Conf-950564--2 ANL/IFR/CP--83964

EBR-II SECONDARY SODIUM LOOP PLUGGING TEMPERATURE INDICATOR CONTROL SYSTEM UPGRADE

Reed B. Carlson, Roger L. Gehrman
Integral Fast Reactor Operations Division
Argonne National Laboratory
P. O. Box 2528
Idaho Falls, Idaho 83403-2528

ABSTRACT

The Experimental Breeder Reactor II (EBR-II) secondary sodium coolant loop Plugging Temperature Indicator (PTI) control system was upgraded in 1993 to a real-time computer based system. This was done to improve control, to remove obsolete and high maintenance equipment, and to provide a graphical CRT based operator interface. A goal was to accomplish this inexpensively using small, reliable computer and display hardware with a minimum of purchased software. This paper describes the PTI system, the upgraded control system and its operator interface, and development methods and tools. The paper then assesses how well the system met its goals, discusses lessons learned and operational improvements noted, and provides some recommendations and suggestions on applying small real-time control systems of this type.

INTRODUCTION

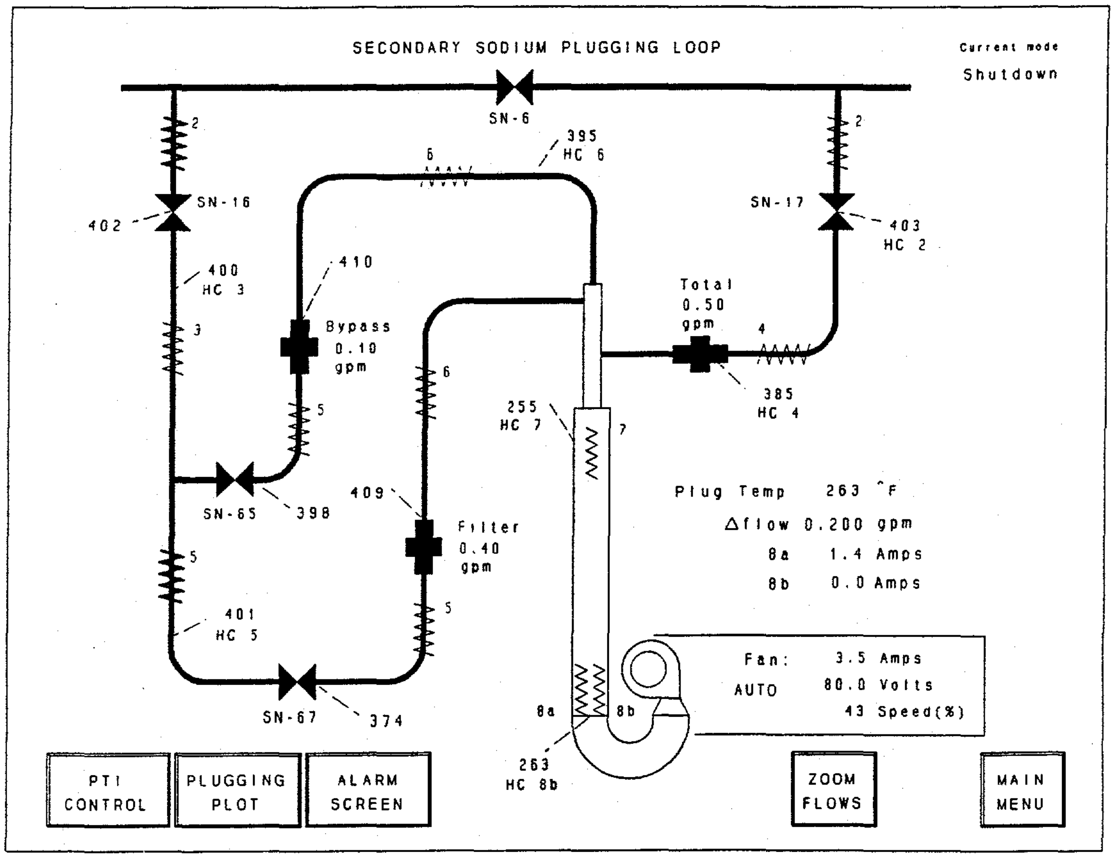
Plugging Temperature Indicator System

EBR-II contains two sodium coolant loops - a primary loop and a secondary loop. The primary loop transfers heat from the reactor core to the secondary loop which, in turn, transfers the heat to the power plant steam system.

The secondary PTI is used to determine the temperature at which a stream of liquid sodium from the secondary coolant loop starts to plug a stainless steel filter element. This temperature provides an indication of the level of dissolved impurities in the sodium. The lower the temperature at which the sodium plugs the filter, the lower the impurity level. It is important to keep the sodium impurity level low to prevent flow and heat transfer restricting particulate buildup in piping and heat exchangers.

* Work supported by the U.S. Department of energy, Reactor Research Technology under Contract NO. W-31-109-ENG-38.

The PTI consists of sodium piping, five flow control valves, nine heaters, a cooling fan, a filter element, three flow transmitters, 27 temperature sensors, and a control system. Figure 1 shows an overall view of the PTI system as it appears on one of the operator interface graphic screens. The valves are labelled SN-6 through SN-67, the heaters are indicated by the resistor symbols, and the flow transmitters are labelled Total, Bypass, and Filter. The apparatus with the fan and heaters 8a and 8b on its lower end consists of an outer cooling air manifold, an inner sodium supply pipe, and two inner pipes for outgoing sodium flow. The sodium supply is through the total flowmeter, the outgoing flow is through the pipes that contain the bypass and filter flowmeters. The filter element is located in the area of heaters 8a and 8b in the pipe that goes to the filter flowmeter.



Current mode
Shutdown

Total
0.50
gpm

Bypass
0.10
gpm

Filter
0.40
gpm

Plug Temp 263 °F
Δflow 0.200 gpm
8a 1.4 Amps
8b 0.0 Amps

Fan: 3.5 Amps
AUTO 80.0 Volts
43 Speed(%)

Figure 1: PTI Process Summary

When the system is in operation sodium enters through valve SN-17, flows through the total flowmeter, down to the area of the fan and heaters 8a and 8b. From there the sodium flows upward through the two outgoing flow paths, one flow path through the filter element and the filter flowmeter and the other path through the bypass flowmeter. The sodium then exits the system through valves SN-65, SN-67, and SN-16. The filter flowmeter measures the amount of sodium flowing through the filter element. As the temperature of the filter element changes through heating or cooling, the flow through the filter element varies. As the filter is cooled, sodium deposits on the filter element, restricting flow through the filter and increasing flow through the bypass pipe. As the filter is heated, the sodium deposits on the filter element melt, increasing flow through the filter and decreasing flow through the bypass pipe. This action, when appropriately controlled, results in an accurate determination of the sodium plugging temperature.

During operation the piping temperature is maintained by pipe heaters using on/off control with dead-band. The filter is heated by heaters 8a and 8b and cooled by the fan. Heater 8b is controlled by a Proportional-Integral-Derivative (PID) control function using filter temperature as its control variable. Heater 8a and the fan are controlled by another PID control function using delta-flow as its control variable. The delta-flow is the difference between the filter flow and the bypass flow. The fan is controlled inversely with the heater, as heater control output increases, fan output decreases. The valves are adjusted manually to maintain a nominal total sodium flow through the system. All the temperature, flow, fan, and heater current inputs are continuously monitored and alarmed.

The control system operates the PTI system through six modes of operation. The modes are shutdown, startup - heater 2, startup - continue, preheat, operate, and plugging run. These modes allow the system to be shutdown, heated up to where the first valves can be opened, heated to where sodium can flow through the entire system, maintained near operating temperature, operated with filter heater 8b at a setpoint of 580 degrees F, and, finally, operated where filter heater 8a and the fan are controlled with the setpoint of -0.05 delta-gpm to determine the plugging temperature.

Control System Upgrade

The old control system consisted of many discrete parts, including temperature and current indicators, on/off switches, rheostats, two PID controllers, and a circular chart type controller. The system was manually operational intensive, was subject to frequent breakdowns, and was difficult to calibrate and maintain. Certain components had become obsolete and, as a result, maintaining adequate numbers of spare parts was a problem.

The system was upgraded to improve reliability and accuracy, to reduce maintenance, to ease troubleshooting and calibration, to increase operational flexibility, to provide more continuous monitoring and alarming of system inputs and parameters, and to provide an improved operator interface. It was a goal to do this in a cost effective manner, utilizing a real-time computer system with a color graphical CRT based operator interface. The PTI system is small enough that it could not justify the more expensive workstation based graphical interfaces and software and separate real-time or PLC based front-ends that were used on larger systems. As a result, the upgraded control system consists of an STD Bus based computer with a single CPU board, several analog and digital I/O boards, and a color terminal with a trackball.

CONTROL SYSTEM AND ITS OPERATOR INTERFACE

Computer Control System

The computer control system consists of a V53 (comparable to an 80286) microprocessor based CPU board with a 80287 math coprocessor, four 16 bit analog input boards, a 12 bit analog output board, an open collector digital output board, a digital input board, and a watchdog timer board. The color graphic terminal is a DEC VT340 terminal that is operated in its color graphic mode. The terminal mouse has been replaced with a trackball. The computer is connected to the PTI system through solid state and electro-mechanical relays, several signal conditioning modules, current transformers, thermocouples, magnetic flowmeters, a voltage controlled power supply, and various filtering circuits.

The CPU board contains the entire software that controls and monitors the PTI system and drives the VT340 terminal. The software is contained in two flash memory IC's. The computer system operates in an interrupt driven mode with no operating system. The display software utilizes the ReGIS graphics language that is built into the terminal. A high level set of flowcharts for the software is shown in Figure 2.

As can be seen from the flowcharts, the software is interrupt driven except for the VT340 screen handler block. On a computer system startup or reset the communications and timer hardware is set to generate the interrupts and the system then proceeds to the first graphic terminal operator interface screen. The real-time control for the process is done at every other 1/2 second interrupt. The remaining two interrupts are for analog input signal conditioning and for handling any trackball or keyboard input.

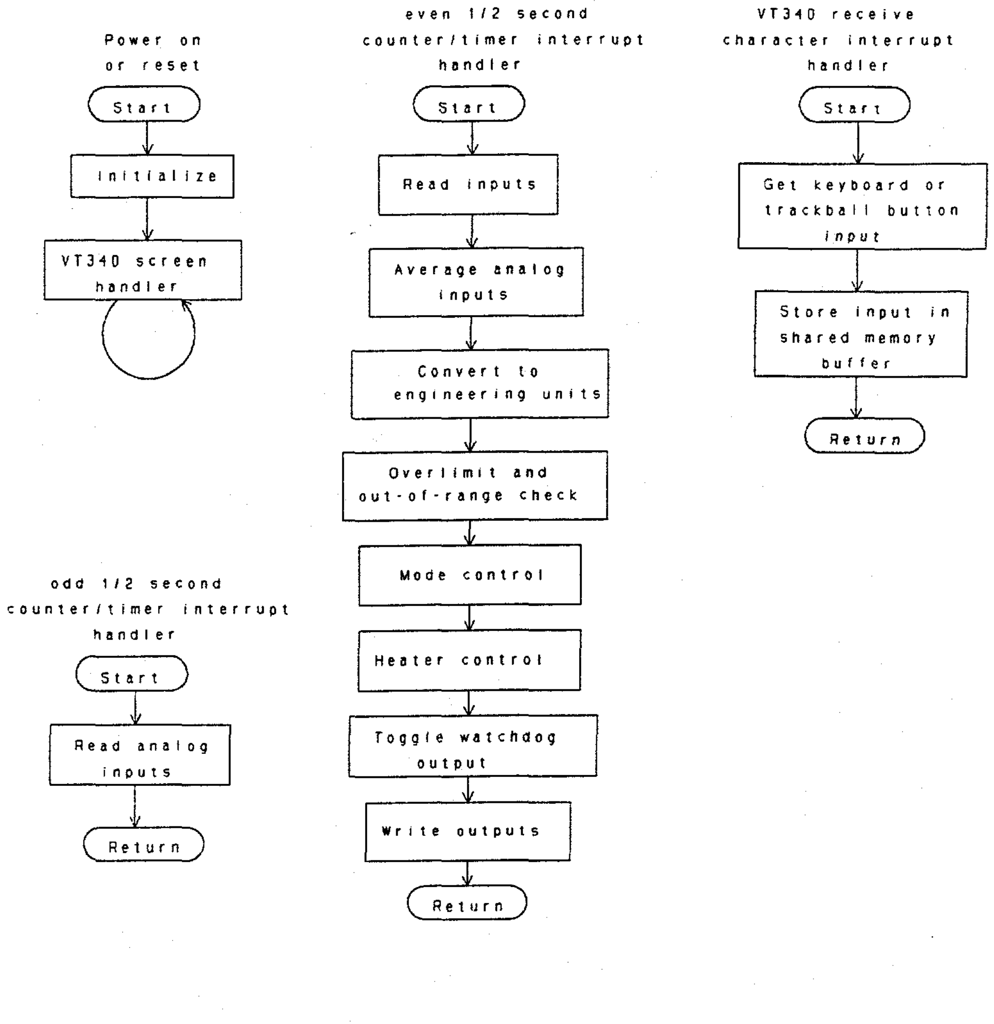

Power on or reset
Start
Initialize
VT340 screen handler
even 1/2 second counter/timer interrupt handler
Start
Read inputs
Average analog inputs
Convert to engineering units
Overlimit and out-of-range check
Mode control
Heater control
Toggle watchdog output
Write outputs
Return
VT340 receive character interrupt handler
Start
Get keyboard or trackball button input
Store input in shared memory buffer
Return
odd 1/2 second counter/timer interrupt handler
Start
Read analog inputs
Return

Figure 2: High Level Software Flowcharts

Operator Interface

The operator controls and monitors the PTI entirely through graphical and text based screens. The initial screen displays a main menu consisting of labelled buttons that allow selection of other screens. The screens are selected by moving the trackball cursor to the desired button and clicking the left trackball button. The screens allow the user to monitor the PTI, change its operating mode, view and respond to alarms, tune the PID controllers, change setpoints, and perform troubleshooting and calibration activities. Figures 1 and 3 through 8 show some of the screens.

The screens update their number and text displays and graphical dynamics in real-time in response to changes in input data or changes due to operator interaction through the keyboard or trackball. Some common features of the screens include use of color to emphasize active parts of the system, use of inverse text to call attention to out-of-spec values, use of on-screen buttons, ability to get directly to selected screens without going through the main menu screen, operating mode information available on all screens that the operator will normally use, and password control on setpoint, tuning factor, and maintenance screens. The following paragraphs describe some of these features in more detail.

Color dynamics apply to graphical items, text displays, and numeric data displays. When an item, such as a heater or fan is energized, its symbol is displayed in red; when de-energized it is displayed in green or as an outline filled with the screen background color. The screen background color is black. The selection buttons are blue for the normal operator selectable buttons and gray for the calibration and digital output check buttons. When a button on the PTI Control screen is selected, the button border color changes so it appears depressed and the button color changes to yellow. On the PTI Process Summary screen, the alarm screen button changes from blue to red when an input goes into an alarm or off-normal state. Clicking on the alarm screen button brings up the Input & Alarm Status screen where the alarming item will appear in inverse red. Clicking on the inverse red number or text acknowledges the alarm for that item and the color changes to green or non-inverse red.

The Plugging Plot screen shows a plot of plugging temperature versus delta flow. This screen updates in real-time as the plugging run progresses. The plot line is red so it stands out well against the black background. This screen can be exited and reentered at any time during, before, or after a plugging run. It either shows the plugging run that was last completed or the current, in progress, plugging run. When the screen is entered, it redraws the plot line from the beginning of the plugging run. The numeric displays to the right of the plot always display the x and y values of the last point plotted.

On the PTI Control, Setpoints, and PID Tune screens changing the control mode, entering a new setpoint, or entering a new tuning factor is a two step operation. After a new mode is selected or new value entered, a confirmation button appears at the bottom of the screen. This button must be clicked on in order for the new mode or value to take affect. Exiting the screen without clicking the confirmation button cancels the change.

The PTI Control and Input & Alarm Status screens provide additional information to the operator during mode selection and alarm display. When a mode is selected that is not allowed, such as selecting plugging run while in the off mode, the plugging run button will not appear depressed and a text string will be temporarily displayed in the upper right portion of the screen. This text gives the reason why the mode selection was not allowed. Certain input values can go into alarm for more than one reason, such as total flow too high, too low, or not equal the sum of the other two flowmeters. For these items additional text will appear with the data label or value display indicating the specific reason for the alarm. In the case of the total flow display; the words high, low, or unbalanced will appear in red to the left or above the "total flow" label.

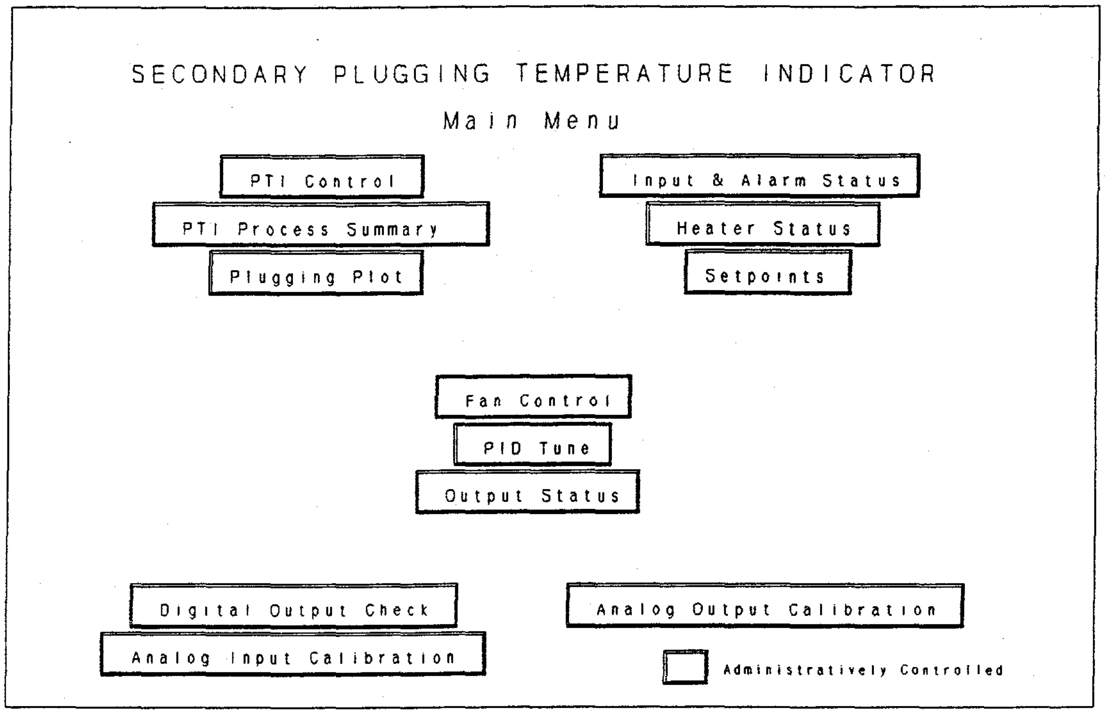
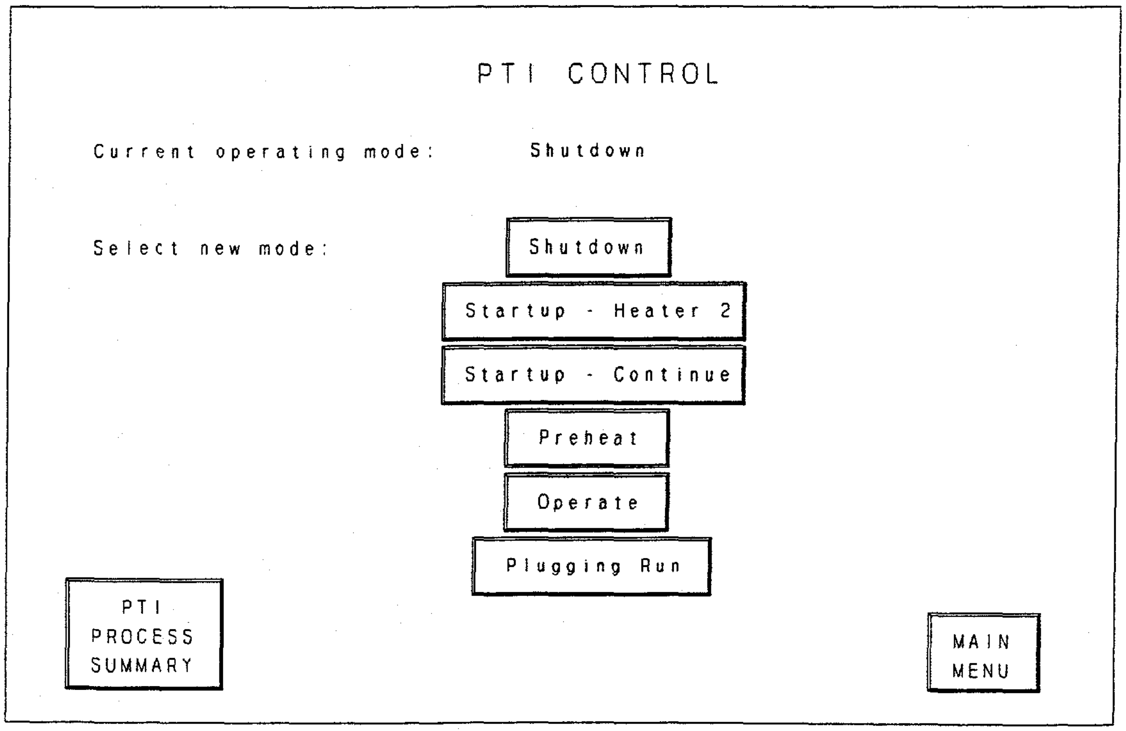
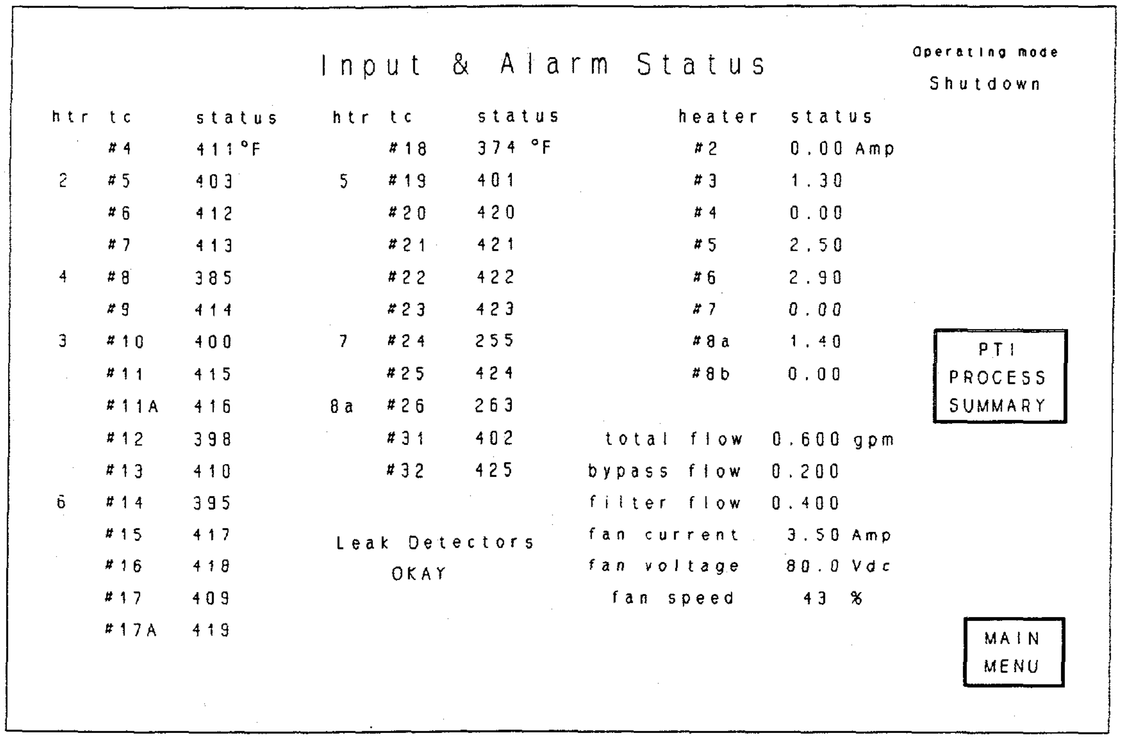


Figure 3: Main Menu



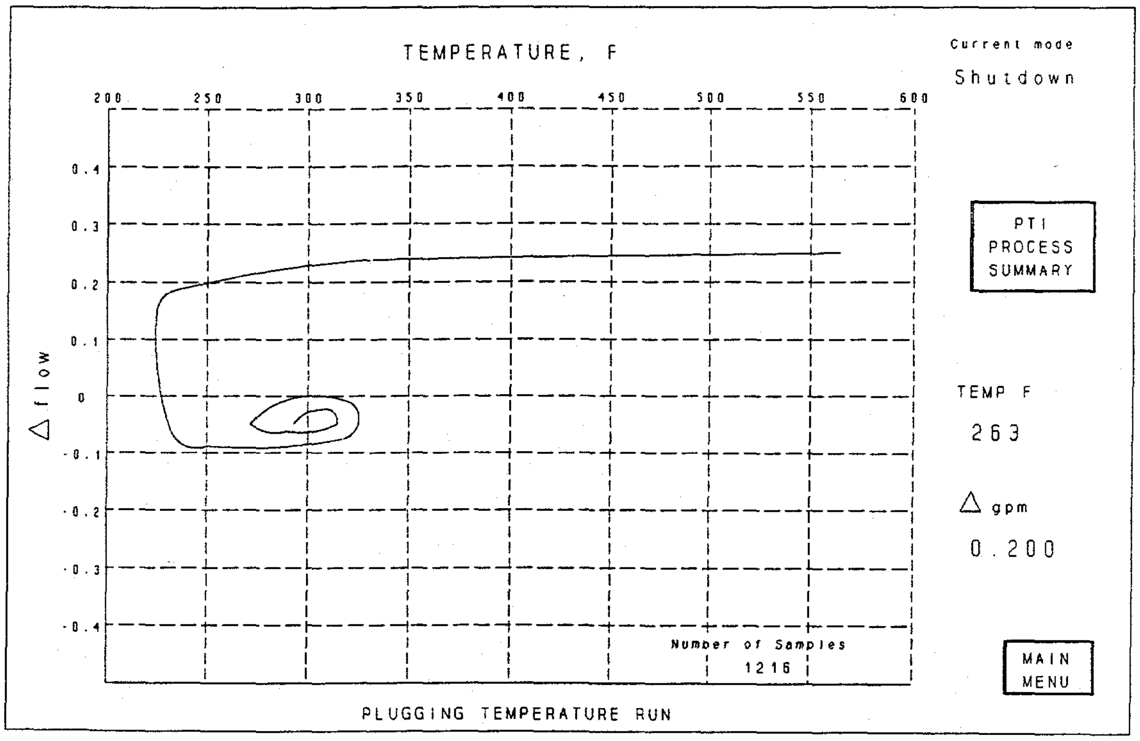
Current operating mode: Shutdown

Figure 4: PTI Control



```
                 Input & Alarm Status                          Operating mode
                                                                  Shutdown
htr  tc     status     htr  tc    status        heater  status
     #4     411°F           #18   374 °F          #2    0.00 Amp
2    #5     403        5    #19   401             #3    1.30
     #6     412             #20   420             #4    0.00
     #7     413             #21   421             #5    2.50
4    #8     385             #22   422             #6    2.90
     #9     414             #23   423             #7    0.00
3    #10    400        7    #24   255             #8a   1.40          PTI
     #11    415             #25   424             #8b   0.00        PROCESS
     #11A   416        8a   #26   263                               SUMMARY
     #12    398             #31   402       total flow   0.600 gpm
     #13    410             #32   425       bypass flow  0.200
6    #14    395                             filter flow  0.400
     #15    417        Leak Detectors       fan current   3.50 Amp
     #16    418            OKAY             fan voltage   80.0 Vdc
     #17    409                              fan speed     43  %
     #17A   419                                                       MAIN
                                                                      MENU
```

Figure 5: Input & Alarm Status

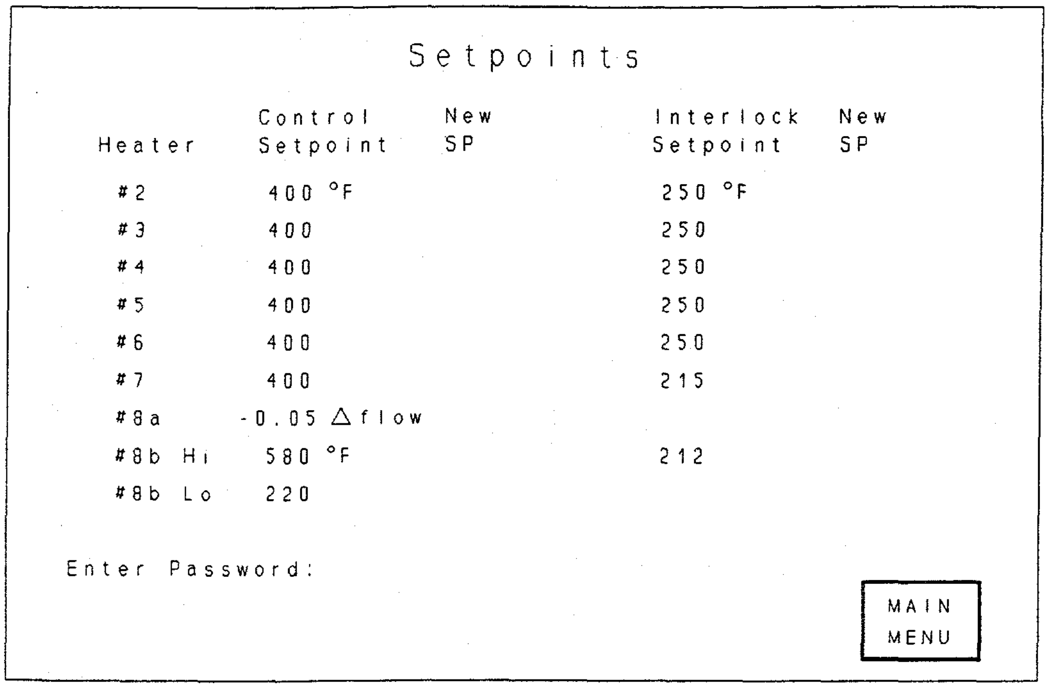


Number of Samples 1216

TEMP F 263

Δ gpm 0.200

Figure 6: Plugging Plot

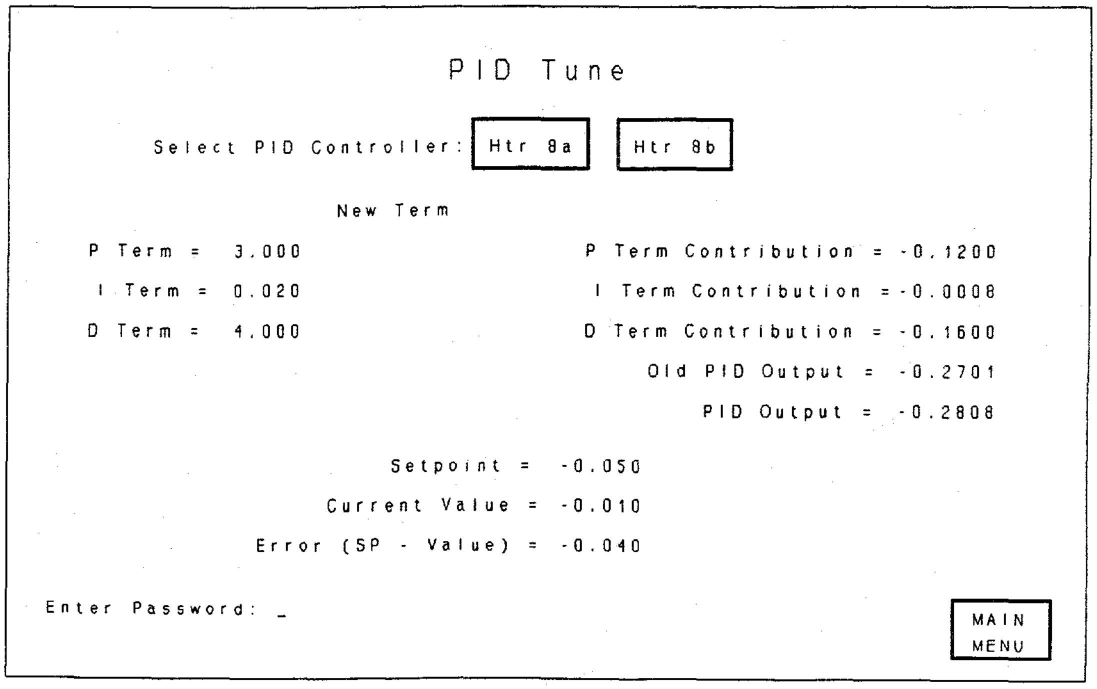


Setpoints

Heater	Control Setpoint	New SP	Interlock Setpoint	New SP
#2	400 °F		250 °F	
#3	400		250	
#4	400		250	
#5	400		250	
#6	400		250	
#7	400		215	
#8a	-0.05 △flow			
#8b Hi	580 °F		212	
#8b Lo	220			

Enter Password:

Figure 7: Setpoints



PID Tune

```
Select PID Controller:  Htr 8a   Htr 8b

                  New Term
P Term =  3.000                     P Term Contribution = -0.1200
I Term =  0.020                     I Term Contribution = -0.0008
D Term =  4.000                     D Term Contribution = -0.1600
                                         Old PID Output = -0.2701
                                             PID Output = -0.2808

                   Setpoint =  -0.050
              Current Value =  -0.010
         Error (SP - Value) =  -0.040

Enter Password: _
```

Figure 8: PID Tune

DEVELOPMENT METHODS AND TOOLS

Workstation Development Platform

The software was initially developed, debugged, and tested on a real-time enhanced Unix workstation. This provided a more powerful and efficient environment than would be available if all the development was done with the target computer system. The software was developed using standard C library functions that are available on both the workstation C compiler and target system C++ compiler. Compiler directives were used to isolate any machine dependent areas of the code. The workstation environment allowed a PTI simulation task to be run with the control software as it was being developed and checked out. The PTI simulation program was run as a separate task on the workstation and interfaced with the control software task through shared memory and an intertask signalling mechanism.

The use of a real-time enhanced multi-tasking workstation operating system allowed the software to be developed and run in an environment very similar to that of the target system. A repetitive interval timer was set up that interrupted the screen control software and ran the real-time control code in the same manner as on the target system. The actual PTI graphics terminal was connected to the workstation through a serial port and interacted with the screen software in the same manner and at the same baud rate as on the target system.

PTI Simulation

The PTI simulation program was developed by an individual who was very familiar with the dynamics and physics of the PTI. This program was originally written in Basic for use on a personal computer. It was rewritten into C and expanded to provide a PTI simulation for development and checkout of the new control system software.

Use of the simulation task allowed the control and screen software to be thoroughly debugged and checked out prior to installation on the actual PTI system. The main limitation was the inability to run target system I/O board driver software and CPU board hardware setup and low level interrupt code. This code was written but bypassed via compiler directives when the software was run on the workstation. Appropriate functions were written to replace the drivers and setup routines to provide similar functionality.

A significant benefit of operation with the simulation was realistic operation of the control software and operator interface for demonstration, training, and operator feedback purposes. This was valuable in getting the control software and screen displays basically right before installation.

Target System Development Interface

The target system CPU board was purchased with a development PROM set and software utilities that allow target system executable code to be downloaded and run remotely from a computer with a remote debugger. This development tool is the Ziatech Inc. Virtual-Target-Interface (VTI). It is designed to allow code to be run in the target system real-time environment using the Borland Turbo Debugger in its remote debug mode. The VTI includes startup code, library modification routines, and relocation utilities that allow software compiled and assembled with Turbo C++ and Turbo Assembler to be linked, downloaded, and run on the target system. The target system does not need any sort of an operating system (such as DOS).

When the software is ready to be burned into PROMs the VTI includes the utilities and start up code to create the ROMable executable software in a form that can be used by a standard PROM programming device. For the PTI this was an Intel Hex file. This file was used to program two 128Kbyte flash memory IC's. Once the flash memory is ready the VTI PROM set is removed from the CPU board and replaced by the application flash memory PROMs.

PTI Software Checkout and Initial Operation

The PTI software was initially developed and checked out in the workstation environment with the simulation task. Next, the software was moved to a PC-AT where it was compiled, assembled, linked, and relocation information added for the target system VTI environment. Compiler directives were used to bypass workstation specific code and include the target system specific code. The software was then debugged and checked out on the target system via the remote debugger.

The new control system cabinet with signal conditioning modules and circuits, power supplies, and output interface devices was built in preparation for installation in the PTI. This cabinet was complete to the point that, when placed in position, only the power, PTI system sensor, and actuator wiring needed to be connected to terminal strips in the cabinet for the physical installation to be complete. The new control computer was installed in the cabinet and tested before the cabinet was installed. Test equipment and, in some cases, typical actuators were connected to the cabinet terminal strips on a point-by-point basis to verify proper cabinet wiring and software I/O and display operation. The software was run using the VTI environment during this phase.

After the new control system cabinet was installed in the PTI system, the control system was, once again, run using the VTI. The individual sensors were read in and each actuator activated to verify that the wiring from the PTI system to the cabinet terminal strips was correct. The system was then initially run in the VTI environment. After minor software changes were made and verified to be correct, the software was programmed into flash PROMs and the system was operated in its final configuration.

This software development and checkout method provided a hierarchical approach to a final computer control system. Each step completed, debugged, and verified correct operation of some level of the software. The workstation step completed the higher level control and screen software; the initial remote debugging step completed the target system hardware specific setup, driver, and interrupt handling software; the PTI cabinet remote debug step completed and verified software and hardware out to terminal strip interfaces; and, after installation in the system, the final step completed and verified hardware from the terminal strips out to the sensors and actuators. This approach localized checkout and debugging to appropriate levels of the hierarchy; items further up the hierarchy, having been checked out and verified, would not be the location of problems that appeared at a lower level. The result, after the system was installed, was that any sensor or actuator problems were known to be from the cabinet terminal strips out, not in the computer system software or cabinet hardware.

GOAL ASSESSMENT

The abstract stated that a goal for this project was to accomplish this upgrade inexpensively using small, reliable computer and display hardware with a minimum of purchased software. How well did this system meet this goal?

Expense

Typical plant control system upgrades at EBR-II that involve real-time control with color graphical CRT based interfaces are done with two computers; a real-time frontend computer and a Unix based workstation level computer with a graphical interface. The workstation graphics are usually developed using a purchased graphics development tool that also includes libraries and utilities for running the finished user interface application. The costs involve a PLC or STD Bus based frontend computer, frontend computer software development tools and compiler, a workstation, the workstation compiler, and the workstation graphics tool. These control systems are fairly expensive. As a result these types of systems are applied to the larger control system upgrades.

The PTI control system, on the other hand, involved only one computer and a relatively inexpensive color terminal. Target system code development involved the use of a readily available PC compatible computer and the Turbo C++ compiler, assembler, debugger, linker, and VTI development tools. The total compiler and development tools cost was less than the cost of a workstation single user C compiler. The ReGIS graphics language native to the VT340 terminal eliminated the need to buy any type of graphics development tool. Since the final control system was ROM based without an operating system, no operating system or licensing costs were involved. This proved to be an inexpensive upgrade solution in comparison to the typical EBR-II control system upgrade.

Small

The size of the STD Bus computer makes it relatively easy to use in upgrades. It fits in a 5 1/4 inch high 19 inch rack. The power supply is small and no cooling fans were required.

Reliability

ROM based STD Bus computers have been used extensively in the plant. They have proven to be very reliable. There are no disks or fans to wear out. The VT340 monitor has also proven to be a reliable part of the system, the only maintenance is an occasional cleaning of the trackball to keep it working smoothly.

The reliability of the system is enhanced through the use of the STD Bus watchdog board. This board times out if it does not receive pulses via a front edge connector from some other board in the system. If the pulses stop, the board shuts the STD Bus power off via a relay. The system is designed so that powering the bus down puts all I/O board outputs in a failsafe condition. The pulses are sent from a digital output board channel that is driven by software that is part of the real-time control portion of the computer system.

Performance

The 12 MHz V53 board with the math-coprocessor is a very fast board for this relatively small application. With the math-coprocessor there are no constraints against floating point use. All of the real-time control, including the I/O reads and writes, only takes about 17 milliseconds. This leaves over 98% of the CPU time available for handling the VT340 terminal. This results in a responsive and fast system where there are no noticeable delays on screen updates or operator interaction.

RECOMMENDATIONS AND SUGGESTIONS

The following paragraphs discuss some of the lessons learned from doing this project and provide some recommendations and suggestions on implementing control systems of this type.

Reentrancy

One shortcoming with compilers that are designed to run in PC's running DOS or Windows is that they are not designed to run in a preemptive multi-tasking environment. As a result some of the C and math library functions are not designed to be reentrant. This is an important consideration in using Turbo or Microsoft C++ for real-time systems. The software design has to take into account this reentrancy problem and disable interrupts in certain places and/or use hardware to avoid non-reentrant library functions.

The PTI control system is run in a preemptive multi-tasking fashion so both disabling interrupts and a math coprocessor are used to avoid reentrancy problems. Sometimes it is difficult to know which functions are non-reentrant. In the PTI, during system checkout, it was observed that about once every two or three days, when one of a particular set of screens was selected, the computer system would lock up as the new screen was being initially displayed. This was narrowed down to the C sprintf function. Careful use of the interrupt enable/disable calls eliminated this problem without affecting system performance.

Operating System

When doing control system upgrades utilizing real-time computers it is not always necessary to have an operating system. If the computer system has no disks, has no need for a windowing operating system or some other need that requires DOS, one should consider not using an operating system. This eliminates the unknowns associated with an operating system such as hidden bugs and real-time performance degradation. Without an operating system, every aspect of the software can be known and maintainable. This may be important in certain critical applications.

Development Language

A high level language should be used rather than coding solely in assembly language. Use a language that is common, known by a number of people in the organization, and that is available on many different types of computer systems. C/C++ works well since it is available for all machines, is portable, and contains low level calls that eliminate the need to use assembly language in many cases.

The application should be coded in a machine independent manner. Machine dependencies should be isolated to specific functions or controlled through compiler directives. The PTI control system software ran in both the workstation environment and the target system through the use of compiler directives. This eliminated the need to keep track of two sets of software. The same files that run on the workstation were recompiled and run on the target system.

Code Flexibility and Reuse

The application software should be modular and well documented. This results in a system that can be changed and adapted as modifications are made to the system being controlled and provides a source of software functions that can be used on other applications. If the process expands to the point where a more powerful computer is needed, the software can be moved to the new platform with modifications only required to the machine dependent portions of code. Depending on how modular and well documented the code is this may not be a difficult task. The PTI control system code utilizes a number of functions that were originally written for in-plant Z80 based real-time control systems.

Knowledge and Experience

If possible, the people that are knowledgeable and experienced in doing systems of this type should be utilized. This will keep the job short, manageable, and cost effective. If the people doing the job are inexperienced, realize that a significant portion of the time spent on the job is associated with learning. Having some experienced people on the job will shorten this learning cycle significantly. Subsequent jobs will take dramatically less time and more easily incorporate new technologies if these same people are reused on these jobs.

Human Factors

In the area of human factors, there are a number of recommendations and suggestions based on the experience of doing this upgrade. These items are listed below.

1. Listen to the people who will operate the system. Develop the screens from their point of view. The information must be meaningful and logical from an operations point of view, not a developers point of view. There may be certain naming conventions from prior system usage and certain ways of viewing the system that are important to maintain.

2. After the system is installed don't assume all the work is done. Listen to the operators over a period of time as they use the system. Keep a log book next to the graphics screen so they can write down suggestions or short comings. Appropriate suggestions should be implemented.

3. Watch the system as it is operated and correct any obvious deficiencies. Often the people using the system aren't aware of the capabilities of the computer control or display system and will assume that certain things can't be done or corrected. A case in point on the PTI control system was that the operators didn't realize the impact of having all information displayed on a graphics terminal. On the old equipment they were used to making certain manual valve adjustments while looking at some instrument readouts that were fairly easy to see from a distance. Now the screen showed the same information in a very small size that could not be seen from 10 feet away where the adjustments were made. This was an obvious deficiency that required the addition of a user selectable screen that displayed the information in very large alpha-numeric characters.

4. When a screen uses a mixture of upper and lower case letters for text, small lower case letters can be hard to read. Consider making the text all upper case in these situations. On the PTI it became obvious that the labelling for certain components had to be in upper case.

5. Functionality of on screen objects needs to be apparent from the way they look. Items that can be clicked on, such as buttons, should not appear the same as rectangular indicators. Don't make the indicators have shading and shape that looks identical to push buttons.

6. Make it obvious when an item is selected. If it is a pushbutton, show the button depressed or depressed and in a different color. If an item can't be selected, but looks like it should be able to be, display the reason why. In the PTI system, where mode changes can't be made out-of-sequence, the screen will display the reason why if an attempt is made to change to an improper mode. Other alternatives are to change the color of the button, the button text, or to not display the button.

7. During long computer operations, such as displaying an on screen graph, that take a period of time to complete, give some indication that the computer is still working. This will assure the

operators that the computer is still working and hasn't locked up for some reason.

8. If the computer can provide information that is useful to the operators, display it rather than have the operator try to deduce it or read it from a graph. For instance, in the PTI, the final plugging temperature and delta flow is displayed in larger text by the graph rather than forcing the operator to try to read the graph. Another example are items that can alarm for more than one reason, for each alarm display some additional text next to the alarming item to make it clear what the cause is.

9. When new values such as setpoints or operating mode changes are entered, provide a confirmation type function. On the PTI this was done with a unique looking accept button that appeared only after new mode change, setpoint, or tuning factor data was entered. This helps avoid the inadvertent entry of invalid or incorrect data.

10. Find out which screens are normally used during operation and make it easy to move among these screens without taking unnecessary trips into other screens.

11. Place common information, that the operator needs to know, on all the screens if the information is important to system operation. In the case of the PTI this is extra screen select buttons on the PTI summary screen and the current operating mode display at the upper right of many of the screens.

12. Restrict access to screens that are only used for maintenance or tuning. On the PTI this was done through an external key switch and through passwords. The main idea is not distrust of the operators but, rather, to keep the casual passerby from affecting the system.

13. Make screen operation obvious if at all possible. Provide graphical or textual aids that help the operator on those screens that are used very infrequently.

14. Be careful in color usage. Certain colors have certain meanings, such as red for alarm. Don't put colors together that clash or are hard to look at together. Don't use too many colors.

15. In columnar textual data display screens, group data into logical groupings. Use spacing, labelling, color or some other graphical means to group the data.

When a control system upgrade is completed it is important that the operations people like and are comfortable with it. If they like the system and see that you are willing to work with them to correct any deficiencies, they will be more willing to promote and accept upgrades on other systems.

CONCLUSION

The PTI control system upgrade was successful in meeting its goal of providing a real-time computer based control system with a color graphical user interface in a small, reliable, and cost effective manner. It would have been difficult to justify the upgrade on a plant system this small if the approach described in this paper had not been taken. Utilizing people that have experience in applying real-time systems; using a hierarchical development approach; paying attention to human factors; and supporting the system after it has been upgraded will result in timely and cost effective upgrades that operations people are satisfied with. This upgrade approach is recommended for small plant systems that require cost effective real-time computer control with a color graphical user interface.