Hytep_H_2 Refuel Test Station - Code Overview

Prior to Jan 1 2016	main document drafted, required editing
Jan 26-29 .2016	edit and added content
Feb 1.2016	edit and added content
June 7.2016	Minor edit and added content

shared

Logins

Initial logins were set up as shown below during development. These may no longer be current once the customer took delivery of the system and amended these settings.

WES7 (*)	**Teamviewer**	**TightVNC**
● cDAQ login: ***Admin*** ● pwd: first letter is capital ● pwd clue: Sun's main element ● login info main have been changed by end-user	● ID: 372-376-809 ● pw: ***BSISupport*** ● used for WAN access, required internet connection ● A whitelist of remote computer may be added for security ● no longer runs at startup so operator had to initiate TeamV support exe	● requires known IP ● access pw clue: Sun's main element ● setting change pw clue: Sun's main element ● used on LAN access, no need for internet access. Will be blocked by routers.

Remote Access

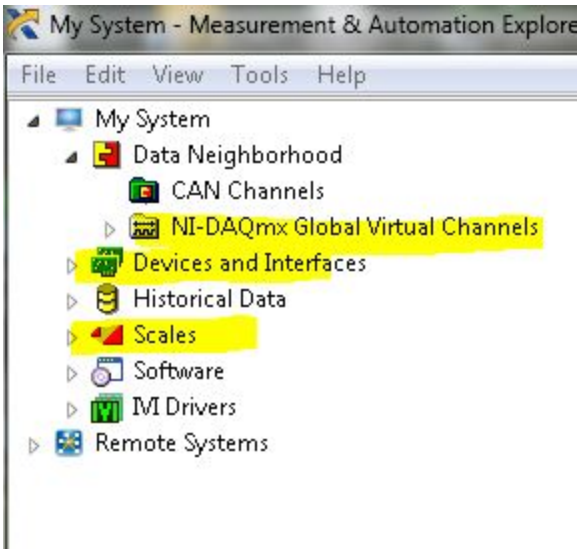
Options

1. Using Teamviewer ver 7.x for WAN access when internet connection is available
2. Using TightVNC for LAN or direct connection with straight network cable.

I/O Channels

All the hardware modules, I/Os channels, scales are defined outside the LabVIEW program. They are instead laid out as a **NI configuration file** that can be viewed in the NI Measurement and Automation Explorer (NI-Max). This way each channel's operation can be tested independently from the LabVIEW application.

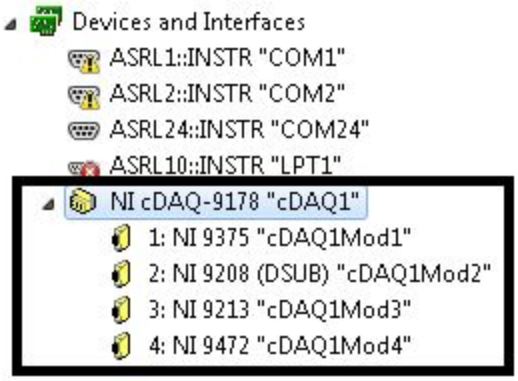
NI Measurement and Automation Explorer (MAX)



In MAX, the system appears as a tree structure with branches. Of specific interest in our case, the following branches are defined

- Device and Interfaces
- NI-DAQmx Global Virtual Channels
- Scales

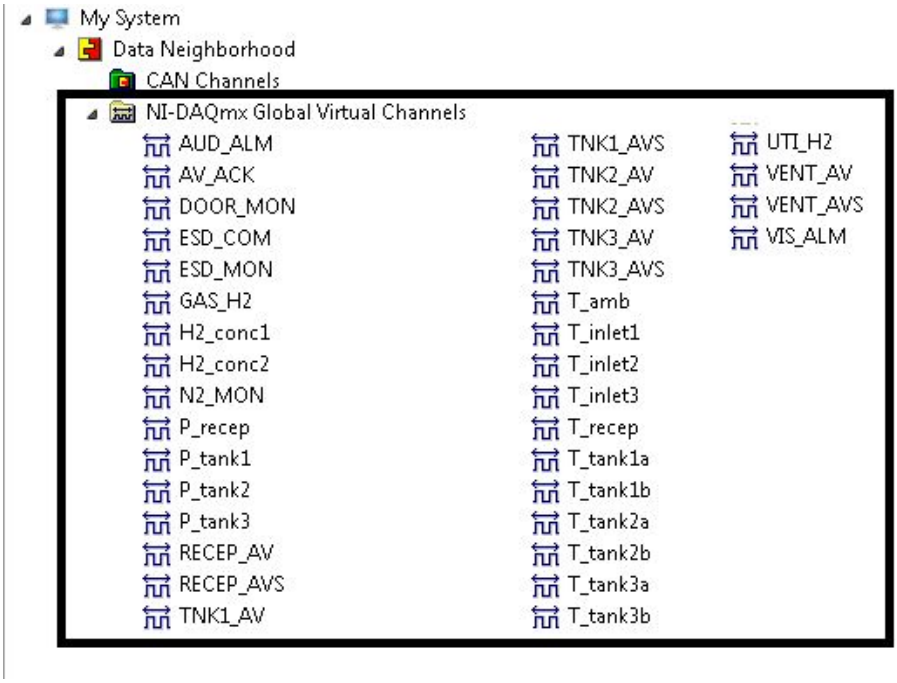
Device and Interfaces



the HyStep hardware is a cDAQ controller that **must** remain defined as "**cDAQ1".** This device denomination is used in the LabVIEW program to identify which device is used. Below cDAQ1 are all the hardware modules found in the system. Each module can have all its hardware channels exposed for operation check.

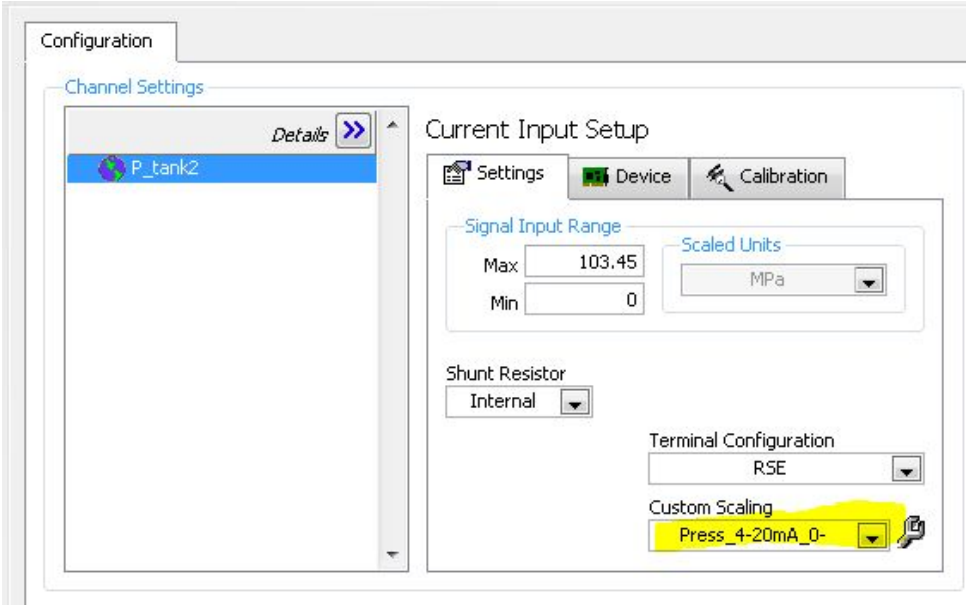
NI-DAQmx Global Virtual Channels

Global Virtual Channels encapsulate physical channels, channel types and scaling.



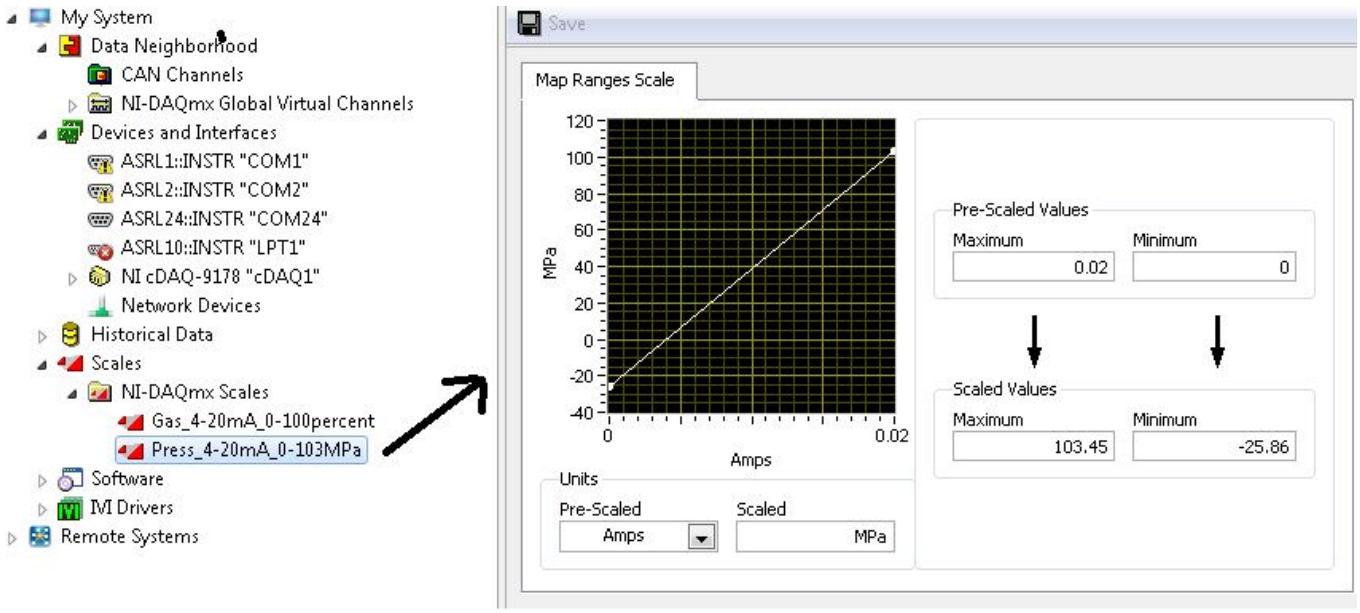
All the I/O channels used in the HyStep application are defined here. **Their names** are critically important, and should not be changed, as they are used inside the HyStep application. Some of the channels, such as the Analog Inputs are treated dynamically, such as the channel names with “P_” and “T_” prefixes since they can be grouped for graphing and saving together. Others, like individual digital outputs (DOs) are used individually in the Hystep program as they relate to specific conditions.

An example of pressure channel setting is shown below with its **signal range** and **custom scaling**



If a sensor was to be replaced, ranges can be be changed accordingly. The **Custom Scaling** refers to the **Scales** branch in the system tree. They are used to produce an engineering value as an output from the sensor measurement in the program.

Scales



For instance, the pressure sensors are using the "**Press_4-20mA_0-103MPa"** user scale that was defined with the mapping showing next to the tree.

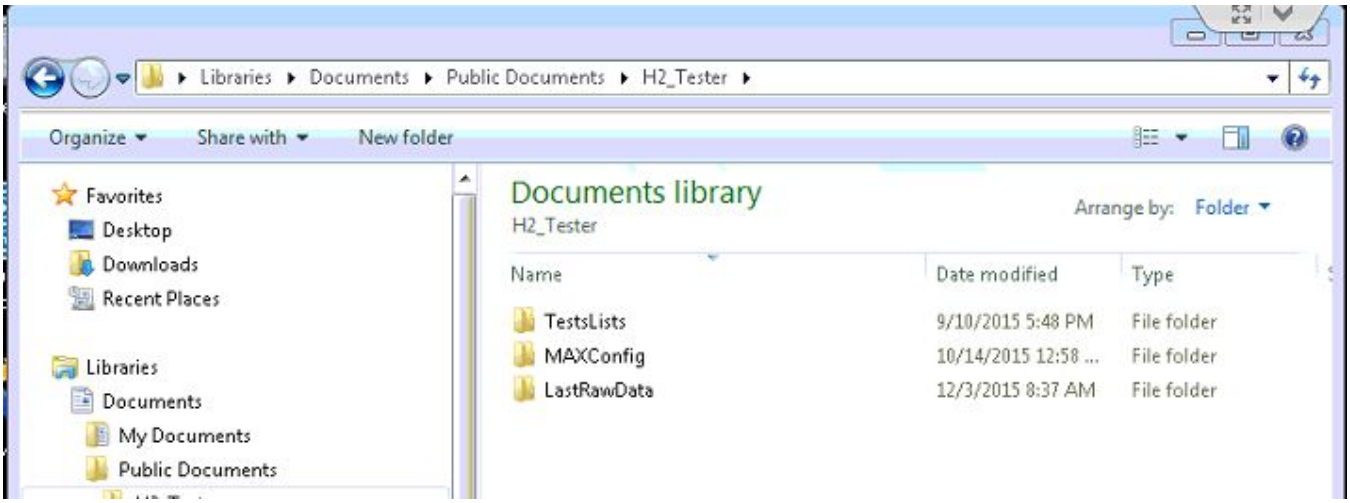
With these settings, performing a channel read in the program will produce a number representing the an engineering value for the variable being read. Scaling is not performed in the program and is not hard-coded, so it can be changed accordingly.

NI Configuration File

This file contains all the settings (channel names, scale, hardware channel mapping etc..) for the hardware setup for the project. It is located on:

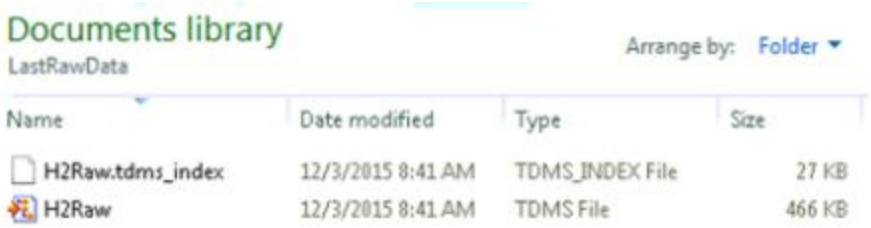
- **C:\users\Public\Public Documents\H2_Tester**MAXConfig

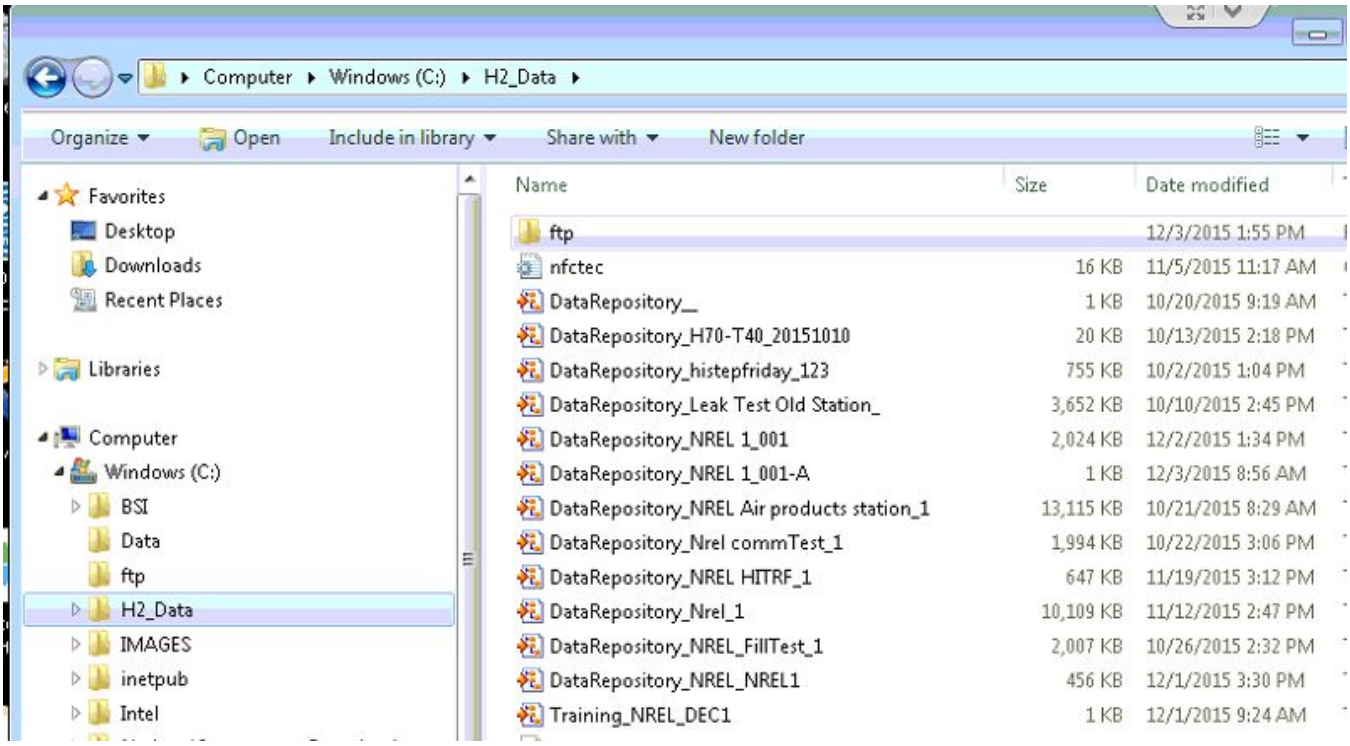
File Organization (Acquisition Application)



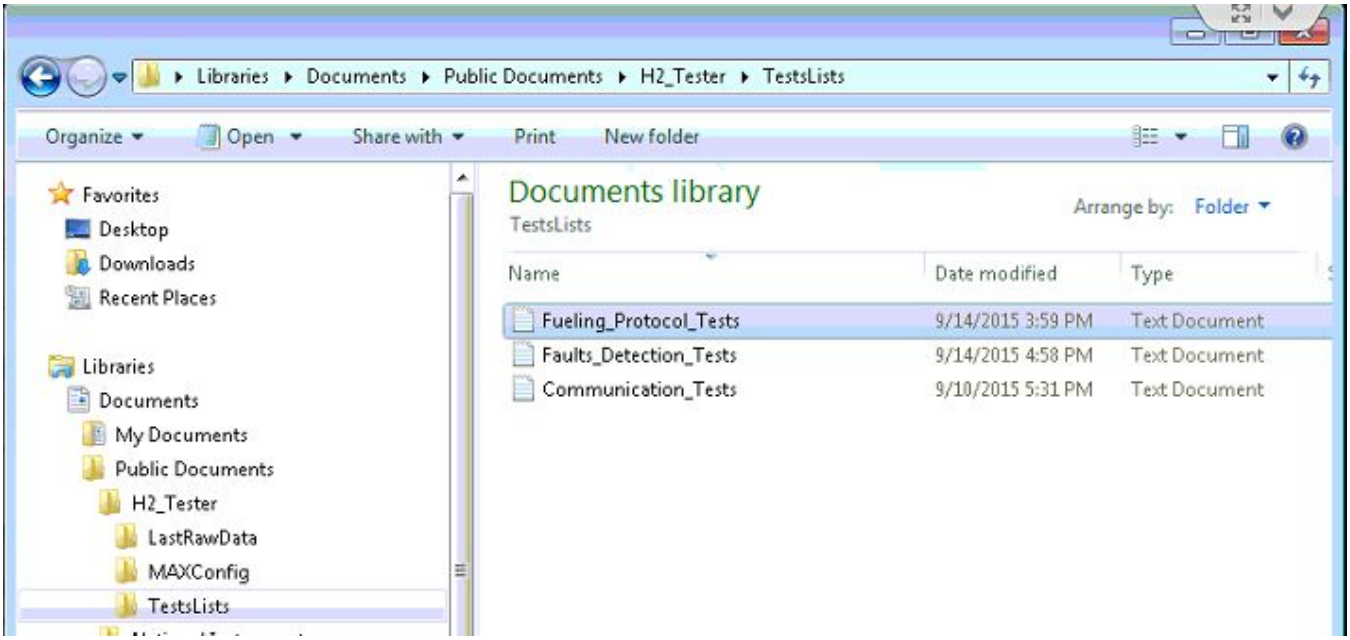
..\Public\Public Documents

- **C:\users\Public\Public Documents** must include the ***H2_Tester*** folder that contains other reference documents such as:
    - C:\Users\Public\Public Documents\H2_Tester***LastRawData*** folder, a placeholder for the current data being acquired and saved during a test.
    - This folder will be automatically created by the application if they do not exist.
    - **H2Raw.tdms** is a container for the current data being acquired. This data is later transferred to a user-defined folder, such as **C:\H2_Data** (default), following a prompt asking the user to either keep or reject the new data.
    - Guideline: 10 min of raw data , 20 channels, 10 Hz saving rate, ~ 830 Kbytes. (almost 1Mb)

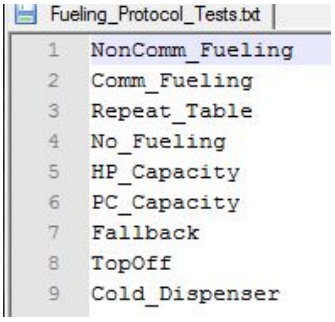




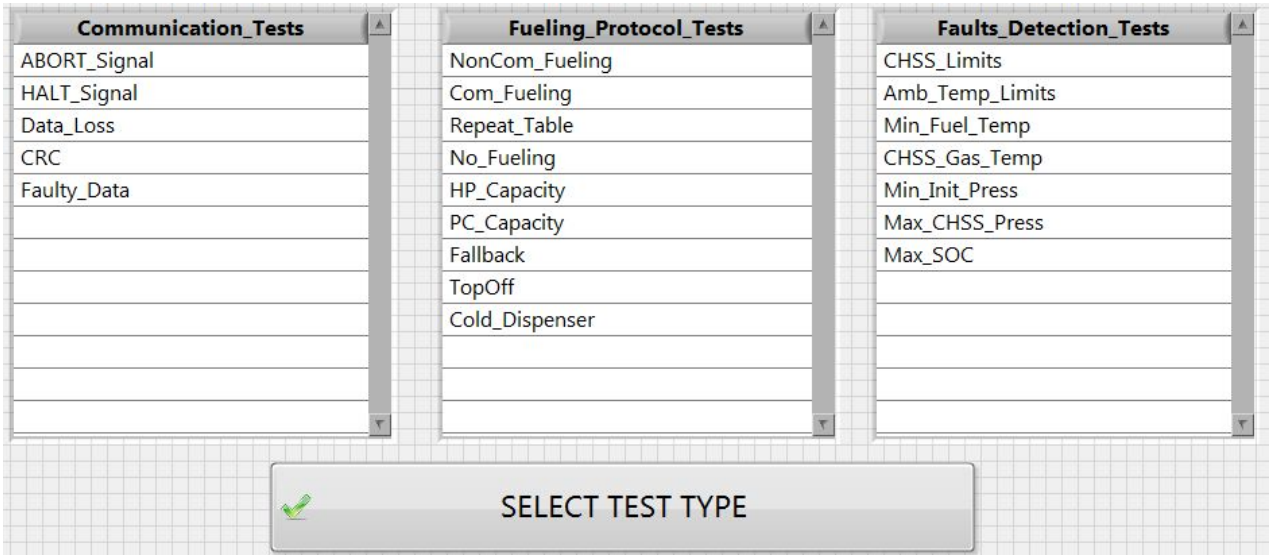
- C:\Users\Public\Public Documents\H2_Tester***TestLists*** folder use to select the test type and name.



For instance, **Fueling_Protocol_Tests** contains the following test types names:

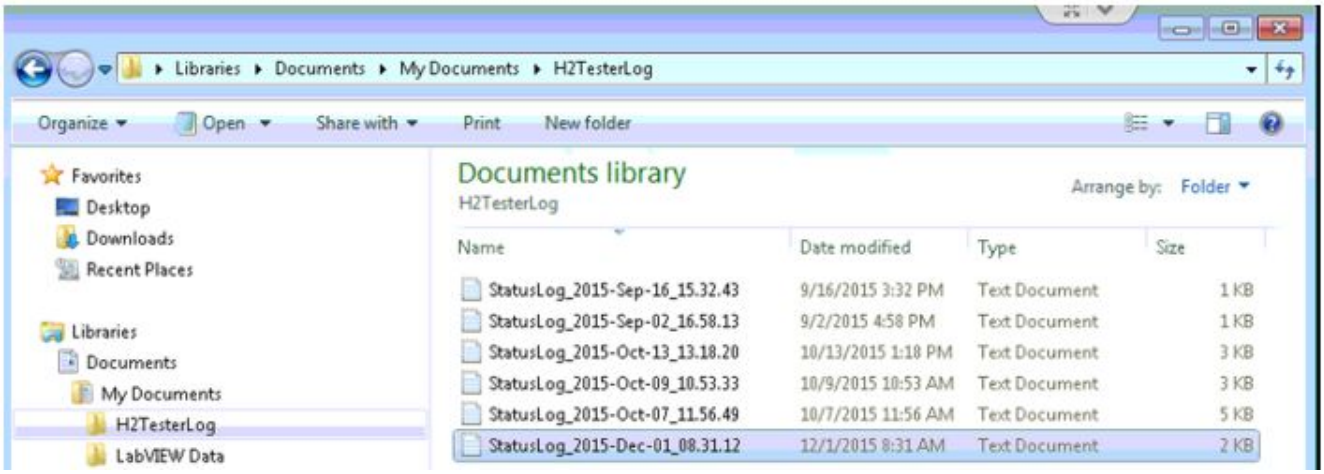


These names, are used to dynamically produce the test list in the test type pop up window.

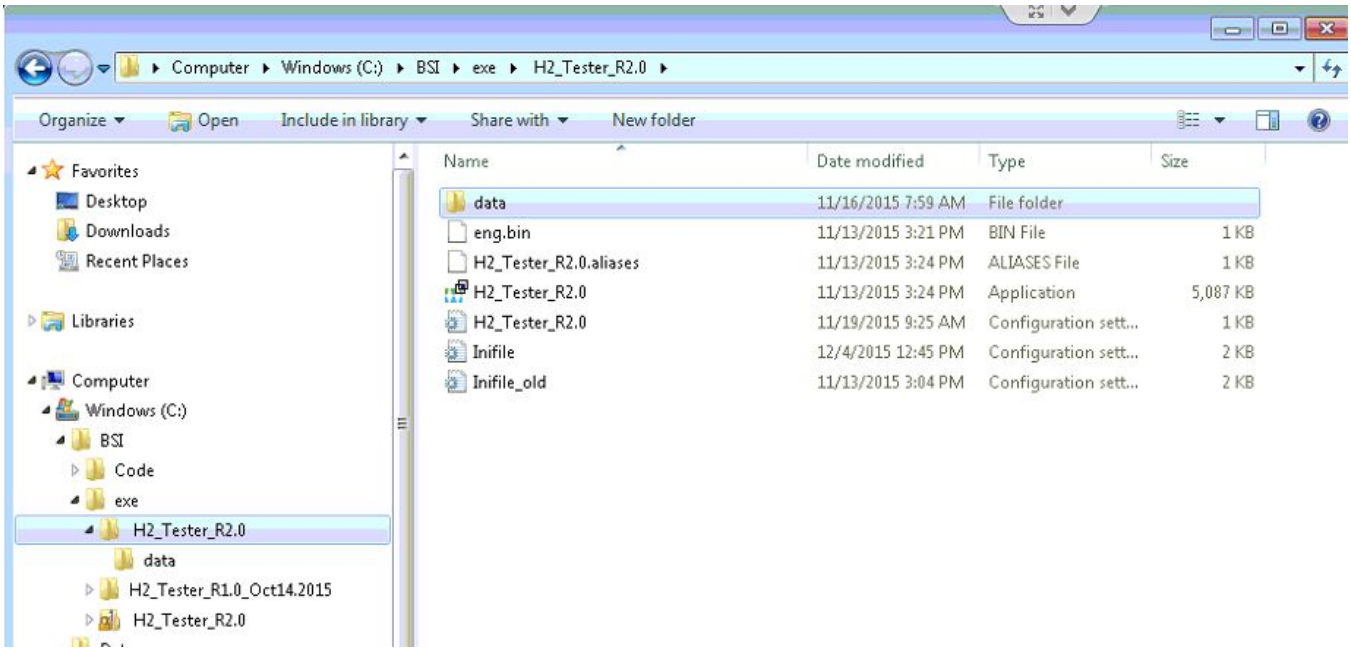


- C:\Users\Public\My Documents**H2TesterLog** folder contains status log that can be exported either when an error occur, or when the user required it via the

Save the content of the status windows to a text file. Is useful for troubleshooting



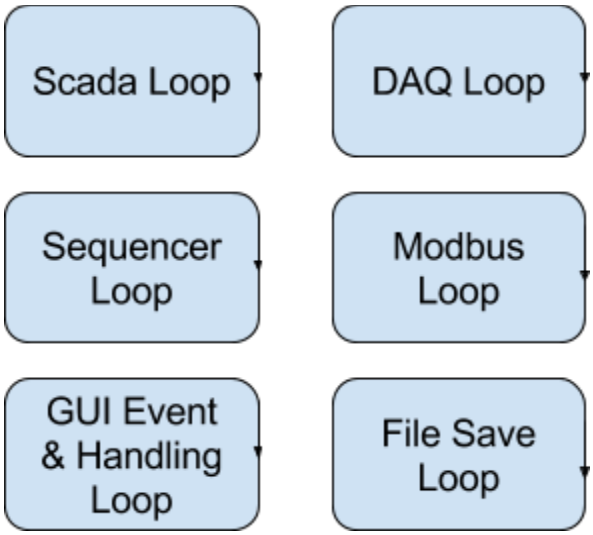
Executables



They are located on C:\windows\BSI\exe\
A shortcut from the .exe file can be copied onto the desktop. When new versions are added, make sure the last eng.bin (password) and Inifile.ini are ported over the new folder containing the new exe version so it recognizes the engineering login, and uses the latest settings.

Code Overview

Overall Architecture



The code uses a producer/consumer architecture with several parallel loops, each with specific tasks:

- **GUI Event & Events Handling Loop** : State Machine and Event Structure meshed together:

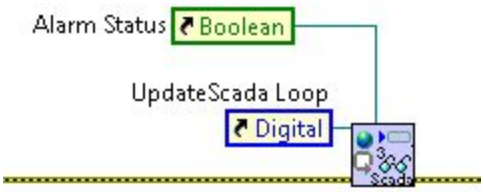
This loop manages all user interaction with the application as well as initialization steps. No critical hardware operation such as DO action must take place in this loop

- **cDAQ AI/DI Loop - Init DO** loop:
  This loops takes care of the Analog Input (AI) / Digital Inputs (DI) acquisition and Digital Outputs (DO) initialization (only). Acquired data are displayed from this loop and also placed into a queue so they can be shared with other loops for other operation such as file saving, alarms etc…

- **Modbus Loop**:
  As its name implies this loop focuses on the Modbus serial interface with the IRDI device.

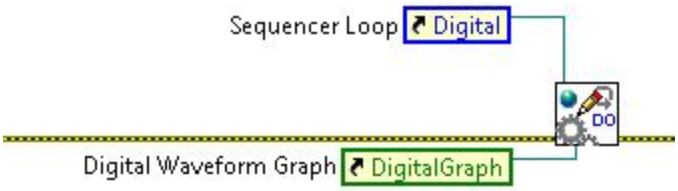
- **Save Loop:**
  This is where the data are saved to disk in two steps. A raw data file is used to always save new data. If the operator confirms this latest data set is to be kept, the raw data set is transferred over to a permanent user-data file.

- **Update SCADA loop:**
  Unlike the above loops that are coded explicitly on the main diagram, this loop is built inside a subVI.



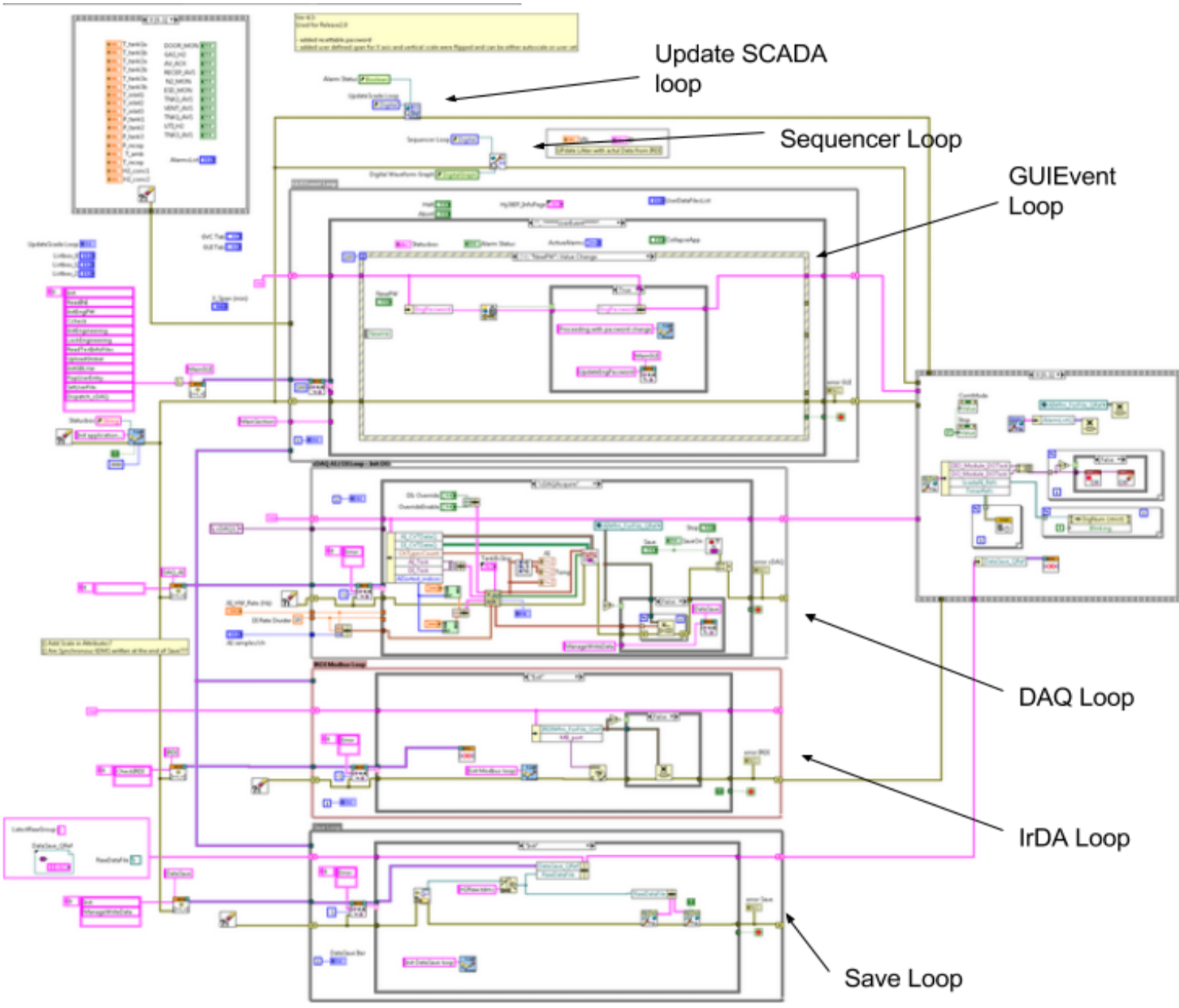
This loop takes care of displaying all the AI and DI indicators as per the acquired data from the cDAQ loop. It also checks for alarm levels. If an alarm is detected, resulting action(s) are handled by the Sequencer Loop.

- **Sequencer Loop:**



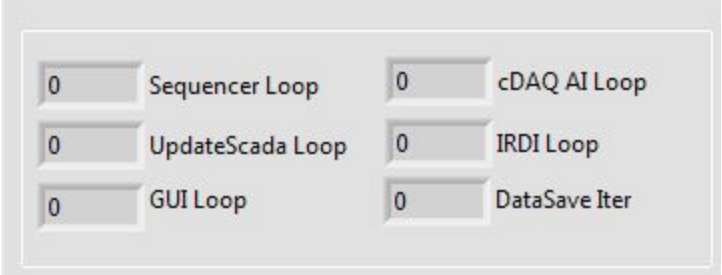
Also built inside a subvi, this loop takes care of all the actions that the application must perform in terms hardware output, mainly DOs. All the steps involved in Fueling, Defueling, Alarm Actions are handled by this loop since they involve opening and closing valves in a given sequence.

The image below provides a bird's-eye view of the code and where the loops reside.



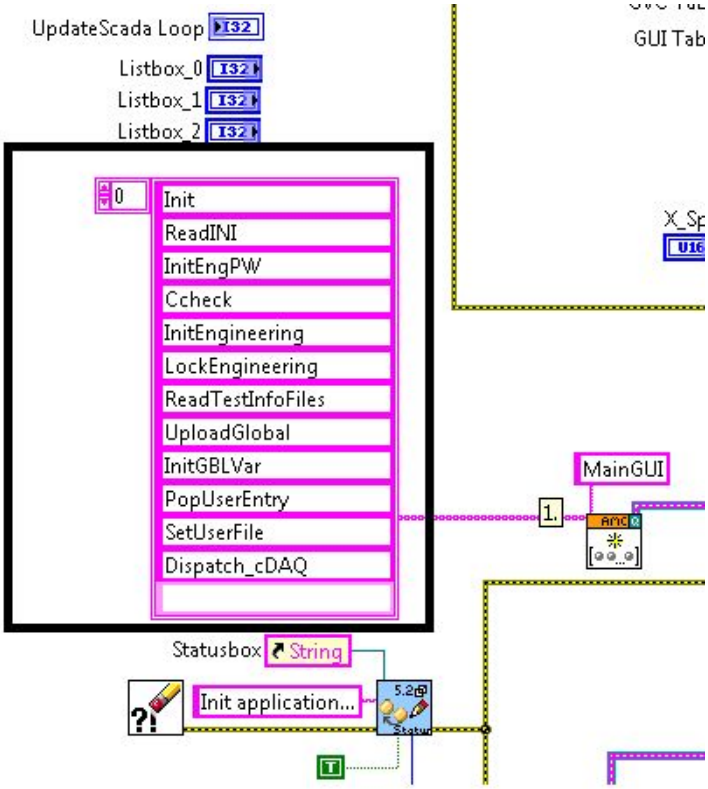
Loops Operation

Loops iterations can be traced under the Health Tab. Each loop has a counter that increments each time the loop iterates once.

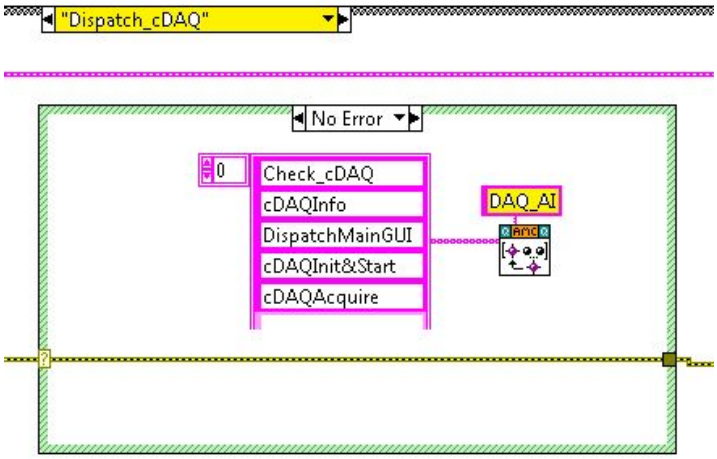


GUI Event Loop

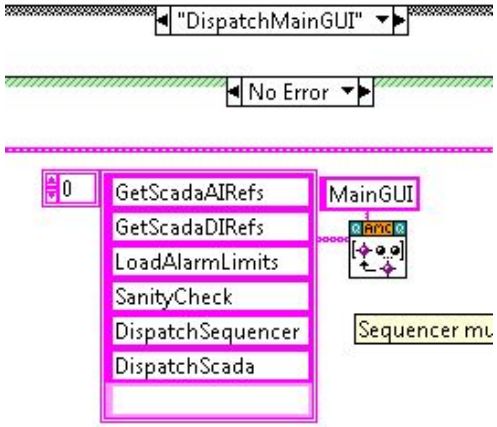
The initialization steps are preset by a constant array listing all the loop states that must operate first: Init, ReadINI, InitEngPW, Ccheck etc….



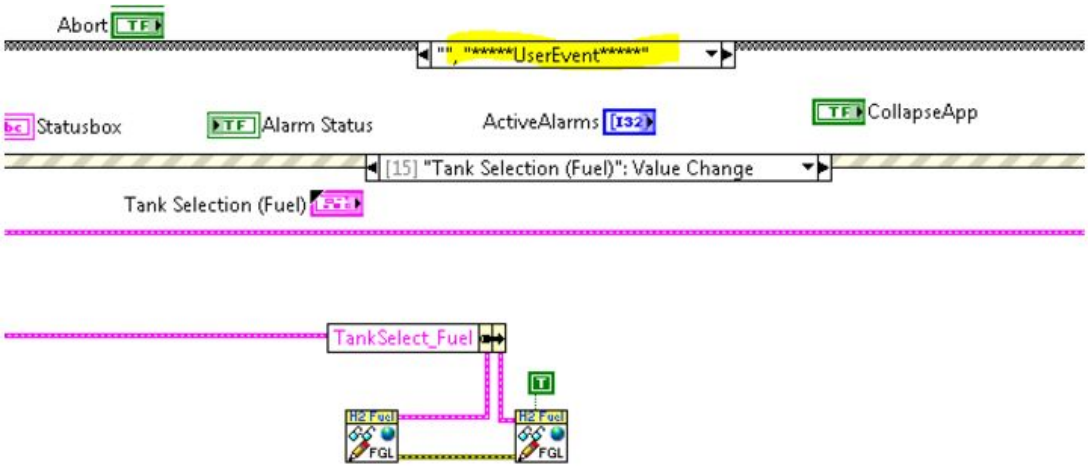
The last init step is **Dispatch_cDAQ**, and is shown below. In this step, the cDAQ acquisition loop is effectively set to initialize and start acquiring data.



Once the cDAQ information has run (cDAQInfo), note that the sequence reaches a state called **DispatchMainGUI**, which refers back to the **MainGUI** loop. At this stage, the loop is set to complete extra steps (GetScadaAIRefs, GetScadaDIRefs…) including the dispatch of both Sequencer and Scada loops.



Front that point on, most of the loop actions are derived from the user event structure that is nested under the "*****UserEvent*****" of the GUI Loop.



cDAQ AI/DI Loop - Init DO

All I/Os tasks are created in this loop. All AIs are hardware timed, while DIs are software time. The DOs are turned On or Off from inside the Sequencer loop based on the action to be accomplished (open or close valve(s) etc…) . Because the NI-9375 module is a "serial digital I/O module" , DIs have to be software timed.

Digital I/O Considerations for C Series Devices

Digital I/O module capabilities depend on the type of digital signals that the module can measure or generate and the chassis the module is used in.

The NI 9375[1], NI 9403, NI 9425, NI 9426, NI 9476, NI 9477, and NI 9478 are serial digital I/O modules.

The NI 9401, NI 9402, NI 9411, NI 9421, NI 9422, NI 9423, NI 9435, NI 9472, NI 9474, NI 9475, NI 9481, 9482, and NI 9485 are parallel digital I/O modules.

The following table shows what types of tasks these modules can perform in the different chassis.

Chassis	Tasks
NI cDAQ-9172	Serial and parallel modules: • Software-timed digital input/output tasks Parallel modules: • Hardware-timed digital input/output tasks (when used in slots 1 through 4) • Counter/timer tasks (when used in slots 5 and 6) • Accessing PFI signal tasks (when used in slots 5 and 6)
NI cDAQ-9138, 9139, 9171, 9174, 9178, 9181, 9184, 9188, 9188XT, and 9191	Serial and parallel modules: • Software and hardware-timed digital input/output tasks[2] Parallel modules: • Counter/timer tasks • Accessing PFI signal tasks (can be used in up to two slots)

Parent topic: Digital I/O

[1] The NI 9375 is not supported in the NI cDAQ-9172.

[2] Timed digital input/output restrictions:

- You cannot use parallel and serial modules together on the same hardware timed task.
- You cannot use serial modules for triggering.
- You cannot do both static and timed tasks at the same time on a single serial module.
- You can only do hardware timing in one direction at a time on a serial bidirectional module.

All AIs and DIs are synchronized (hardware timing) and fed into a queue acting as a Current Value Table holding the last value produced by the cDAQ. This data is produced at 10Hz, while the original AIs are sampled at 100Hz and averaged down to 10 Hz.

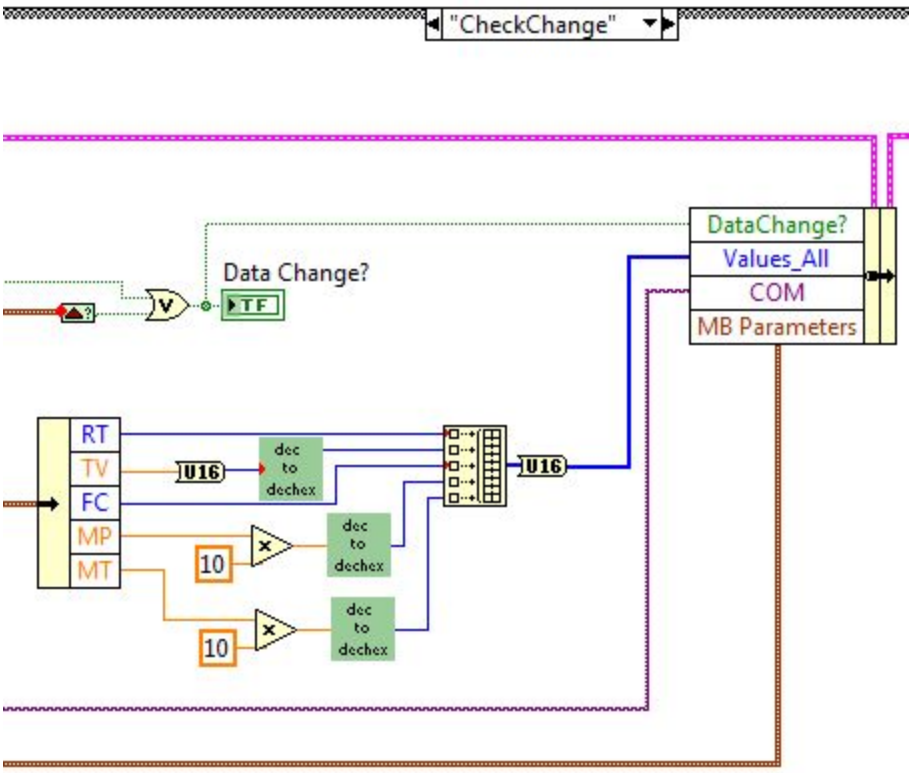
Modbus Loop

This loops has two main purposes:

- (1) is to push the SAE J2799 required parameters (VN, RT, TV, FC, MP, MT) as measured by cDAQ onto the IRDI controller via modbus. In turn, the IRDI controller, will produce this data as the required rate at the IR interface with the H2 dispenser's fuel line.
- (2) as the operator requires, command the IRDI hardware to create corrupt CRC strings meant to be injected in the communication stream with the fueling station under test.

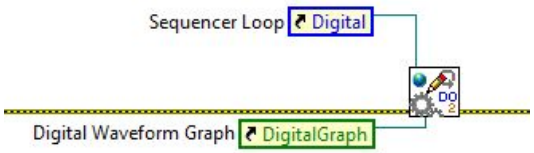
The LabVIEW application does not generate the corrupt CRC, nor it controls how these are generated. It only instructs the IRDI controller to produce the fake CRC via a command. The command mechanism is to write a given numeric in a predefined IRDI controller's modbus address.

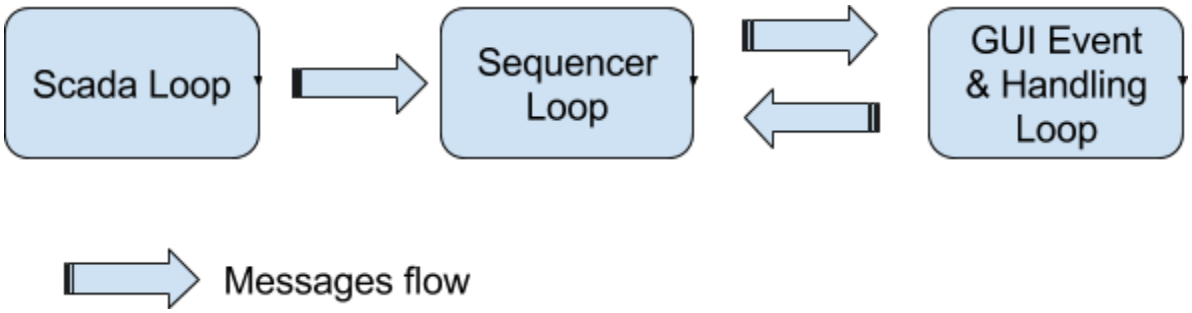
For overriding the cDAQ values passed to the IRDI controller, user-specified data value(s) (VN, RT, TV, FC, MP, MT) replace the actual measurement if requested.



Note that the modbus data are updated into the IRDI controller only if there is a change in one of the parameters. If no change occurs, no update is passed to the IRDI controller which will keep passing the unchanged parameters to the fuel dispenser unit.

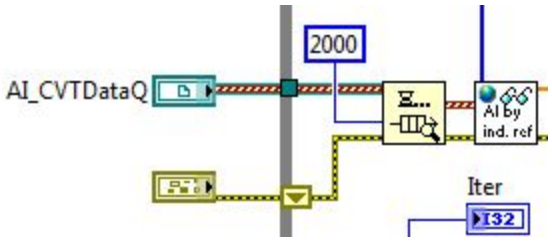
Sequencer Loop



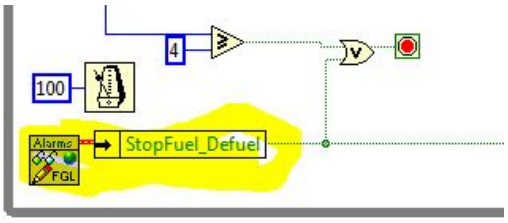


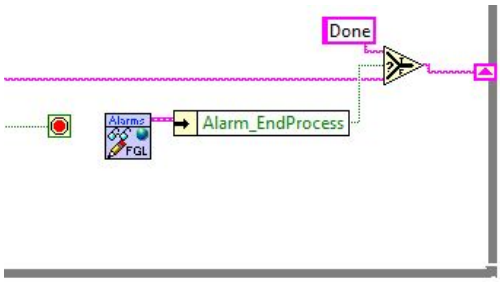
The sequencer loop receives command messages mainly from the GUI Event & Handling (i.e. Purge, Fuel, Defuel…). Also to a lesser extend, from the Scada Loop when an alarm level is detected and some DOs action must take place. Messages sent from the Sequencer Loop to the GUI Event & Handling are mostly status or test results at the end of a defuel, fuel (...) sequence(s).

At init time, the channel of interest (AI, DO) are tabulated based on their names, hence the recommendation that channel names should not be changed inside MAX. AI channels data are obtained from a Current Value Table (CVT) in the form of a Queue reference updated inside the cDAQ loop.

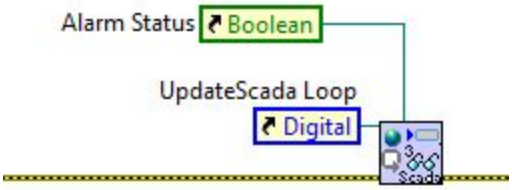


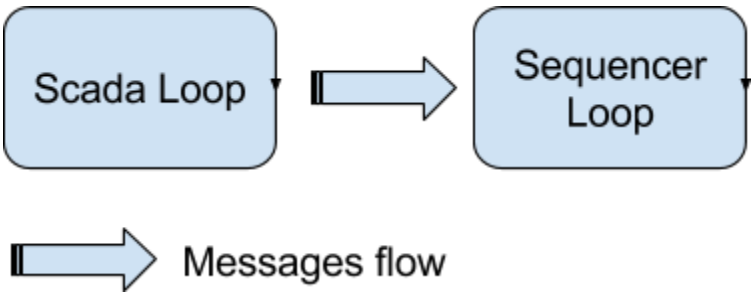
Note: The sequencer loop must not “hold” processing of alarms or important steps that can occur during operation. Still, some of the sequence, such as defuel or fuel, require some logic based on pressure reading varying with time. In these cases, while loops are used to read the analog values of the CVT and **must** include a global variable (LabVIEW Functional Global: FGV) to stop the loop in case of an emergency so that the code can break out of the loop.





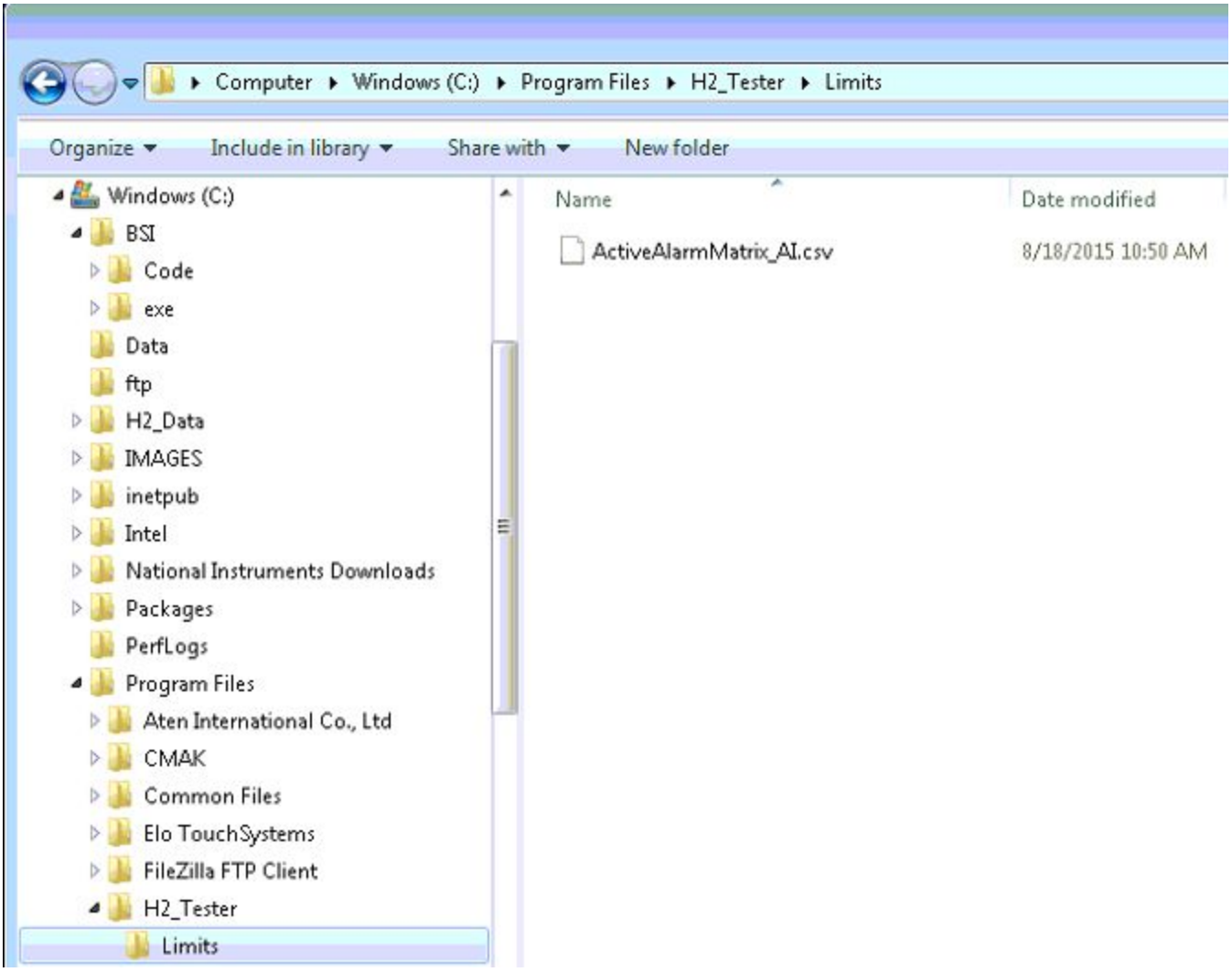
Scada Loop





The Scada loop, once initiated from the main GUI Event & Handling loop monitor the cDAQ AI and DI and push the CVT to the GUI indicators. It is also in charge of checking for the alarms. Most command messages flow from the Scada loop into the Sequencer loop to switch DOs in case of alarm.

Alarms



The following is based on the alarming matric file :

Alarm Parameters Example

Reset Point: it is an hysteresis
Latching: it does not get “reset” or ack by user.

Fault Description	**Alarm ID**	**Period when Alarm Active (Always, Idling, Fueling, Defueling, Purging, Tank Gas Transfer)**	**Cond.**	**Alarm Setpoints**	**Latching**	**Auto Reset**	**Fault Reset Setpoint**	**Alarm Priority/ Ranking**

High Warning	Point ID + A01	Fueling	>=	85		Y	80	3
High Alarm	Point ID + A02	Fueling	>=	88	Y		85	1
Low Warning	Point ID + A03	Fueling	<=	1		Y	2	3
Low Alarm	Point ID + A04	Fueling	<=	0.5	Y		0.5	1
Wiring Fault	Point ID + A05	Always	=	Max Scale		Y	80	2
Wiring Fault	Point ID + A06	Always	=	Min Scale	Y		2	2

Alarm Effect Summary

In this order the associated application's object names are:

Columns are searched by name, so the column position does not matter.
Rows on the other hand, for the headers must remain in place. Added Channels, if any, can be added as a consistent group of 6 parameters as in the previous table: High Warning, High Alarm etc…

Alarm DOs and GUI

Hardwired ESD Shutdown	**cDAQ ESD Shutdown**	**Audible Alarm**	**Visual Alarm**	**System Operational**	**HMI Screen Alarm Notification**	**HMI Screen Alarm Notification Locked**	**HMI pop up**
n/a	ESD_COM *	AUD_ALM	VIS_ALM	?	GUI (to be def)	GUI (to be def)	GUI (to be def)

Valves DOs - IRDI

IRDI Signal	RECEP_AV	TNK1_AV	TNK2_AV	TNK3_AV	VENT_AV

Valve DIs check for Valve DOs

Valve DO (action)	Valve status check (DI)
RECEP_AV	RECEP_AVS
TNK1_AV	TNK1_AVS
TNK2_AV	TNK2_AVS
TNK3_AV	TNK3_AVS
VENT_AV	VENT_AVS

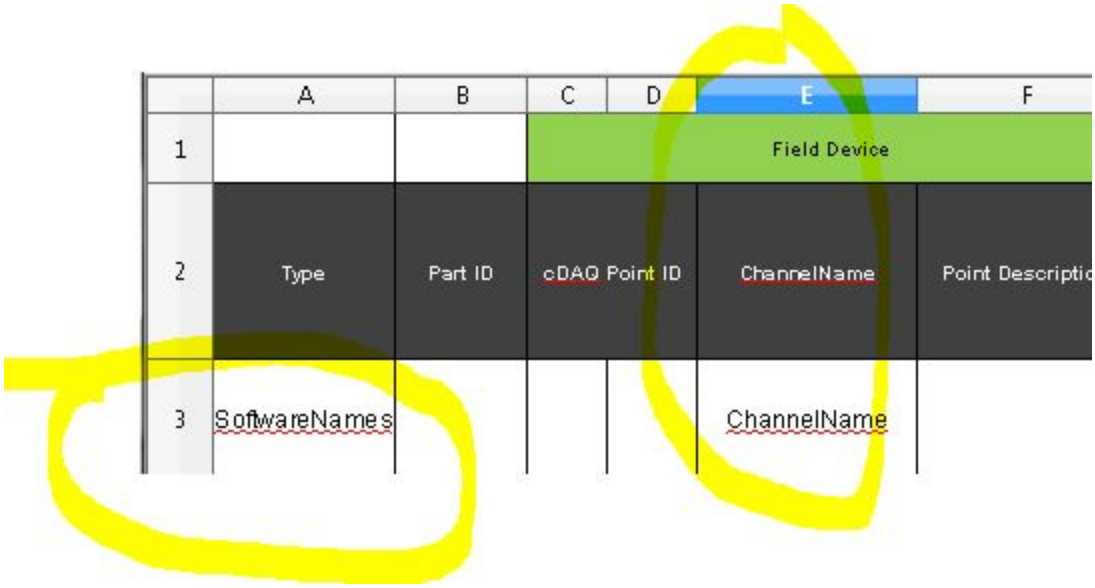
When a DO is changed, the status should be checked against its associated DI. Assume 2-3 sec before a change in DO is reflected in DI.

Important Note: There is an alarm priority field in case several alarms are in effect at the same time. The alarms priority is 1 for the highest, 2, 3 etc… for lowest. So as a result:

- Actions (actuation of DOs) for highest priority will take precedence over lowest level actions.
- All actions within a priority level are identical so no conflict can arise from a common priority level amongst several channels.

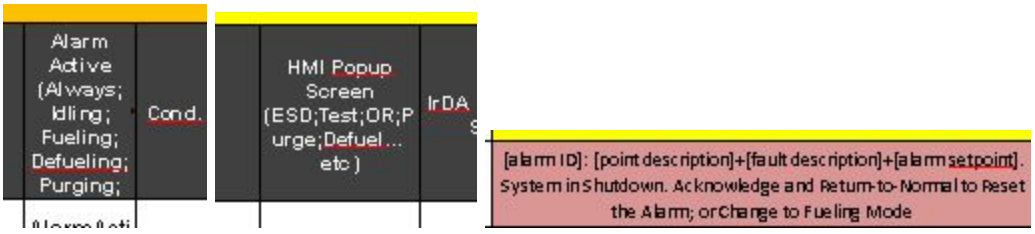
Alarm File Changes

These changes were required to migrate the original alarm matrix prepared by PowerTech into a comma delineated file used by the application to look up each alarm and associated properties.

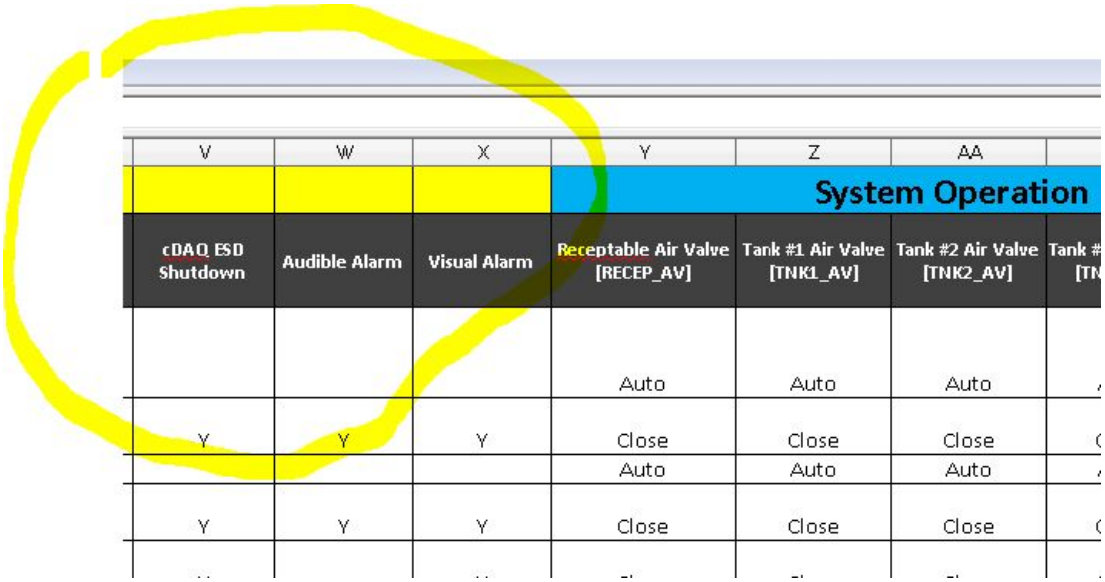


	A	B	C	D	E	F
1			Field Device			
2	Type	Part ID	cDAQ Point ID		ChannelName	Point Descriptio
3	SoftwareNames				ChannelName	

1. Add column called “ChannelName”
2. Add row called “SoftwareNames” (I’ll send the content later)
3. <u>Remove all commas</u> in description in headers and HMI text description.

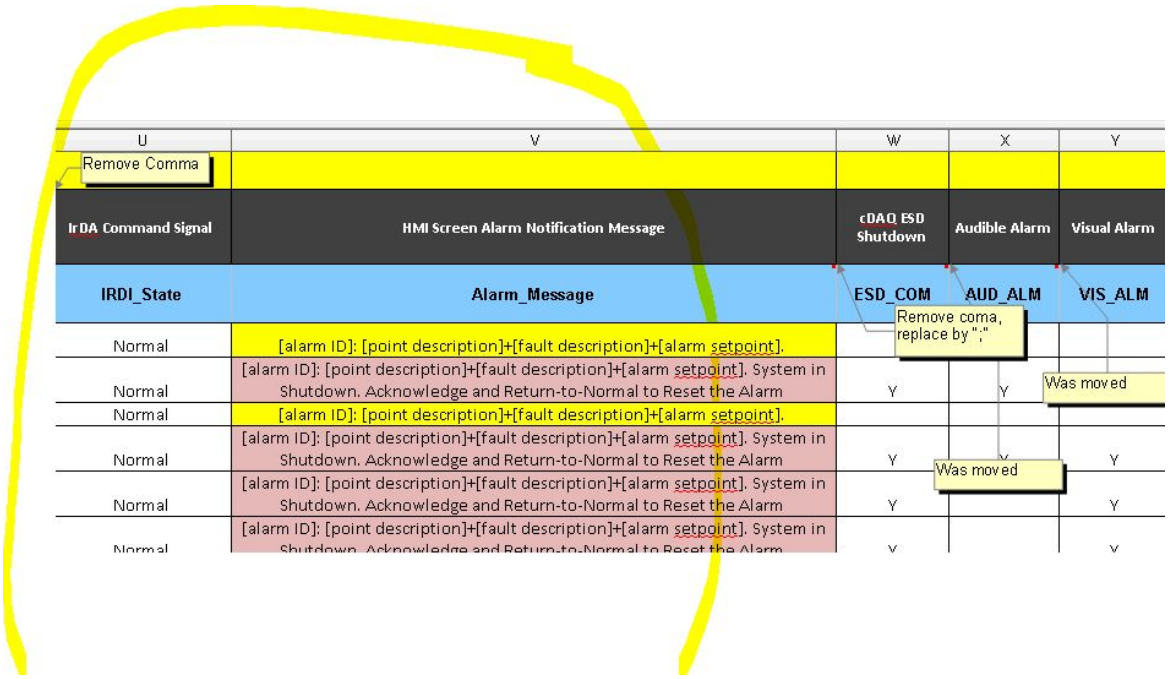


4. Add NI Channel Name in “ChannelName” column for each ID.
5. Move Columns “cDAQ ESD Shutdown”; “Audible Alarm”; “Visual alarm” next to the System operation columns, they may be replaced by semi-colums.



V	W	X	Y	Z	AA	
			System Operation			
cDAQ ESD Shutdown	Audible Alarm	Visual Alarm	Receptable Air Valve [RECEP_AV]	Tank #1 Air Valve [TNK1_AV]	Tank #2 Air Valve [TNK2_AV]	Tank # [TN
			Auto	Auto	Auto	
Y	Y	Y	Close	Close	Close	
			Auto	Auto	Auto	
Y	Y	Y	Close	Close	Close	

6. Move HMI Screen Alarm Message in column V



U	V	W	X	Y
Remove Comma				
IrDA Command Signal	HMI Screen Alarm Notification Message	cDAQ ESD Shutdown	Audible Alarm	Visual Alarm
IRDI_State	Alarm_Message	ESD_COM	AUD_ALM	VIS_ALM
Normal	[alarm ID]: [point description]+[fault description]+[alarm setpoint].			
Normal	[alarm ID]: [point description]+[fault description]+[alarm setpoint]. System in Shutdown. Acknowledge and Return-to-Normal to Reset the Alarm	Y	Y	
Normal	[alarm ID]: [point description]+[fault description]+[alarm setpoint].			
Normal	[alarm ID]: [point description]+[fault description]+[alarm setpoint]. System in Shutdown. Acknowledge and Return-to-Normal to Reset the Alarm	Y		Y
Normal	[alarm ID]: [point description]+[fault description]+[alarm setpoint]. System in Shutdown. Acknowledge and Return-to-Normal to Reset the Alarm	Y		Y
Normal	[alarm ID]: [point description]+[fault description]+[alarm setpoint]. System in Shutdown. Acknowledge and Return-to-Normal to Reset the Alarm	Y		Y

Remove coma, replace by ","

Was moved

Was moved