to Cryogenic Systems Interface Control Document	4
LCLSII-4.9-IC-0058	Cryogenic Distribution System	3
LCLSII-2.7-IC-0277	Low Level Radio Frequency Interface Control Document	0
LCLSII-4.5-ES-0415	CM Instrumentation Specification	0
LCLSII-4.9-ES-0404	Cryogenic Distribution System Distribution Box Specification	2
LCLSII-2.7-ES-0964	Cryogenic Control Engineering Specification	0
LCLSII-4.5-ES-0974	Cryomodule Operational Conditions and Control	0
LCLSII-4.9.EN-0289	CDS Control Loop Functions	2

IV. Overview of the PLC Program Structure

Studio 5000 software is used for programming the ControlLogix L81E PLCs for the LCLS-II cryogenic plant. All the programming is done in ladder logic with exception of Turbine Efficiency calculation which is done in structured text format. A ControlLogix 5000 controller supports multiple tasks to schedule and prioritize the running of programs based on specific criteria. This balances the processing time of the controller. The controller runs only one task at one time. A different task can interrupt a task that is running and take control. In any given task, only one program runs at one time. Most systems will only have single task running, but the utility system PLC has an additional task for Modbus communication which will interrupt the main task periodically every 100ms.

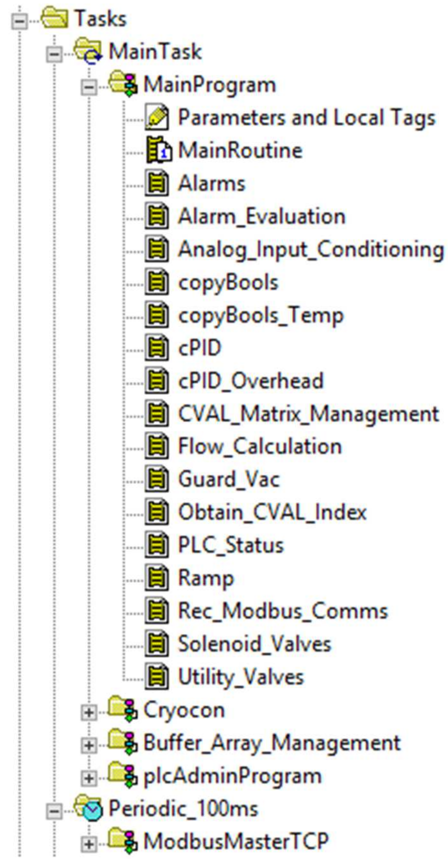
A. Main Task

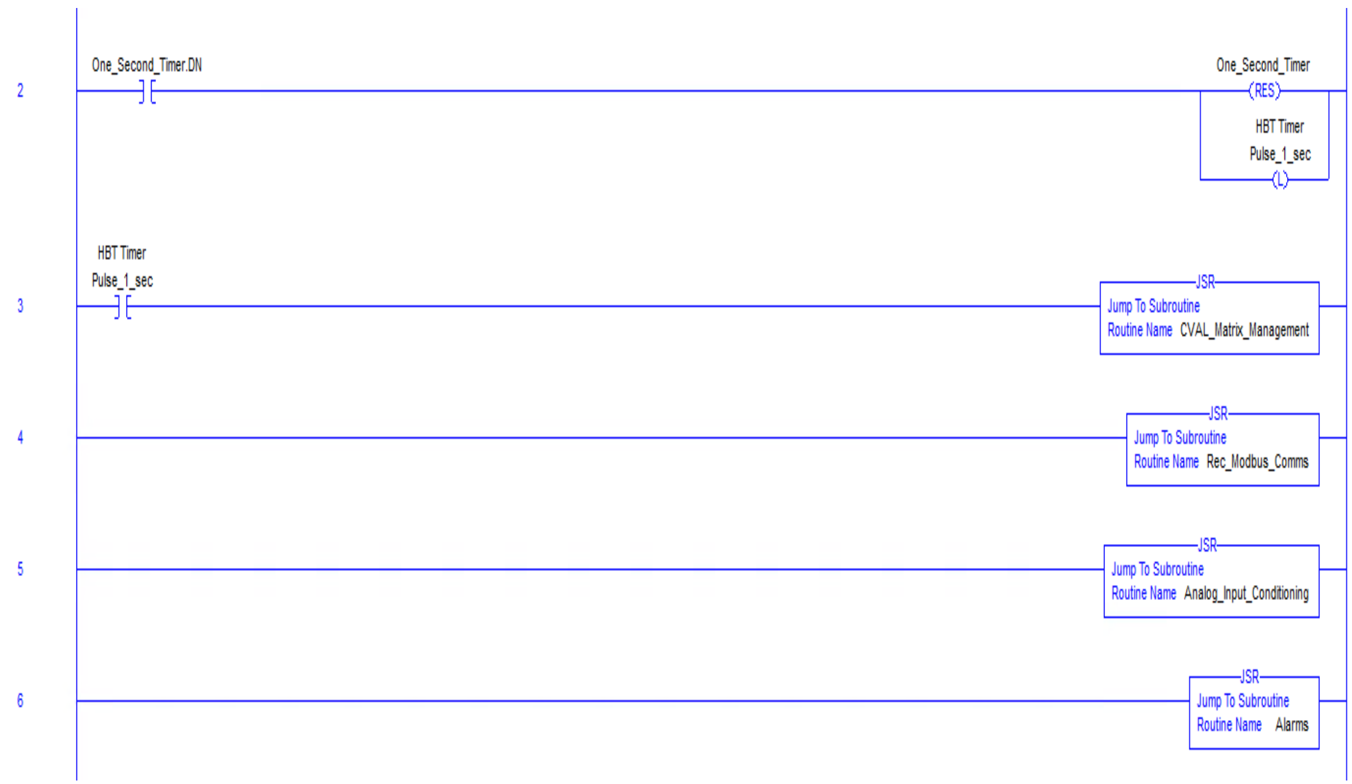
The PLC task is structured into a multiple programs. The MainProgram, Buffer_Array_Management, plcAdminProgram and Cryocon programs, which have their own set of routines. MainProgram serves as the program that contains the functional logic such as PID loops, sequencers, trips, and interlocks, while the other programs perform data management roles. This is the program excutes tasks sequentially and perform control actions on a given system.



B. Main Routine

The MainProgram is structured into multiple sub-routines. The MainRoutine is executed continuously and jumps to other sub-routines. All the sub-routines in the program should be called out in the MainRoutine for execution. In other words, sub-routine will be executed only if it is directly or indirectly called from the MainRoutine. Refer to the functional narrative for basic subsystem operation.

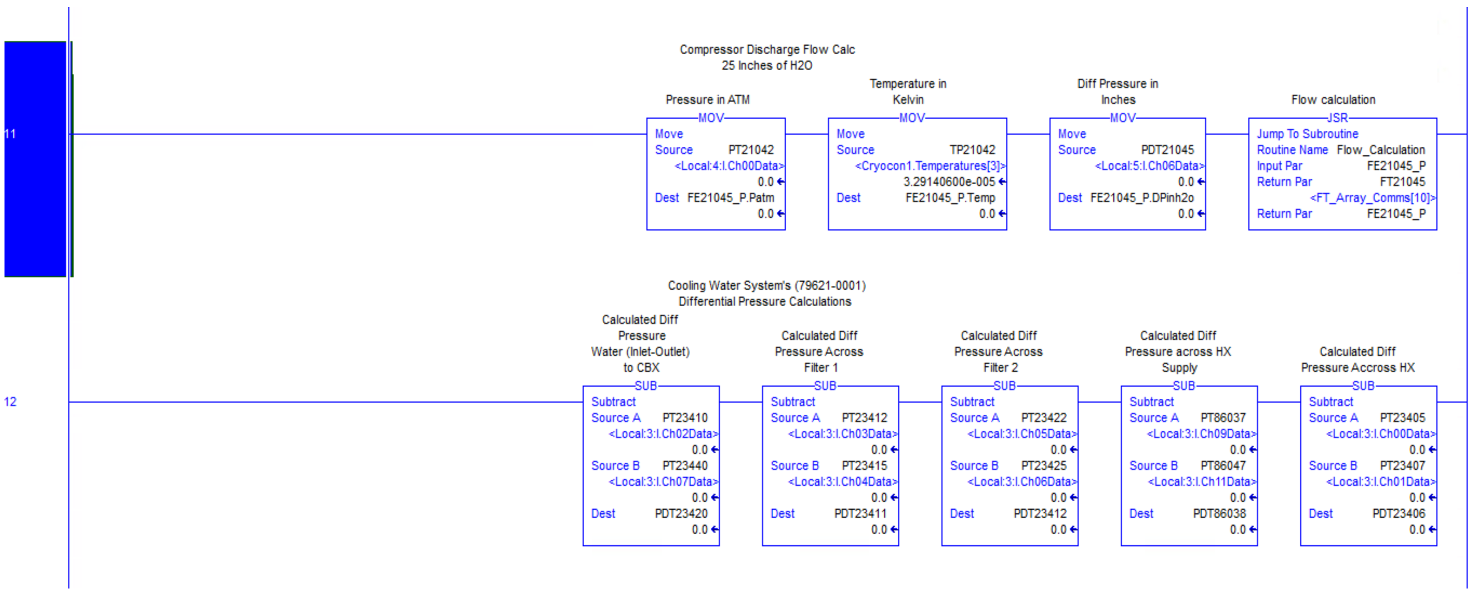




Example: *FlowCalculation sub-routine is not requested from the MainRoutine but when Main Routine executes Analog_Input_Conditioning it request FlowCalculation sub-routine indirectly and executes it.*

C. Analog_Input_Conditioning

This function is used to do various calculations on the analog inputs, including rescaling or converting units which are not scaled by the analog input card interface. Some differential pressures are calculated using the appropriate pressure transmitters and any signal/tag which is supposed to be calculated in software is performed in this routine. It also moves all of the input conditions into the appropriate arrays and calls the Flow_Calculation routine with those arrays as inputs for calculating the flow through a venturi flow meter in g/s.



D. Flow_Calculation

This function calculates the flow through a venturi flow element using the inlet pressure, temperature and differential pressure between the inlet pressure and venturi pressure. The controls engineer has to enter all the constants (Beta_4, C, Ks, MW, Z) from the venturi data sheet.

Tag Structure	Description	Tag type
Beta_4	β^4 Esitmated	Constant
C	Coefficient of Discharge	Constant
Density	Density upstream	Calculated Variable
DPinh2o	Diff Pressure in inches	Analog Input
Ks	Isentropic Exponent	Constant
MW	Molecular Weight	Constant
Patm	Pressure in ATM	Analog Input
Temp	Temperature in Kelvin	Analog Input
Y	Expansion Factor	Calculated Variable
Z	Compressibility upstream	Constant

$$Density = \frac{Pressure * Molecular\ Weight}{Compressibility\ upstream * 8314.4621 * Temperature}$$

$$Expansion\ Factor = \frac{Ks * (1 - \beta^4) * \{(DP * Ks) - (Pressure * Ks) + (Pressure)\}}{DP * (Ks - 1) * \{(DP * Ks) - (2 * \beta^4 * Pressure)\}}$$

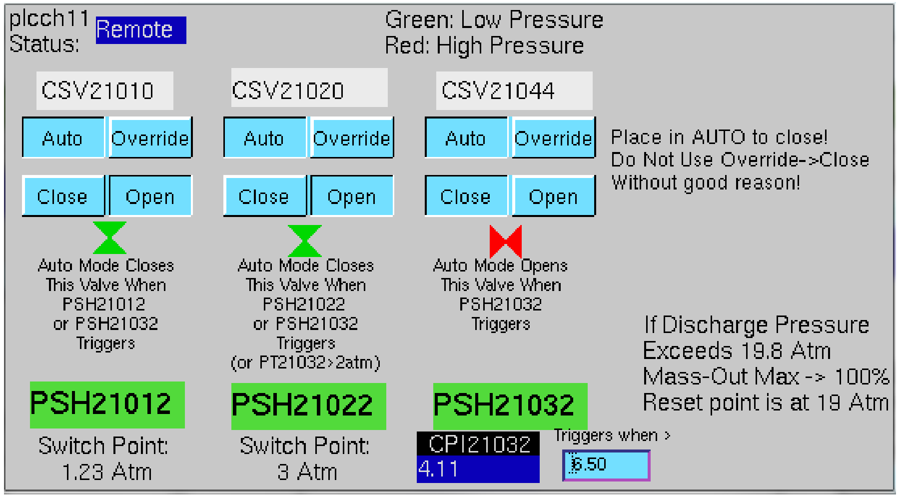
$$Flow = \sqrt{\frac{DP}{Density}} * 1414.214 * Coefficient\ of\ Discharge * Expansion\ Factor$$

***Note: Pressure and DP are converted to Pascal during calculation

E. Solenoid_Valves

This routine solely controls the operation of the solenoid valves. It copies override commands based on PLC local or remote status and sets or unsets solenoid valve commands based on the override command from one of the HMIs. It also opens and closes solenoid valves based on their process value setpoint.

HMI Suffix	EPICS Suffix
_Command_PX	_Command_X
_Override_PX	_Override_X



F. Buffer_IO

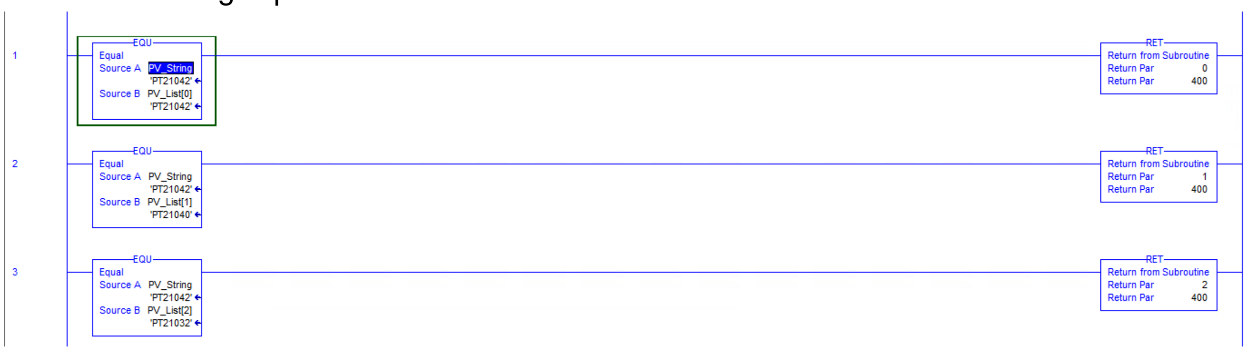
This subroutine takes in all of the input and output modules and creates buffer arrays. This is done so that a routine cannot directly write to outputs. An example of such a routine is a cold box protection routine that checks to make sure the output to a particular valve is at 100% before proceeding. These buffer arrays are used in various subroutines, i.e. CVAL_Matrix_Management.

G. EPICS_Watchdog

This routine handles a heartbeat between EPICS and the PLC and is called once per second. The two manipulate the same variable between 0 (PLC) and some positive number (EPICS). When the PLC does not get a response from EPICS (bit remains 0), it triggers a flag to indicate that the communications have failed and increments a counter, but this doesn't affect the normal control of the PLC. This only means that the operator can no longer make changes from EPICS to the PLC as the communication is lost. EPICSTMOUT is the only tag which should be tied into the main alarm tree.
EPICSHNDSHKVAL must be output from EPICS to the PLC which will be set in softIOC.

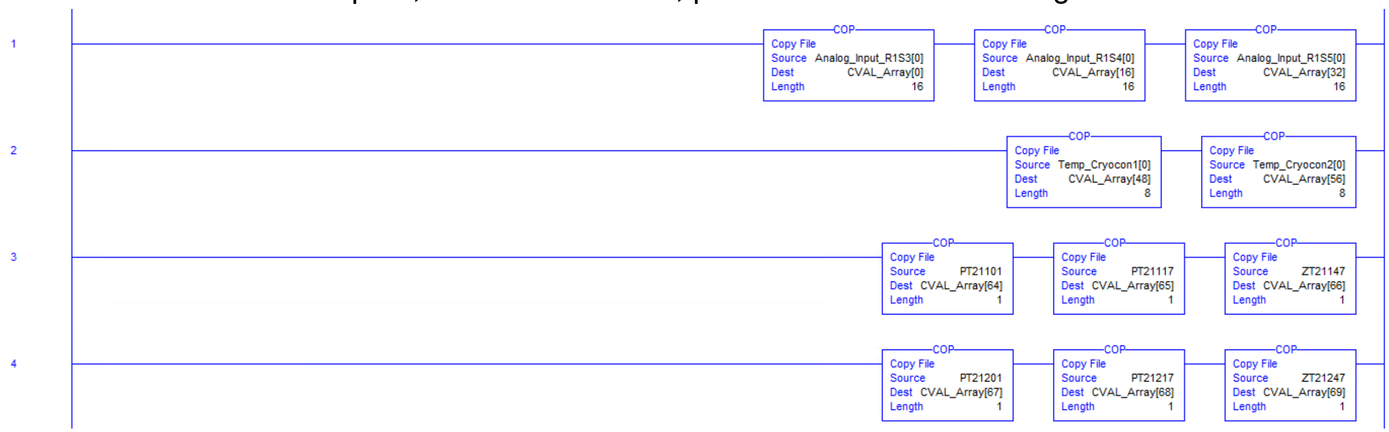
H. Obtain_CVAL_Index

This subroutine takes a user entered string (a PV tag number) from EPICS and utilizes an indexing method where this subroutine returns the index value of a channel corresponding to that string. This index value is then be utilized to grab the PV until a change in the PV source is detected and this subroutine is called again to get a new PV index. This method reduces the computing overhead required by the PLC by reducing the number of string equate checks needed.



I. CVAL_Matrix_Management

This subroutine puts all of the controllable signals into a matrix so they can be used by individual PID loops for later use via pointing to the appropriate index point in the array (See Obtain_CVAL_Index). Buffer arrays are used to create a matrix for analog input modules and temperature Cryocons, while individual signals are entered into the matrix for outputs, calculated values, produced or consumed signals.



J. cPID_Overhead

This subroutine handles the movement of PID settings between EPICS, the HMI/touchpanel, and the PID loop in the PLC. It accepts the setpoints for the PID in local (from the HMI) or remote (from EPICS) and moves them to the appropriate registers. Any changes made either on the HMI or EPICS will be reflected on the other. The changes made by the operator won't go into effect if the PID loop is in the breakdown condition (Breakdown_Condition). This breakdown condition will override all the settings and drive the PID loop output via a predefined program. It can be enabled in any routine. This routine accepts the string and looks for the index value in the Obtain_CVAL_index matrix routine and sets an error bit if the PID loop control variable is not in the matrix.

K. cPID

This routine handles the PID loop calculations and sets the PID loop output. They are velocity type PID loops and don't have an issue with integral wind-up (accumulation of integral term when at the outputs maximum or minimum). The integral term is simply a 'sum' that is added or subtracted from the output based on

Change = Error * Sample Time * Integral_Gain. At a maximum or minimum position this change request is ignored, there is no 'windup' that happens.

This routine accepts data from the PID_Overhead sub-routine and calculates the PID loop output based on whether the PID loop is position type or change type. “Position” refers to the PID loop output as a flat value that goes up or down (a simple 4-20mA output, for example). “Change” refers to a PID loop that simply outputs a positive or negative change request with a magnitude. Such a ‘change’ type is commonly used to control pneumatic slide valves or electric valves that rely on a digital output to drive it open or closed. The PID loop can be set into 4 different modes: manual, auto, sequencer and protector. Changes from manual or sequencer mode to auto mode are seamless and auto mode picks up from wherever manual mode was commanded to by setting the initial output value of the PID loop to the previous output value from sequencer or manual mode.

The modes are described below.:

a) Manual Mode

The PID loop sets the output to user desired value. This mode can only be overridden in a breakdown condition.

b) Auto Mode

The PID loop normally operates in this mode and it is the lowest priority mode.

c) Sequencer Mode

This mode is activated if the sequencer mode (.smod) bit is set in the program because certain conditions are met. Valves or any analog output will be operating in this mode during sequential fuctions (cooldown of the coldbox, shields, etc). This will take sequencer mode values to calculate the PID loop output. This mode can be overridden by manual mode.

d) Protector Mode

This mode is activated if the PID loop Process Variable () is out of range or if the error bit () was set from Obtain_CVAL_Index subroutine. In this case, if the fail safe bit (FSB) is active the PID loop will go to predefined position, otherwise it will lock the output to the current position. This mode can be overridden if sequencer or manual mode is activated, or if a valid tag name is entered as the Process Variable in the PID loop.

e) PID Loop Structure

(1) There are two user-defined data types. One is the PID loop itself that contains the relevant gains, minimums, maximums, and so on. The other is the ‘Overhead” that handles interface requests from the EPICS/HMI system.

(2) The elements of the PID/Overhead structure are below. These are accessed by “PID.**** and PID_Overhead.**** Where **** is the Name.

Black Text – EPICS Signals;
Blue Text – HMI Signal;
Red Text – PLC Signals just for information

Element Name	Type	Description
	PID LOOP STRUCTURE	
MODE	INT	0: Manual,1:Auto 2: Sequencer, 3: Protector
TYPE	INT	0: Change type, 1: Position type
CVAL	REAL	Process Variable
ORBV	REAL	Output readback Value

Element Name	Type	Description
OVAL	REAL	Output Value
MIN	REAL	Minimum Output Value
MAX	REAL	Maximum Output Value
DMIN	REAL	Minimum Output Change
DMAX	REAL	Maximum Output Change
VAL	REAL	Setpoint of PID loop
SVAL	REAL	Sequencer Output Request
MVAL	REAL	Manual Output Request
KP	REAL	Proportional Gain
KI	REAL	Integral Gain
KD	REAL	Deriviative Gain
err	REAL	Previous error
derr	REAL	Previous Difference In Error
MDt	REAL	Sample Time
dt	REAL	Previous Time Interval
smod	BOOL	Command for cPID to be in Sequencer
mmod	BOOL	Command for cPID to be in Manual
dm	REAL	Change in Output
odm	REAL	Output Change for dm Not Calculated
ct	DINT	Clockticks when PID Scan Occurs
OneShot	BOOL	
OneShotOff	BOOL	
pmod	BOOL	Command for cPID to be in Protector Mode
PVAL	REAL	Protector Output Request

Element Name	Type	Description
PEVAL	REAL	Protector Mode Fail Safe Valve Condition
FSB	BOOL	High priority Fail Safe Bit
OVERHEAD STRUCTURE (Accessed by "PID_Overhead.** Where **** is the Element name)**		
Mode_Rqst	DINT	Mode Request from EPICS (0: MANUAL,1:AUTO)
Mode_Rqst_P	BOOL	Mode Request from HMI (0: MANUAL,1:AUTO)
Mode_Status	DINT	Auto/Manual Indicator to EPICS (0: MANUAL,1:AUTO 2: Sequencer, 3: Protector)
Mode	SINT	Mode
ManPos	REAL	Manual Position
ManPos_Rqst	REAL	Manual Position Requested From EPICS
ManPos_Rqst_P	REAL	Manual Position Requested From Panel View
TieBk	REAL	Loop Tieback Value
SP	REAL	Loop Setpoint to EPICS
SP_Rqst	REAL	PID SP request from EPICS
SP_Rqst_P	REAL	PID SP request from Panel View
PV	REAL	Loop Process Variable to EPICS
PV_Iconics	REAL	Alternate Process Variable selected by Engineer through EPICS
PV_Iconics_EN	BOOL	User Request, buffered, from Iconics to switch PV Source. Normally this bit should be OFF before enabling PV_SourceSel_EN
PV_pid	REAL	Process Variable for PID loop
PV_Sel_Rqst	BOOL	User Request from Iconics to switch loop PV source (0: Default, 1: Engineer PV from Iconics)
PV_Source	BOOL	Flag to let Iconics know which PV the PLC is actually using (0:PLC

Element Name	Type	Description
		default, 1: Engineer PV from Iconics)
Override_Status	DINT	0:None,1:Auto, 2:Manual, 3:ValPos, 4:SP, 5:Auto&SP, 6:Manual&ValPos, 7:SP Lim
Override_Code	DINT	0:None,1:Auto, 2:Manual, 3:ValPos, 4:SP, 5:Auto&SP, 6:Manual&ValPos, 7:SP Lim
CV	REAL	Loop Control Variable to Iconics/PanelView
CV_pid	REAL	Control Variable for PID loop
WRITE	DINT	Move Iconics Loop Staged Values to PLC when this is set to 1
WRITE_P	REAL	Move Panel View Loop Staged Values to PLC when this is set to 1
MAX_X	REAL	Staged Value from Iconics for PID Max Limit
MAX_PX	REAL	Staged Value from Panel View for PID Max Limit
MAX	REAL	Value for PID Max Limit
MIN_X	REAL	Staged Value from Iconics for PID Min Limit
MIN_PX	REAL	Staged Value from Panel View for PID Min Limit
MIN	REAL	Value for PID Min Limit
DMAX_X	REAL	Request for maximum change of valve position from EPICS
DMAX_PX	REAL	Reqest for maximum change of valve position from touch panel
DMAX	REAL	Actual maximum change of valve
DMIN_X	REAL	Request for minimum change of valve position from EPICS
DMIN_PX	REAL	Request for minimum change of valve position from touch panel
DMIN	REAL	Actual minimum change of valve
ST_X	REAL	Staged Value from Iconics for PID update time (s)
ST_PX	REAL	Staged Value from Panel View for PID update time (s)
UPD	REAL	Value for PID update time (s)
GAINP_X	REAL	Staged PID Proportional Gain from Iconics

Element Name	Type	Description
GAINP_PX	REAL	Staged PID Proportional Gain from Panel View
GAINP	REAL	PID Proportional Gain
GAINI_X	REAL	Staged PID Integral Gain from Iconics
GAINI_PX	REAL	Staged PID Integral Gain from Panel View
GAINI	REAL	PID Integral Gain
GAIND_X	REAL	Staged PID Derivative Gain from Iconics
GAIND_PX	REAL	Staged PID Derivative Gain from Panel View
GAIND	REAL	PID Derivative Gain
Mode_LR	BOOL	Control Mode (0:Local, 1Remote)
PV_String_PX	STRING	Process Variable channel (Local)
PV_String_X	STRING	Process Variable Channel (Remote)
PV_String	STRING	
CVAL_Index	DINT	
Breakdown_Condition	BOOL	
Breakdown_Value	REAL	
Mode_String	STRING	
NoMatch	BOOL	PID Input Variable didn't match
CFdisable	INT	Channel Fault Disable
CHFault	BOOL	
ControlFail	BOOL	Channel fail

f) PID Calculation

A discrete form of the PID algorithm is as follows:

M(n) = KP*E(n) + KI*SUMi(E(i)*dT(i)) + KD*(E(n) -E(n-1))/dT(n) + Mr where,

M(n) Value of manipulated variable at nth sampling instant
KP,KI,KD Proportional, Integral, and Differential Gains (Also called GP/GI/GD)
Positive gains are for a Reverse-acting PID loop, Negative gains are for a Direct-acting one.
NOTE: KI is inverse of normal definition of KI
E(n) Error at nth sampling instant
SUMi Sum from i=0 to i=n
dT(n) Time difference between n-1 and n
Mr midrange adjustment

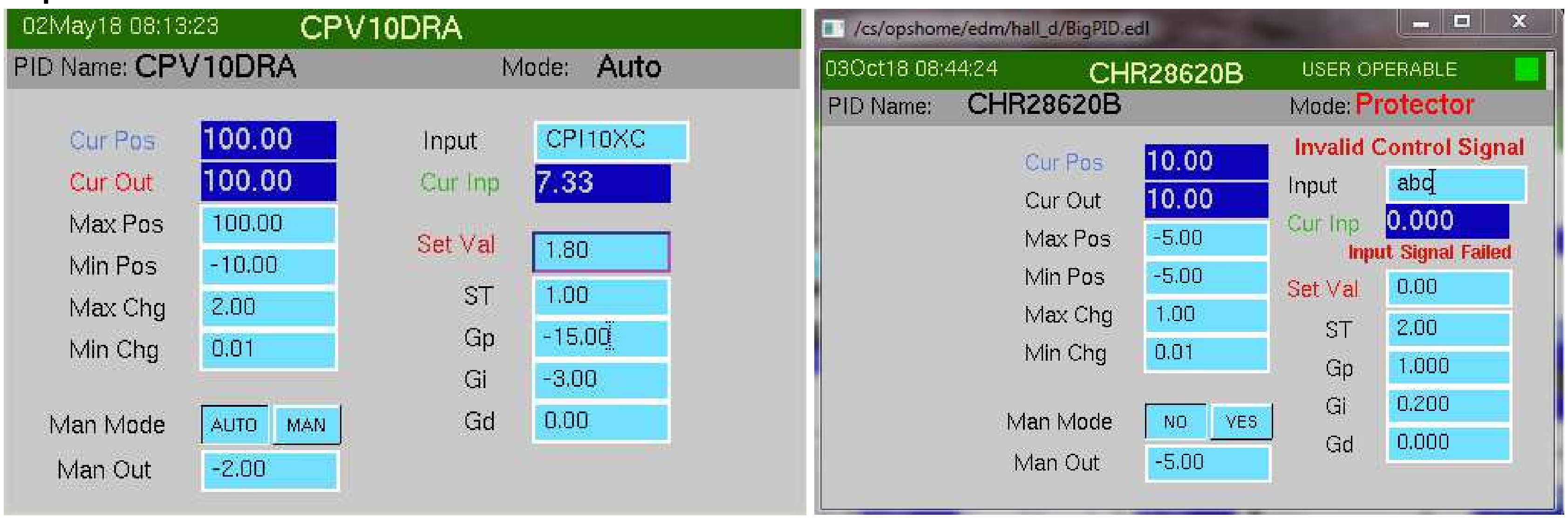
Taking first differential yields
delM(n) = KP*(E(n)-E(n-1)) + KI * E(n)*dT(n) + KD*((E(n)-E(n-1))/dT(n) - (E(n-1)-E(n-2))/dT(n-1)) or using variables defined in following **dm = kp*de + ki*e*dt + kd*(de/dt - dep/dtp). If dm > Max change, then dm = Max Change. If dm < Min Change, then dm = 0**

(1) Calculate the elapsed time since the last time the loop was called
(2) If the elapsed time is longer than the sample time
  (a) Then calculate the error and new valve position
    (i) If dt is greater than 0 and dtp is greater than 0
      (a) Then d = KD*(de/dt-dep/dtp)
    (ii) If dt or dtp is less than 0, d = 0.

Function	Description	Tag
Delta time		dt
Previous delta time		dtp
		derr
	Previous delta error	dep
Current error	Previous process value minus current process value	e
Previous process value		VAL

Delta error	Current error minus previous error	de
Previous error	Previous error	err
Proportional component	KP times delta error	p
Integral component	KI times current error times delta time	i
Derivative component		d
	Sum of proportional, integral and derivative components	Dm
Sample Time	Time between executions of the pid loop. Shown as ‘ST”	Mdt

Sample screens:



Note: *Positive gains are for a Reverse-acting PID loop, Negative gains are for a Direct-acting one.*

Naming convention:

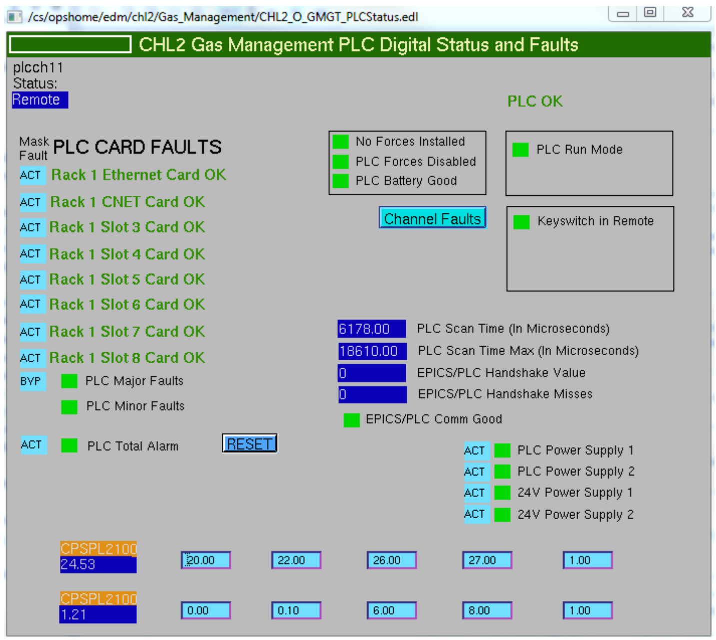
Example:

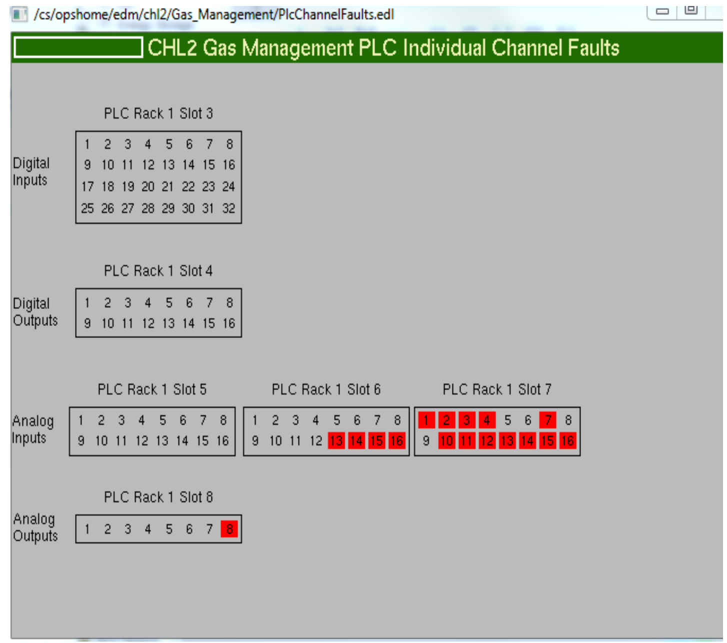
PID Loop Name	Description	Equipment Tag	Process Variable Tag	Process Variable Set Point
CPV71010	HGD to Recovery Compressor	CPV71010	CPT71010	1.07
CPV71010A	HGD to Recovery Compressor	CPV71010	CTP71010	300

Here, the recovery compressor valve is being controlled of two diferent process variables. So while naming the PID loop, ValveName (CPV71010) will be the PID loop name for primary loop while ValveName**A** (CPV71010A) will be PID loop name for secondary loop.

L. PLC_Status

This subroutine is for determining relevant alarms on the PLC system. It records all the PLC faults, as well as determines individual channel faults. This routine is redundant with plcAdminProgram with some additional features. SLAC will make a decision whether to use or not those features. This routine provides information to operator about PLC health and modules in detail.





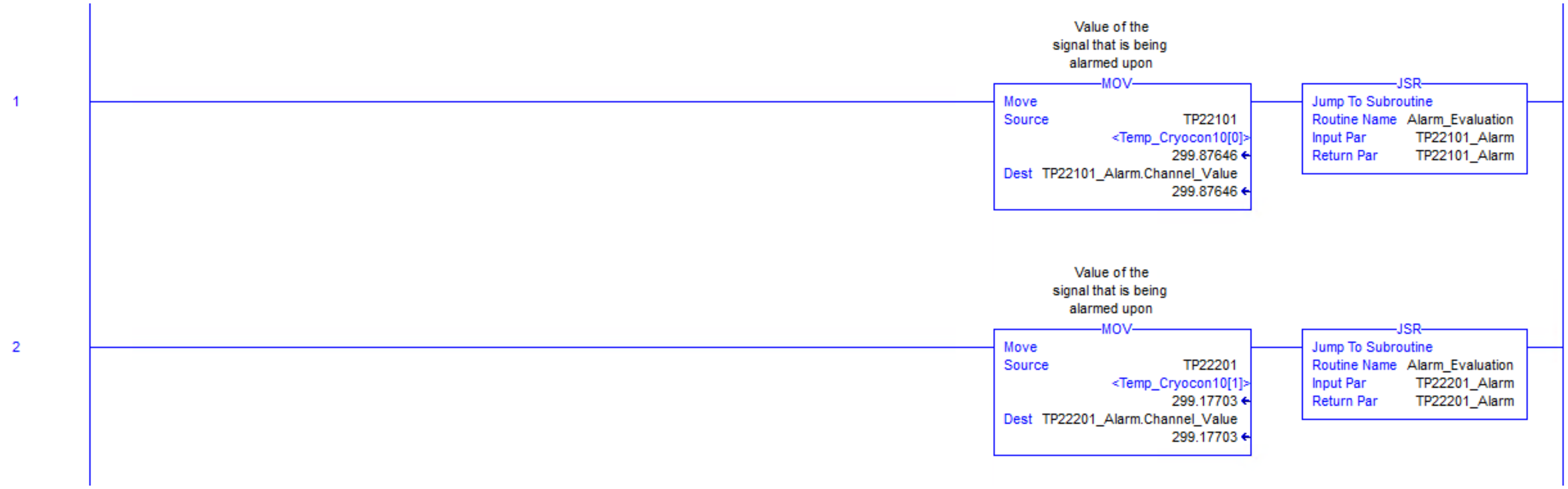
***Note: The EPICS developer has to look at the PLC_Status Routine and use appropriate tags when developing this screen.

Tag	Type	Description
ChannelFaults[x].y	DINT	Faults Array for Digital Input Module
ChannelFaults[x].y	INT	Faults Array for Digital Output Module
ChannelFaults[x].y	INT	Faults Array for Analog Input Module
ChannelFaults[x].y	INT	Faults Array for Analog Output Module
ChannelFaults[x].y	INT	Faults Array for Digital Input Module
PLC_Alarm[x].y	DINT	Faults Array for PLC Faults

PLC_Reset.0	DINT	PLC Fault Reset Array

M. Alarms

The alarms routine loads the current channel value into the _Alarm structure and calls the Alarm_Evaluation routine.



Example: If system alarms on pressure PT71010 then the tag associated with high alarm setpoint will be PT71010HiHi and for low pressure alarm set point it will be PT71010LoLo. While PT71010_Alarm.Status will be an actual interger which will represent which level (Hi,HiHi,Lo,LoLo) of alarm is active.

N. Alarm_Evaluation

All of the following analog signals will have two high alarms and two low alarms, the first set of alarms is a warning ‘yellow alarm’ and the second set is a serious alarm ‘red alarm’ and operator should take actions. All alarm set points are to be modifiable by operators. Process values greater than the Hi alarm or lower than the Lo alarm are in a state of ‘yellow alarm’. Process values greater than the HiHi alarm or lower than the LoLo alarm are in a state of ‘red alarm’. Process values greater than the MaxRange alarm or lower than the MinRange alarm are in a state of

'SLAC defined alarm color'. A red alarm supersedes a yellow alarm. These alarms' settings will exist in the software even if there is no intention to use a specific alarm setting and operator can disable particular alarm using a enable.disable button. The setpoint tags use the same engineering units as the process variable.

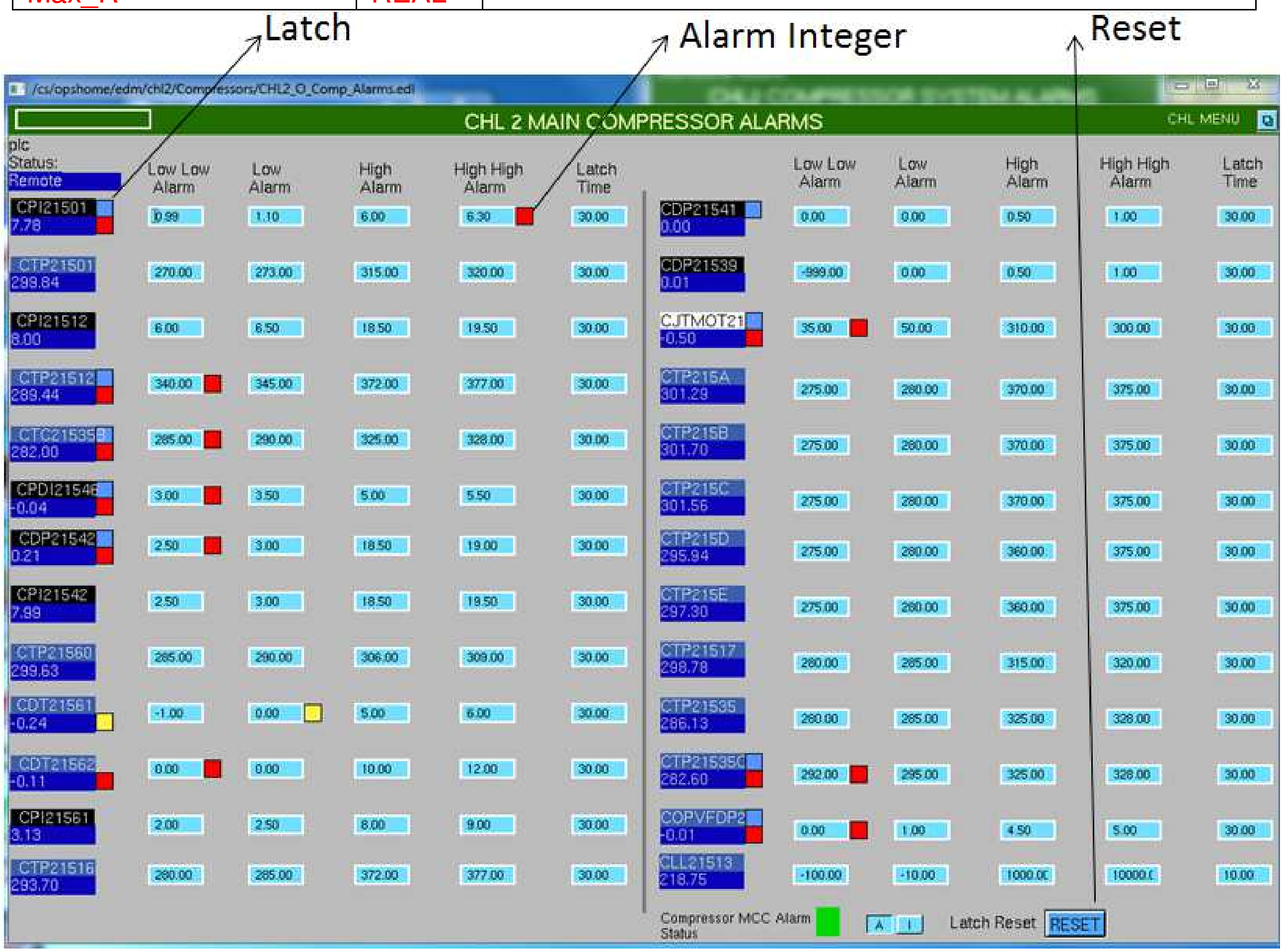
(1) This subroutine handles the alarm status of an alarm channel, and synchronizes setpoints between local (HMI) and remote (EPICS) mode as changes are made.
(2) The channel that is being alarmed upon has to be moved into the alarm channel independently of this subroutine. i.e Alarm_Evaluation subroutine should be called in main/alarm routine.
(3) This subroutine compares the channel value with their corresponding setpoints and generates an integer. (Lo=0, LoLo=1, NoAlarm=2, Hi=3, HiHi=4, MaxRange=5 or MinRange=6).
(4) This subroutine also latches the alarm if the HiHi, LoLo alarm is active for longer than the latch time.
(5) Reset will be required to clear the latch but alarm integer gets cleared once the alarm is ok (i.e auto reset of alarm).
(6) HiHi, LoLo, MaxRange, MinRange alarm setpoints should be displayed on the alarm screen

Black Text – EPICS Signals
Blue Text – HMI Signal
Red Text – PLC Signals just for information (no action required)

Tag Suffix	Type	Description
MinRange	REAL	Minmum Limit after which signal is of bad quality
Lo	REAL	Value for Lo Limit
Lo_X	REAL	SP from EPICS for Lo Limit
Lo_PX	REAL	SP from Touchscreen HMI for Lo Limit
Lo_R	REAL	
Lo_Lo	REAL	Value for LoLo Limit

Lo_Lo_X	REAL	SP from EPICS for LoLo Limit
Lo_Lo_PX	REAL	SP from Touchscreen HMI for LoLo Limit
Lo_Lo_R	REAL	
Hi	REAL	Value for High Limit
Hi_X	REAL	SP from EPICS for Hi Limit
Hi_PX	REAL	SP from Touchscreen HMI for High Limit
Hi_R	REAL	
Hi_Hi	REAL	Value for High High Limit
Hi_Hi_X	REAL	SP from EPICS for HiHi Limit
Hi_Hi_PX	REAL	SP from Touchscreen HMI for HighHigh Limit
Hi_Hi_R	REAL	
Channel_Value	REAL	Value of the signal that is being alarmed upon
Latch_T	REAL	Latch Time (in ms)
Latch_T_X	REAL	Latch Time Remote Rqst
Latch_T_PX	REAL	Latch Time Panel Rqst
Latch	BOOL	Latch Status
Status	DINT	Alarm Status
Write	SINT	EPICS Alarm SP Load Control 0: No action 1: Load Alarm Setpoints into PLC
Write_P	SINT	Touchscreen Alarm SP Load Control 0: No action 1: Load Alarm Setpoints into PLC
Latch_TMR	TIMER	
disable	INT	Disable the alarm
MaxRange	REAL	Maximum Limit after which signal is of bad quality
Min_PX	REAL	SP from Touchscreen HMI for Minimum Range
Min_X	REAL	SP from EPICS for Minimum Range
Min_R	REAL	

Max_PX	REAL	SP from Touchscreen HMI for Maximum Range
Max_X	REAL	SP from EPICS for Maximum Range
Max_R	REAL	



O. copyBools and copyBools_Temp

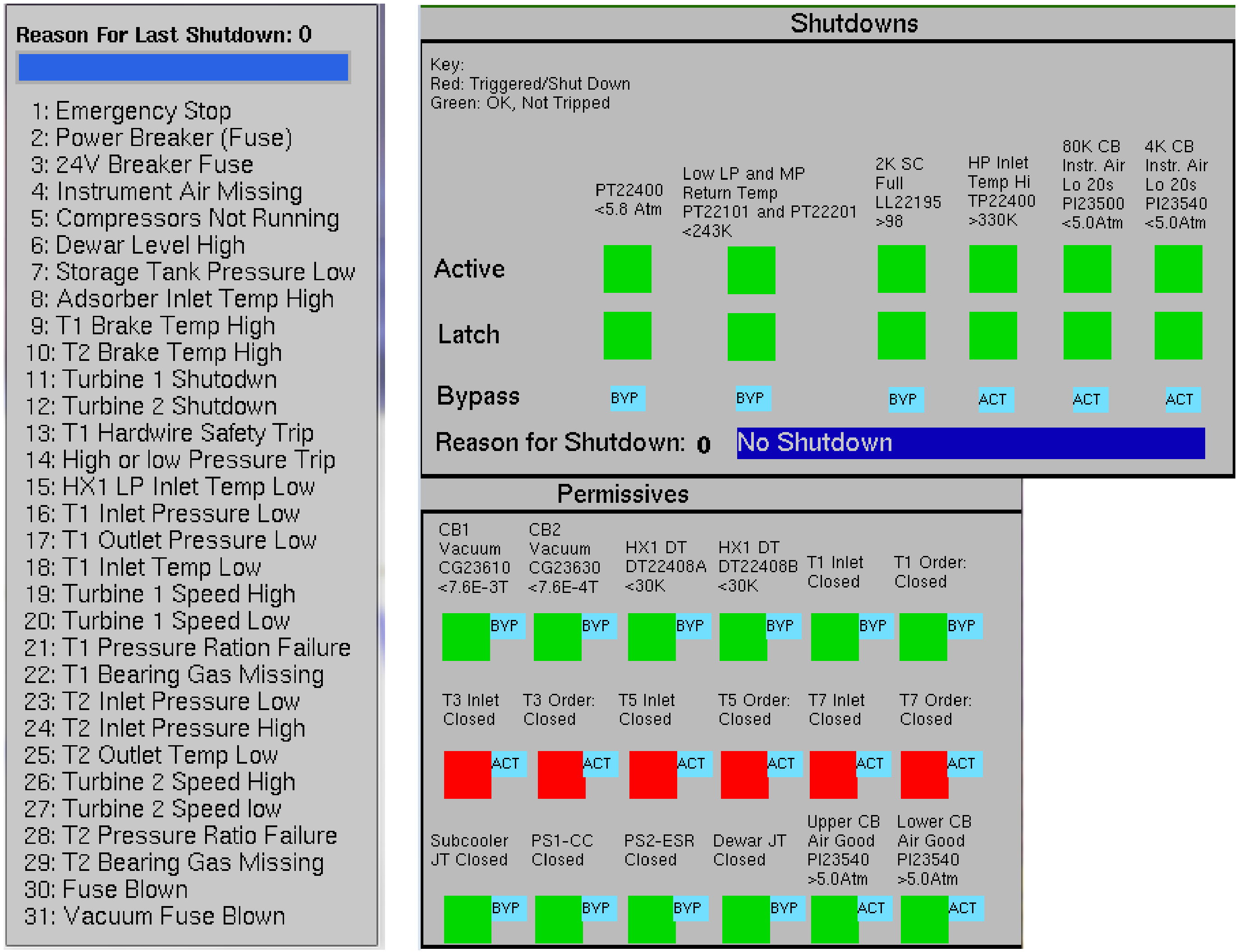
This routine detects if the analog input or Cryocon temperature has failed and periodically (once/second) creates an array "ControlChannelFaultsArray". Detection is done using a buffer fault array or compared with a preset values. This array also used in a PID loop where it can set the channel fault bit and lock the PID loop in protector mode.

P. Trip_Evaluation

Some select signals will have a high trip and/or a low trip. The machine or a part of the machine will shutdown to protect itself if the conditions for trip are satisfied. All trip set points can be read but can not be modified by operators. Unlike alarms, only selected signals will have a trip associated with them. The program will also output a text string and interger code for cause of the trip. The setpoint tags use the same engineering units as the process variable.

(1) This subroutine handles the trip status of a channel.
(2) This subroutine generates an integer and string for each individual trip type..
(3) This subroutine will latch the trip once its triggered.
(4) Reset will be required to clear the latch, trip, shutdown interger and shutdown string.
(5) TripHi, TripLo setpoints should be displayed on the screen.

Example: If system trips on pressure PT71010 then the tag associated with high trip setpoint will be PT71010TripHi and for low pressure trip setpoint it will be PT71010TripLo



Reason For Last Shutdown: 0
1: Emergency Stop
2: Power Breaker (Fuse)
3: 24V Breaker Fuse
4: Instrument Air Missing
5: Compressors Not Running
6: Dewar Level High
7: Storage Tank Pressure Low
8: Adsorber Inlet Temp High
9: T1 Brake Temp High
10: T2 Brake Temp High
11: Turbine 1 Shutodwn
12: Turbine 2 Shutdown
13: T1 Hardwire Safety Trip
14: High or low Pressure Trip
15: HX1 LP Inlet Temp Low
16: T1 Inlet Pressure Low
17: T1 Outlet Pressure Low
18: T1 Inlet Temp Low
19: Turbine 1 Speed High
20: Turbine 1 Speed Low
21: T1 Pressure Ration Failure
22: T1 Bearing Gas Missing
23: T2 Inlet Pressure Low
24: T2 Inlet Pressure High
25: T2 Outlet Temp Low
26: Turbine 2 Speed High
27: Turbine 2 Speed low
28: T2 Pressure Ratio Failure
29: T2 Bearing Gas Missing
30: Fuse Blown
31: Vacuum Fuse Blown
Shutdowns
Key:
Red: Triggered/Shut Down
Green: OK, Not Tripped
PT22400 <5.8 Atm
Low LP and MP Return Temp PT22101 and PT22201 <243K
2K SC Full LL22195 >98
HP Inlet Temp Hi TP22400 >330K
80K CB Instr. Air Lo 20s PI23500 <5.0Atm
4K CB Instr. Air Lo 20s PI23540 <5.0Atm
Active
Latch
Bypass
BYP BYP BYP ACT ACT ACT
Reason for Shutdown: 0
No Shutdown
Permissives
CB1 Vacuum CG23610 <7.6E-3T
CB2 Vacuum CG23630 <7.6E-4T
HX1 DT DT22408A <30K
HX1 DT DT22408B <30K
T1 Inlet Closed
T1 Order: Closed
BYP BYP BYP BYP BYP BYP
T3 Inlet Closed
T3 Order: Closed
T5 Inlet Closed
T5 Order: Closed
T7 Inlet Closed
T7 Order: Closed
ACT ACT ACT ACT ACT ACT
Subcooler JT Closed
PS1-CC Closed
PS2-ESR Closed
Dewar JT Closed
Upper CB Air Good PI23540 >5.0Atm
Lower CB Air Good PI23540 >5.0Atm
BYP BYP BYP BYP ACT ACT

V. General Suffixes

1. All software switches will have the suffixes of RST and SP. These refer to reset and setpoint, respectively.
2. Heaters will have R, CMD, WM and O suffixes for digital input, digital output, analog input and analog output respectively.
3. Emergency stop will have suffix ES.
4. Start will be STR and stop will be STP.
5. Pressure transducers having suffixes L and H represent low range and high range transducer respectively.

VI. Digital Indicators

Description	Colors
Main Motor On/Off	Green (on) or red (off)
Local/Remote Control	Green (remote) or yellow (local)
Alarms, Various	Red (alarm) or invisible on main page, red or green on alarms page
Shutdowns, Various	Red (shutdown) or invisible on main page, red or green on alarms page

A. General points

1. Color scheme shall match existing cryogenics and SLAC schedule.
2. Alarms, shutdowns, valve status, etc, shall be identified by text as well as color.

VII. General Comments

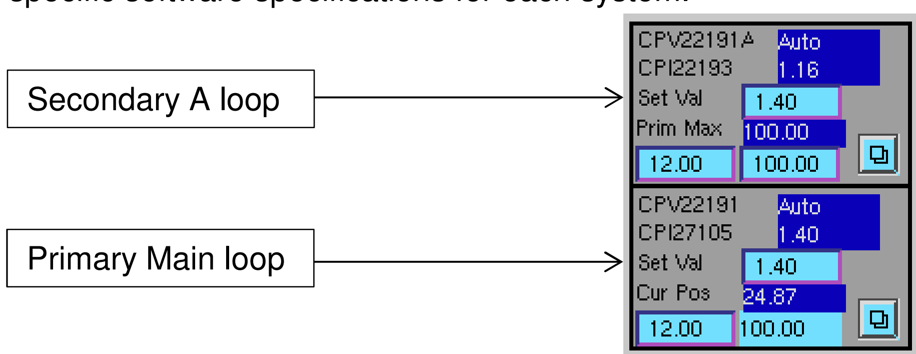
A. Analog Outputs:

Most analog outputs will have a PID loop attatched to it. Pneumatic valves acting in some other manner will be noted in the appropriate specific software specification.

B. Prevention of Trip:

JLAB has two layers of techinques used to prevent trip. The first layer is the addition of secondary PID loops for process critical valves/heaters. This PID loop (A loop) sets the maximum (or minimum) position of the primary PID loop and will limit the opening of valve if the plant operates in any upset condition. Tertiary PID loops can be set up to add a third control point, adjusting the ‘A’ loop or primary loop minimum or maximum as need be.

Second layer is predefined sequences and interlocks which implements programmed steps to either open or close valves. For example, aa interlock closes the gas management mass-in valve in the event of a compressor shutdown or high suction pressure. In this case actions are implemented immediately, unlike the first layer which slowly tries to protect the plant using PID settings. These sequence functions are detailed in the specific software specifications for each system.



C. Digital Commands from EPICS

1. If an on/off, reset, or toggle command from EPICS needs to be accommodated, the tag on the PLC side should be a bit of an integer rather than a BOOL tag. This is done to both distinguish the tag as an EPICS communicating tag and to resolve an issue in some versions of the EtherIP driver in EPICS that make writing to a BOOL tag difficult.