System Overview, Remote Networked Estop System for HET

January 6, 2022

George Damm

Overview

The Remote Networked Estop System (System) consists of a set of PLCs that integrate into HET's Estop system. The purpose of this System is to provide the on-site Telescope Operators the ability to assert an HET Estop while remotely controlling nighttime observing from their homes. This system does not meet the rigorous requirements necessary to protect human life. It is only intended to protect HET equipment while it is being operated remotely. See Figure 1 below.

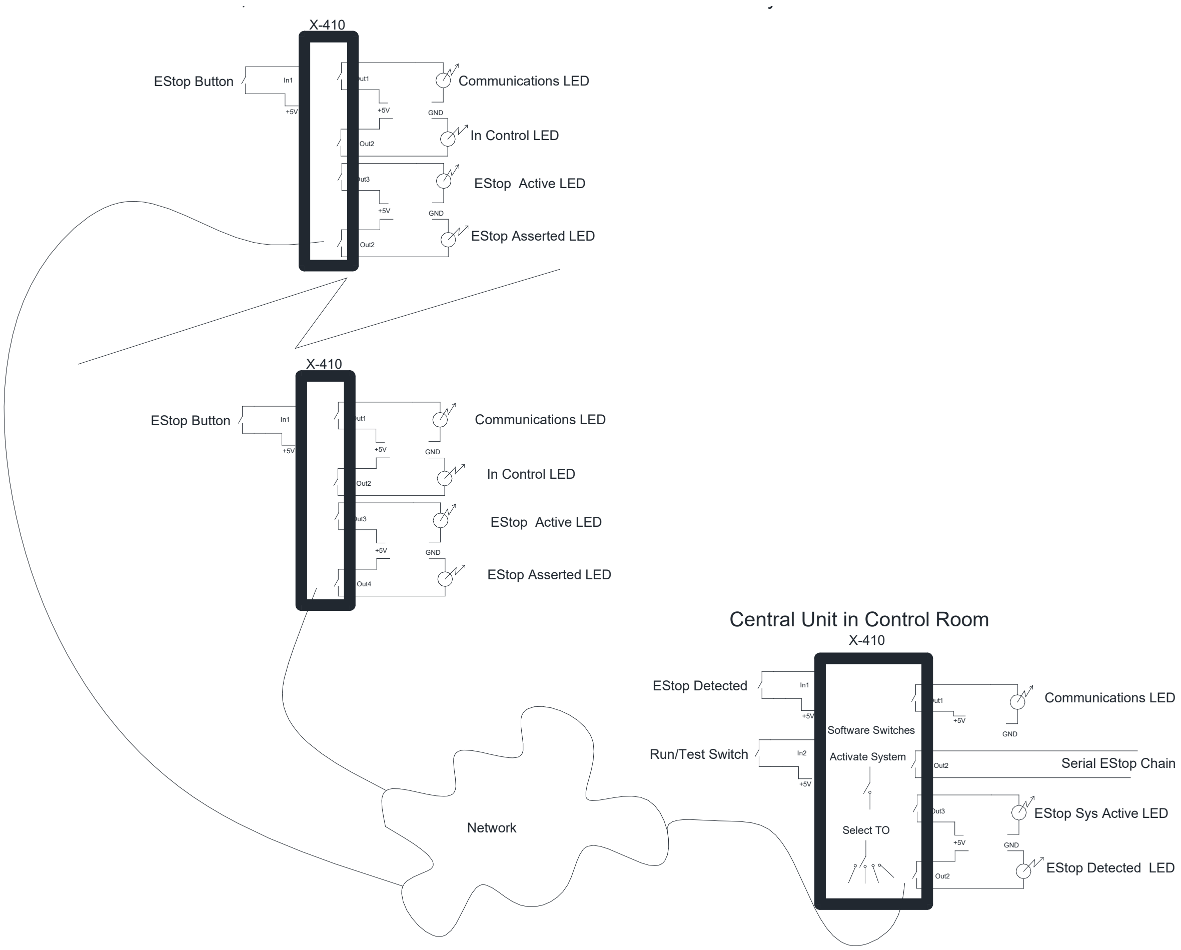


Figure 1. Diagram showing the Central and Remote Units, part of the Remote Networked Estop System.

The System consists of 3 Remote units and 1 Central unit. The Remote units are installed at each of the three houses of the TOs that live on-site. The Central unit is located in the HET Control Room and interfaces with 1) each of the 3 Remote units via the internet and to 2) the HET Estop system through direct wiring. It is important to note the distinction between the two systems. The System defined here is the Remote Networked Estop System. The HET Estop system is a functional hardware based system already in use at the HET.

The Remote unit interface consists of 4 status LEDs and one Estop pushbutton. These LEDs indicate 1) active communications to the Central unit, 2) whether the particular Remote unit has control of the System, 3) operational status of the System, and 4) presence of an HET Estop. The Estop pushbutton can be pressed to assert an Estop at the Central unit which then assert an Estop on the HET Estop system.

The Central unit consists of 3 status LEDs and 1 Run/Test toggle switch (shown if Figure 3 below). These LEDs indicated: 1) active communications to whichever Remote unit is in control, 2) operational status of the System and 3) the presence of an HET Estop. The Run/Test toggle switch is defaulted to Run mode but can be moved to Test mode to check System operations.

Connectivity

Key to the operation of the System is the Xytronix X-410, a web enabled programmable controller that we will refer to as a Programmable Logic Controller (PLC). Both the Remote and Central units have one of these PLCs to provide **Web Communications**, actuate the **Programmed Logic** in each, and read and control the **Variables** and input/output defined by the input switches and the indicator LEDs. In order to properly setup the **Interconnected Program**, certain programming rules, variable definitions and device setup steps must be followed.

Web Communications

The Remote and Central units communicate with each other through shared variables. This information is passed through packets over TCP-IP. In order to communicate across the various networks, each PLC much used hard coded static IP addresses from the other PLCs. Since the communications occurs across different networks and security between each of these networks and the outside world must be maintained, the technique of port forwarding is used. HET also uses a firewall whitelist to allow the 3 Remote units to communicate to sites within the HET network. Figure 2 below shows how these techniques are used.

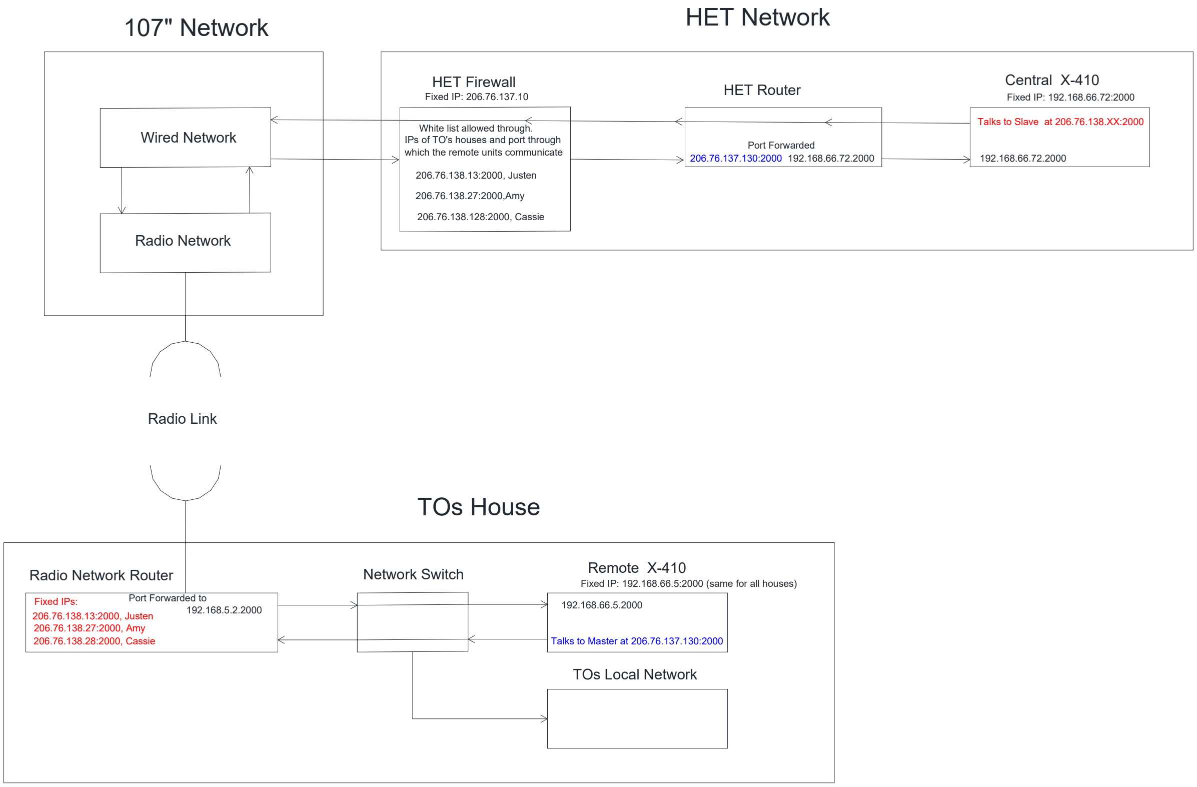
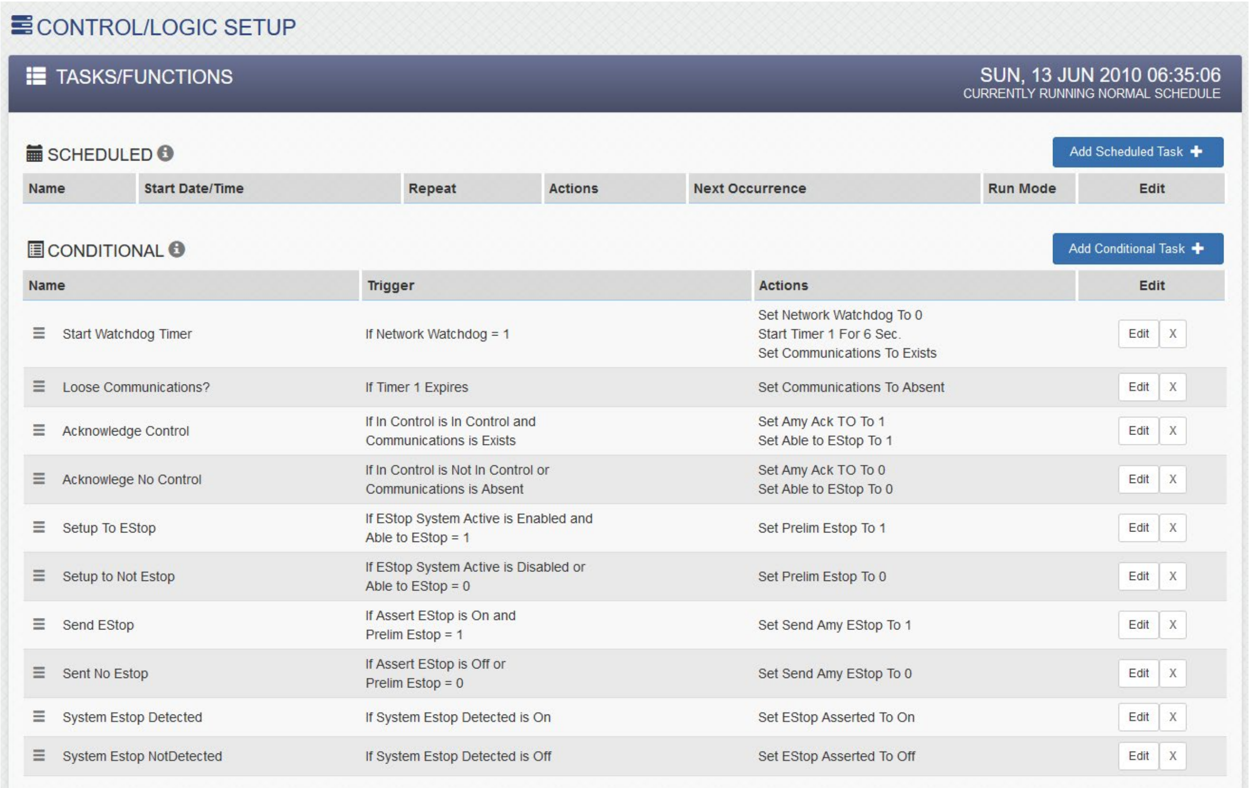


Figure 2. Diagram showing the port forwarding and firewall used for Remote to Central unit communications.

Programmed Logic

The X-410 PLCs in each Central and Remote unit run a series of logic steps that looks at the various input (switches, HET Estop) and variables, and determines which outputs (LED indicators, HET EStop) to assert. Shown below is the logic for each of the Remote units. The program is limited to IF statements, and using AND/OR operations. The IF statements can only read 2 variables for input and modify 3 output variables. These limitations will require multiple programming steps to accomplish what may be done in a single text based programming language. More complex program operations can be implemented by writing a script in an external editor and importing that into the X-410.



CONTROL/LOGIC SETUP

TASKS/FUNCTIONS

SUN, 13 JUN 2010 06:35:06
CURRENTLY RUNNING NORMAL SCHEDULE

SCHEDULED

Add Scheduled Task

Name	Start Date/Time	Repeat	Actions	Next Occurrence	Run Mode	Edit

CONDITIONAL

Add Conditional Task

Name	Trigger	Actions	Edit
Start Watchdog Timer	If Network Watchdog = 1	Set Network Watchdog To 0 Start Timer 1 For 6 Sec. Set Communications To Exists	Edit X
Loose Communications?	If Timer 1 Expires	Set Communications To Absent	Edit X
Acknowledge Control	If In Control is In Control and Communications is Exists	Set Amy Ack TO To 1 Set Able to EStop To 1	Edit X
Acknowlege No Control	If In Control is Not In Control or Communications is Absent	Set Amy Ack TO To 0 Set Able to EStop To 0	Edit X
Setup To EStop	If EStop System Active is Enabled and Able to EStop = 1	Set Prelim Estop To 1	Edit X
Setup to Not Estop	If EStop System Active is Disabled or Able to EStop = 0	Set Prelim Estop To 0	Edit X
Send EStop	If Assert EStop is On and Prelim Estop = 1	Set Send Amy EStop To 1	Edit X
Sent No Estop	If Assert EStop is Off or Prelim Estop = 0	Set Send Amy EStop To 0	Edit X
System Estop Detected	If System Estop Detected is On	Set EStop Asserted To On	Edit X
System Estop NotDetected	If System Estop Detected is Off	Set EStop Asserted To Off	Edit X

Figure 3. Logic steps used to control a Remote Unit.

Variables

Each X-410 has the ability to define variables of several types, some linked to input/output (I/O) hardware while other reside as digital entities within the PLC. The hardware I/O includes relays, digital inputs and 1-wire devices. Within the PLC are registers and timers. For our purposes, we only use relays, digital inputs, registers, timers and a 1-wire temperature sensor, used to monitor the internal temperature of each unit. Figure 4 below shows a screen shot of the Xytronics interface that allows the variables to be defined and some mapped to physical input/output (I/O)

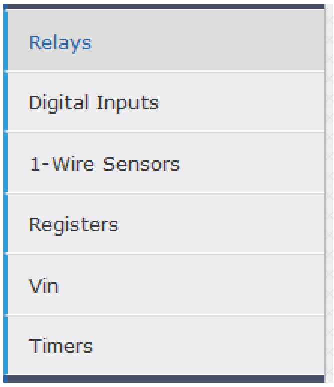
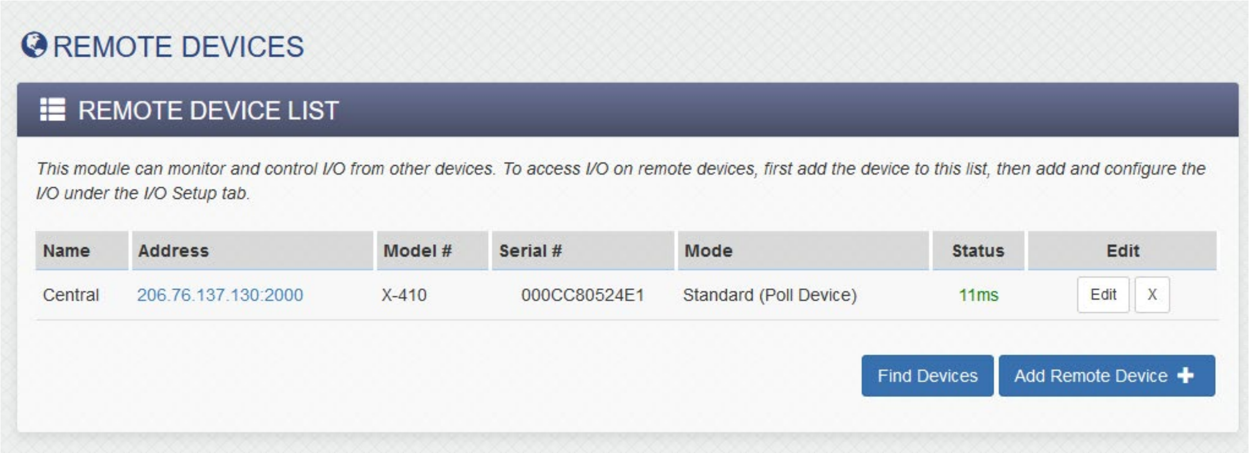


Figure 4. Selection screen for defining variables and I/O.

Key to the inter-unit communications and operations is the ability to share variables. This is done by mounting remote PLCs. Once mounted, then the remote device can share its variables with the local devices. These variables must be named locally and need not be designated with the same name as the remote device. Figure 5 below shows a screen shot of a mounted remote device.



This module can monitor and control I/O from other devices. To access I/O on remote devices, first add the device to this list, then add and configure the I/O under the I/O Setup tab.

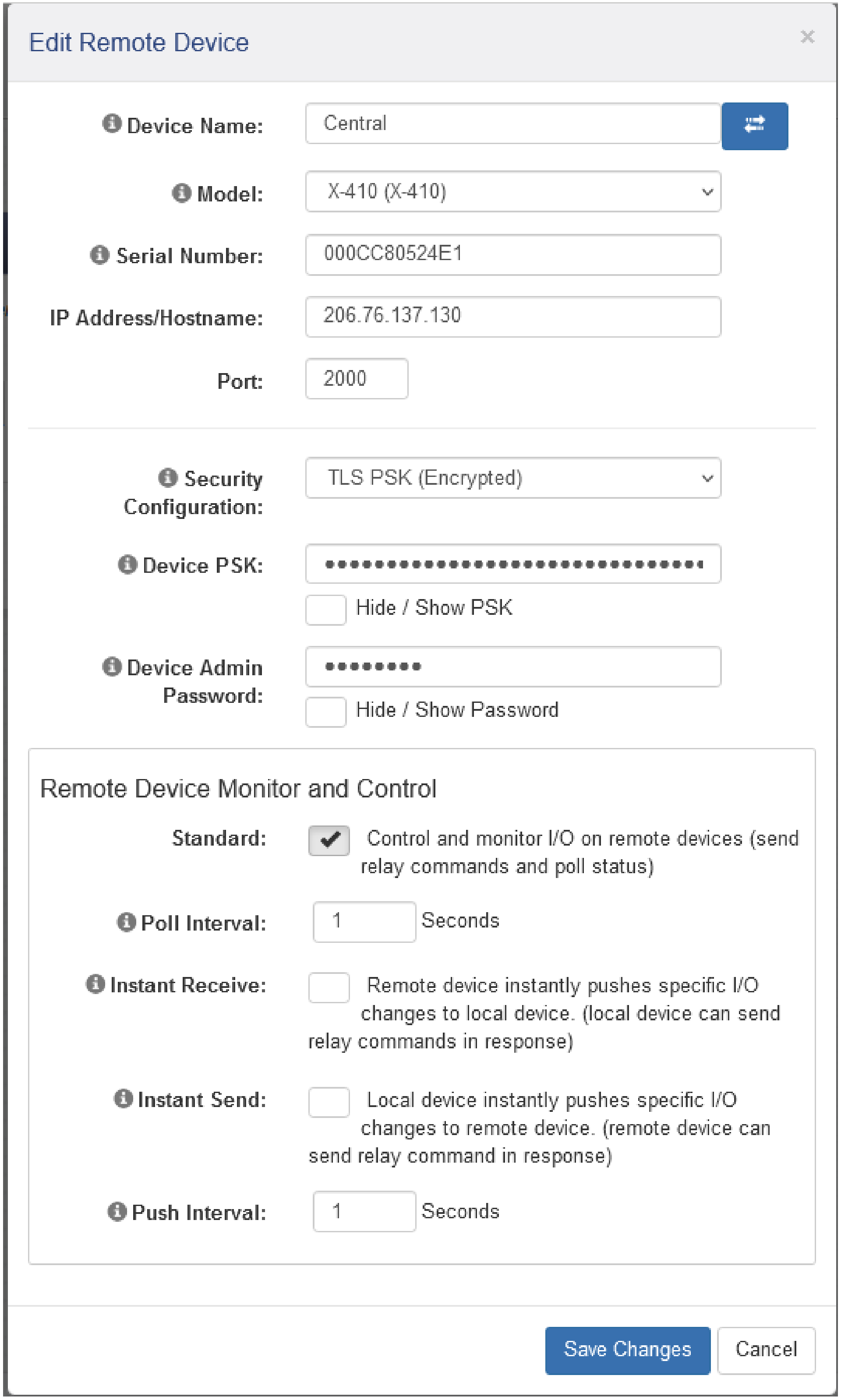
Name	Address	Model #	Serial #	Mode	Status	Edit
Central	206.76.137.130:2000	X-410	000CC80524E1	Standard (Poll Device)	11ms	Edit X

Figure 5. View of the X-410 GUI showing a connected remote device.

Interconnected System

It was only through trial and error and several discussions with Xytronix engineers that we were able to successfully complete this project. The critical information that we learned is not published in their documentation and will be detailed here. This information dealt with how the interconnected devices communicate with each other and informs the user in how to define the programming and variables.

Remote Device Settings: When defining the parameters as to how remotely connected PLCs are handled, there are several options with respect to how shared variables are processed. Figure 6 below shows this detail. In the section entitled Remote Device Monitor and Control, one has the option of selecting a *Standard* communications option. With this option, one must also select a poll and push interval. This defines the cadence as to when the remote variables are read by the local unit and when the local variables get sent to remote units. The present selection is one recommended by the Xytronix engineers but without a background as to why it worked. Their only comment was that this is the way that it has always worked successfully. They recommended that we not select *Instant Receive* and *Instant Send*. This will tend to greatly increase network traffic between connected devices and could be one reason that we would occasionally have stalled timers that were shared between devices.



Edit Remote Device

Device Name: Central

Model: X-410 (X-410)

Serial Number: 000CC80524E1

IP Address/Hostname: 206.76.137.130

Port: 2000

Security Configuration: TLS PSK (Encrypted)

Device PSK: ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

□ Hide / Show PSK

Device Admin Password: ●●●●●●●●●

□ Hide / Show Password

Remote Device Monitor and Control

Standard: ☑ Control and monitor I/O on remote devices (send relay commands and poll status)

Poll Interval: 1 Seconds

Instant Receive: □ Remote device instantly pushes specific I/O changes to local device. (local device can send relay commands in response)

Instant Send: □ Local device instantly pushes specific I/O changes to remote device. (remote device can send relay command in response)

Push Interval: 1 Seconds

Figure 6. GUI page showing the settings for remoted connected X-410 units.

Variable Location: When a variable must be shared by two X-410 units it is important to understand which unit is writing the variable and which units are reading the variable. A systemic approach must be taken to optimize which X-410 unit is holding the variables. Advice from Xytronix engineers suggest that 1) the X-410 unit that is writing a variable that will be used in logic on another X-410 unit should map the variable from the remote onto the writing unit. When this variable is written, the writing process will automatically update this remote variable. The alternative is that 2) the reading X-410 would map the variables from the writing X-410 onto itself. There is a subtle but critical difference between these two methods. In method two, the writing process does not automatically update the variables that are read on the other X-410. Instead a separate *Remote Device* process must be enlisted to update these variables. Method 2 was very problematic in its operations in that it involves much more inter-device communications, especially with the timing variable that must be change continuously. It was not until Method 1 was implemented that the program became stable.