Lower Costs and Risks by Emulating a DCS on a PC

S.S. GODBOLE and G.F. MALAN, Framatome Technologies, Lynchburg, Va.

With today's PCs, it is practical and economical to emulate a distributed control system before making major capital and personnel resource commitments.

With the astronomically enhanced capability of personal computers and related software, and a reduction in their cost, some companies have begun to emulate distributed control systems (DCSs) using PCs. An emulation reproduces both the "look and feel" and the essential operating characteristics of a DCS.

A DCS application (input/output, control logic, and operator interface) can be configured on a PC-based workstation, translated into PC software, and emulated by a PC or network of PCs. Much of the process is automated, thus permitting an economical and quick development of the emulation, which can be tested and debugged until the resulting system meets all necessary requirements.

Training personnel to operate the emulated DCS minimizes the risks to plant/process equipment and personnel inherent in operating an existing or new plant with a new control system. To permit comprehensive testing of the DCS and realistic training, the emulations can be integrated with plant/process dynamic real-time simulation models that respond to simulated start-up, load changes, and upsets in the same time frame as the real plant.

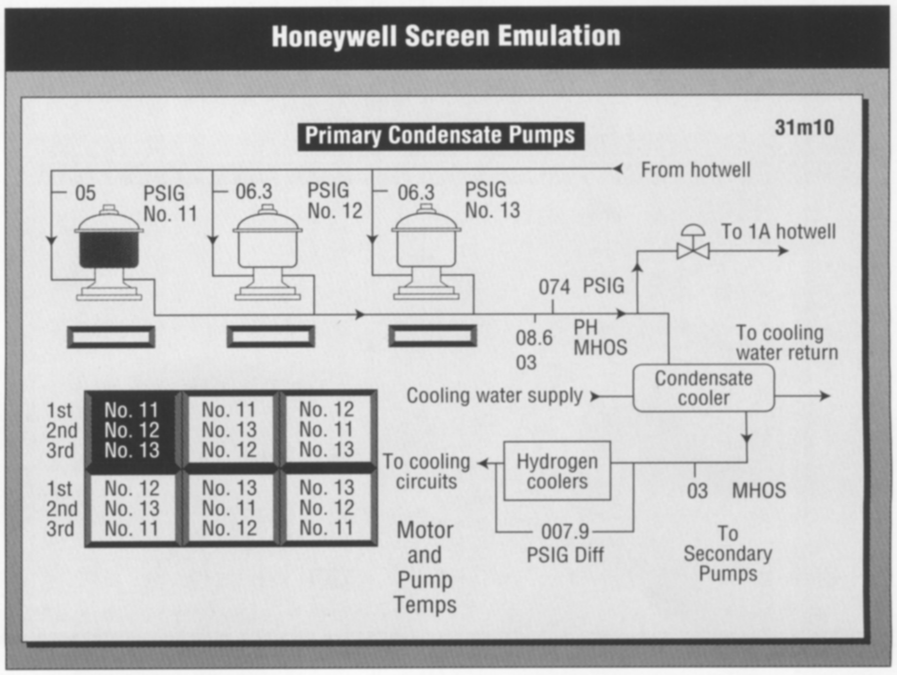
Emulation advantages

Typically, project personnel are not knowledgeable about the specifics of a DCS at the time the procurement decision is made, hence the decision is based on the vendor's presentation and demonstration. As the engineering details are worked out, however, some of the limitations of the DCS become clear. Sometimes this may not happen until the initial start-up of the DCS in the field when it is too late to do much about the limitations, and the customer has to live with them.

The most important advantage of using emulation is having the opportunity to provide, at the design level, an evaluation of the DCS system in an application-specific manner before making a long-term commitment. As a result, the following questions are addressed:

- Can DCS logic be graphically and easily configured?
- Are the control function blocks (PID) of the right complexity?
- Is the documentation of the system clear, accurate, and user-friendly, and does it give enough insight into the working of the control function blocks to permit proper application by the user?
- Does the DCS support one or more common high-level languages (Basic, Fortran, C)?
- Does the DCS microprocessor-based basic building block (BBB) for implementing control algorithms (control processor, multifunction controller) have enough capacity and speed for the application? Is it based on a 286, 386, 486, Pentium, or functionally equivalent system?
- If two or more BBBs are required, will the information flow between them adversely affect performance?
- Will the system be user-friendly in operation and maintenance?
- How user-friendly is the operator interface facility in the DCS? Does it have graphic building capability, such as selecting icons from an on-screen library? How many colors are there? Will it export and import an ASCII file?

It is possible to develop a dynamic



1st	No. 11	No. 11	No. 12
2nd	No. 12	No. 13	No. 11
3rd	No. 13	No. 12	No. 13
1st	No. 12	No. 13	No. 13
2nd	No. 13	No. 11	No. 12
3rd	No. 11	No. 12	No. 11

Fig. 1: This PC emulation of a typical Honeywell TDC3000 screen was produced by translating the corresponding Honeywell DS graphic file. The emulation is performed by a Visual Basic program running under Windows NT.

simulation model of the process or plant at a small additional cost using modeling systems. If the DCS emulation is tested and debugged in conjunction with the dynamic model, the resulting DCS logic, when installed in the field, will have a high probability of working right the first time, thus ensuring the success of the DCS project while reducing the overall time and cost.

For plants already operating with a DCS, it is possible that the particular DCS hardware is no longer available for use in a hardware-in-the-loop application. In such cases, DCS emulation may be the only practical alternative.

DCS architecture and functions

The DCS control logic is typically arranged in two ways—by control functions and by plant systems. Some DCS vendors assign a BBB to a specific plant control function, while others assign a BBB to control function blocks scattered over several plant subsystems.

In the first case, for example, the BBBs may be assigned to feedwater control, rotor-stress monitoring, etc., with one-to-one correspondence between the control function, control logic drawings, and the BBB. For some DCSs, the control logic is configured graphically using a PC workstation by making a drawing and creating the database that is downloaded into the BBB. For others, the configuration is done the old-fashioned way by creating and editing an ASCII file containing the specifications of the control function blocks. The control loop is configured by including the required transducer/transmitter signals, control function blocks, and final control element demand signals.

In the second case, assigning a BBB to function blocks across several subsystems, the BBB may be configured by creating and editing an ASCII file containing specifications of various entities (i.e., control function blocks) arranged in a prescribed order. These entities belong to several plant subsystems—such as combustion turbine generator/heat recovery steam generator (CTG/HRSG), plant electrical systems, motor-operated plant equipment, and cooling towers—and appear in control logic drawings arranged by plant systems. Thus, during debugging, it is not obvious in which BBB a certain entity resides. This information is necessary to make a change either to one entity or to several similar entities included in different BBBs.

Features of a DCS, such as reliability, security, and interaction between DCS modules over LANs can be verified using a scaled-down DCS in conjunction with dynamic simulation.

To overcome this difficulty, a graphic database can be created from the DCS configuration files to provide control logic drawings arranged by control functions. These drawings supplement similar drawings arranged by the plant/process systems. The database is also useful for on-line queries about a specific system entity, its connections with other entities, or changes to an entity. The query or change can be initiated by clicking a particular entity on the drawing and having a dialog box containing the attributes of the entity appear. The database can also revise the drawings arranged by the plant/ process systems as the contents of the configuration files evolve.

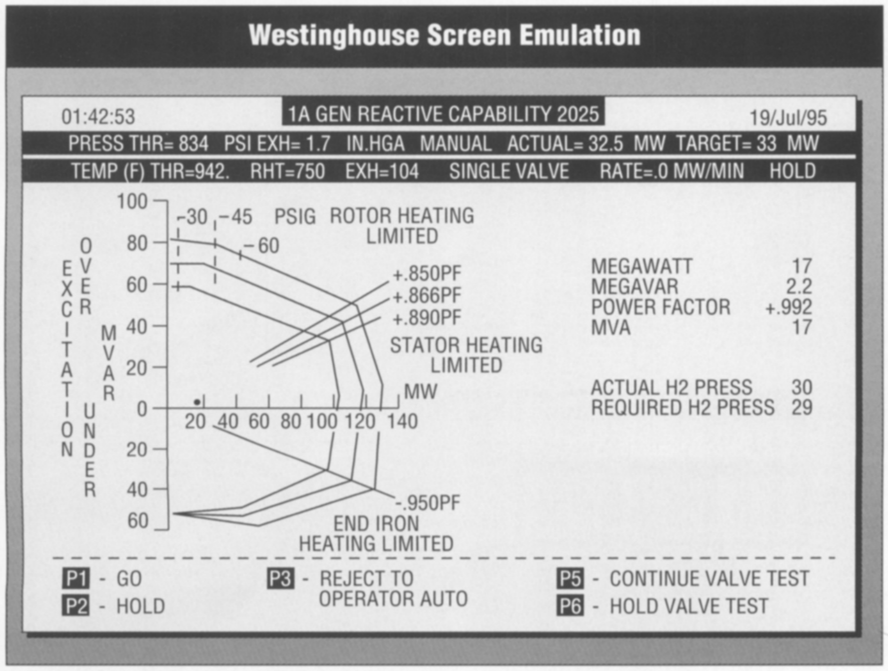
The operator interface is configured by creating graphics of various parts of the plant/process subsystems and associating the graphic elements with input/output signals from the plant or internal values generated in the DCS.

Features of a DCS, such as reliability, security, and interaction between DCS modules over LANs can be verified using a scaled-down DCS in conjunction with dynamic simulation. This hardware-in-the-loop approach is called "stimulation," and usually follows emulation.

Translating to the PC

It is possible to translate the various reports available in a DCS to produce code that can be executed in the PC. Typically the control logic configuration in each BBB is translated into a code block in a high-level language. Similarly, each operator interface graphic can be translated into an identical graphic that can be implemented on one or more PCs and screens (see Figures 1 and 2).

To implement the DCS emulation, translated control blocks are compiled and executed in the PCs, as are the translated pictures. The control logic and operator interface (OI) communicate with each other over an Ethernet network using TCP/IP protocol. If a dynamic model is used to test and debug the DCS control logic and OI, the correspondence between model variable names and DCS variable names is established using database techniques.


```
Westinghouse Screen Emulation

01:42:53        1A GEN REACTIVE CAPABILITY 2025        19/Jul/95
PRESS THR= 834  PSI EXH= 1.7  IN.HGA  MANUAL  ACTUAL= 32.5 MW  TARGET= 33 MW
TEMP (F) THR=942.  RHT=750  EXH=104  SINGLE VALVE  RATE=.0 MW/MIN  HOLD

MEGAWATT          17
MEGAVAR          2.2
POWER FACTOR   +.992
MVA               17

ACTUAL H2 PRESS    30
REQUIRED H2 PRESS  29

P1 - GO            P3 - REJECT TO           P5 - CONTINUE VALVE TEST
P2 - HOLD               OPERATOR AUTO       P6 - HOLD VALVE TEST
```

Fig. 2: PC emulation of a typical Westinghouse WDPF Classic screen. It was produced by translating the corresponding Westinghouse GCC graphic file, and performed by a Visual Basic program, running under Windows NT.

Debugging prior to use

To check out DCS logic and OI prior to initial system start-up in the field, simple test signals are typically injected sequentially at various input points, while observing the expected values at output points. While this is an important part of the testing, much more can (and should be) done prior to initial system start-up in the field to minimize functional bugs in the control logic and OI. Progressive DCS vendors and customers are receptive to the idea of testing and debugging in conjunction with a dynamic plant/process model—initially using an emulated DCS and later using the actual DCS hardware (see Figures 3 and 4). A by-product of this approach is the training of instrumentation and control personnel in a plant-specific environment.

Step 1—Emulation

The most promising distributed control system should be tentatively selected in a DCS project based on a nonhands-on, non-plant-specific evaluation. The control logic and operating procedure will already be defined for an existing plant. For a new plant, procedures can be worked out with the help of the architect/engineer, parallel to the DCS selection. When the DCS's system documentation and engineering workstation are acquired, the DCS vendor can help configure both the control logic and OI.

At this point, the control logic and graphic translators for the chosen DCS can be used to develop the emulation. The plant/process dynamic simulation model should be developed in parallel and kept ready to integrate with the DCS emulation. Plant simulation should then be "hooked up" with the DCS emulation for testing and debugging the DCS configuration.

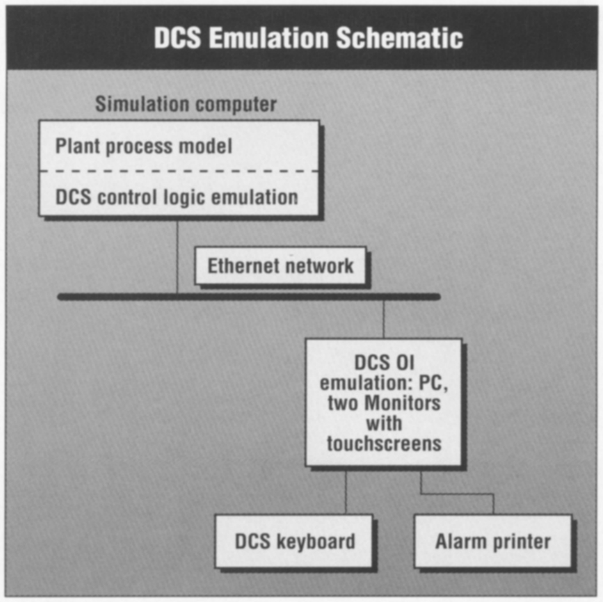


Fig. 3: DCS control logic emulation is integrated with the plant process model on a PC simulator server. The operator interface emulation runs on separate PCs that communicate with the simulator server over a local area network.

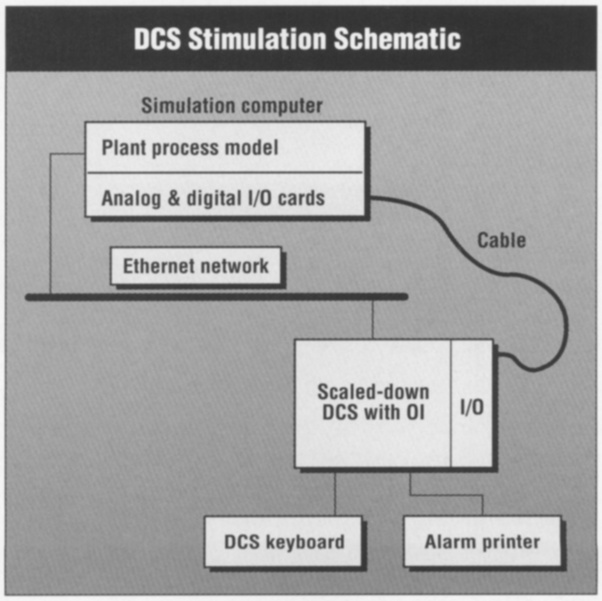


Fig. 4: The actual scaled-down DCS hardware interacts with the plant process model with real-world stimuli.

Step 2—Stimulation

After the emulation is debugged, the control logic and OI can be installed in a scaled-down version of the DCS. The DCS can then be connected with plant simulation, using analog and digital I/O cards in the PC to stimulate the plant model. The DCS can be operated as it would be in the plant, and evaluated to verify performance per the specifications. If the DCS performs satisfactorily up to this point, the full-scope DCS can be procured with a high degree of confidence.

Project examples

In a DCS project targeted to replace an existing analog control system, equipment was selected without a firsthand, application-specific evaluation, and the DCS logic was tested directly in the stimulation mode. A large amount of capital and labor resources were committed over a long schedule. Several shortcomings of the system were discovered relative to the application and the DCS never made it to any plant site.

In a repowering project, the DCS (consisting of equipment from two DCS vendors) was selected and preliminary control logic and OI were developed. As a part of developing a compact simulator for operator training, the DCS was emulated prior to initial start-up in the field. Several bugs in the configuration and OI were identified while testing the emulated system. Plant operating procedures were also verified and refined using the emulation. In fact, the translators used in the simulator development are robust enough to efficiently generate the control logic and OI graphics for other projects under the latest Windows NT operating system. Once tested, the control logic and OI graphics can be converted into the format compatible with the DCS.

Improved performance

Revolutionizing the way plant and process controls are being developed and operated are the rapid advances being made in personal computers, related technologies, and software tools. The emulation/stimulation approach represents a systematic way to evaluate a particular DCS in an application-specific environment before making a long-term commitment. The result is a significantly improved overall performance of retrofit or new control applications in terms of cost, schedule, risk, and overall satisfaction. □

Other major contributors to this article include W.T. Sneed, C.S. McKim, R.B. Brownell, and F.E. Harler. For more information, contact Framatome Technologies at Tel: 804/832-2696; Fax: 804/832-3663; E-Mail: simulator@framatech.com