

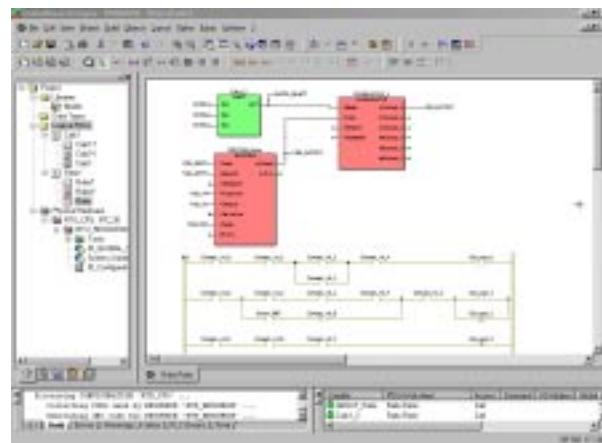
# Control Wave™ DESIGNER

**ControlWave Designer** is a fully IEC 61131-3 compliant programming environment for configuration of continuous and discrete control applications. The advantages offered by IEC 61131-3 has made it the emerging international standard programming environment and is currently employed by most all PLC manufacturers. As a result of its' standardization and widespread acceptance, it allows controls engineers the flexibility to choose the most appropriate controller hardware system without the necessity of learning a new programming language for each platform. The flexibility of IEC 61131-3 further allows each manufacturer to augment the basic set of functions with product specific higher level functions to take maximum advantage of each products distinguishing capabilities, yet strictly adhere to the IEC 61131-3 standards.

## Function Block Libraries speed application development

**ControlWave Designer** includes an extensive library of function blocks designed to take full advantage of the significant features offered by **ControlWave**. This library includes:

- Averager
- Comparator
- Integrator
- Flip-Flop
- On/Off DelayTimer
- High Speed Counter
- Multiplexor
- Demultiplexor
- Total/Trend
- Lead/Lag
- Stepper
- Archive Store
- Sequencer
- AGA8 - Gross
- Audit
- Function
- Accumulator
- Differentiator
- Vlimiter
- Pulse Duration Output
- List
- Hi Select
- HI/LO Limiter
- PID
- Encode
- Command
- AGA3



**ControlWave Designer** is based on a modern Windows 32 bit technology, offering standard functionality of zooming, scrolling, customizable toolbars, drag & drop operations, shortcut manager, and dockable windows. The Designer is used to create, edit, compile, debug, document and print, simple as well as very complex process control applications.

## Alarm System

One of the most important functions of any process automation system is to detect and report alarm conditions. **ControlWave Designer** provides the capability of adding any analog or logical variable to an alarm function block to detect, time stamp, store and report analog limit, discrete status, and change of state alarms.

Unlike most other systems, **ControlWave** detects, time stamps and stores alarms locally in the controller rather than at the PC level, ensuring that alarms are detected even in the event of a communication failure. Alarms are then reported to the PC once the communication link is re-established. The Alarm Record Queue size is configurable. The default is one hundred alarms. Once full, the oldest alarms will be

**Bristol Babcock**



maintained with time stamp. New alarms will be logged but without a time stamp until some of the older alarm space is freed through reporting. Alarms are time stamped to 1 ms for internal action and reported to the PC with a resolution of 20 ms.

*The following alarm blocks and types are supported:*

- **Analog Alarm Function Block**  
Detects High, High-High, Low and Low-Low alarms  
Independent deadbands are provided for high and low alarms  
Alarm reports may be enabled or disabled to suppress nuisance alarms  
Alarm Priority – Event, Operator guide, Non-critical or Critical  
Alarm descriptor text string  
Reports single, momentary and multiple alarm conditions since last report
- **Discrete Alarm Function Block**  
Detects selectable On or Off status  
Alarm reports may be enabled or disabled to suppress nuisance alarms  
Alarm Priority – Event, Operator guide, Non-critical or Critical
- **Change of State Alarm Function Block**  
Detects discrete change of state from On to Off or from Off to On  
Alarm reports may be enabled or disabled to suppress nuisance alarms  
Alarm Priority – Event, Operator guide, Non-critical or Critical

Alarms may be transmitted to up to four network connected PCs. Other PCs may obtain the alarm reports from the four declared PCs. At any time the operator may request all current unacknowledged alarms be re-reported. This is particularly useful for PC HMI software products that do not store the system alarm history. In this case the alarm history can be obtained from **ControlWave** at the operators discretion.

## Historical System

Again, most systems rely on the PC to collect and store historical data. But what happens when the data can't be collected due to a communication line failure. **ControlWave** ensures that important audit and archive data is securely stored, with time stamp, in the controller. To accomplish this, the Designer allows configuration of two special historical function blocks.

The historical data may be logged to either RAM or non-volatile Flash memory. When logged to Flash the data is maintained even after a cold start or application downloading.

The Audit System is responsible for detecting and logging of selected events and alarms. The Audit system is very application specific. It is often used in applications requiring quality tracking and reporting. Certain alarms, perhaps a high alarm on a pressure measurement, can be recorded in the Audit. Significant events may also be recorded. For example, a maintenance technician might replace a valve or a pump with different operating characteristics.

- **Audit Function Block**  
The Audit function blocks allows separate logs for alarms and events. For all alarm signals in the controller, the Audit Function Block generates a one-line message when the alarm signal goes into its alarm state, or when it returns to normal after having been in alarm. An event message is generated for signals that are included in a special signal list. Time stamp resolution for alarm and event message in the Audit storage is 1 ms. Reported alarm resolution is 20 ms. Each log can be up to 64 Kb
- **Archive Function Block**  
Historical archive files are similar to data arrays, except that each column is directly associated with a particular **ControlWave** variable, and each column has a descriptive title. The first column of

**Bristol Babcock**

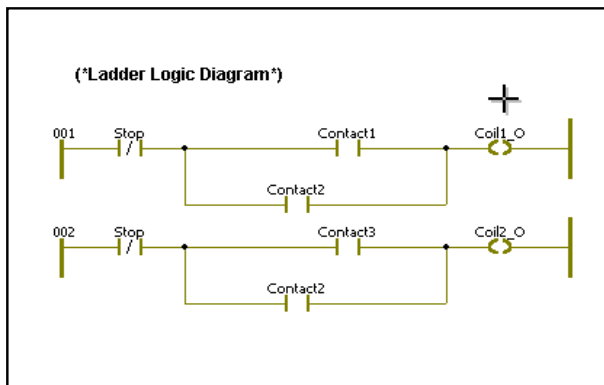
each row contains a time stamp, and the remaining columns in a given row are the variable values (or results of calculations based on variable values collected at the time specified by the time stamp. The data occupying a particular row is referred to as **record**; a record index is kept which points to the row containing the current record. Data storage interval may be 1 minute, 5 minutes, 15 minutes, hourly, or daily. Each file can be up to 64 Kb.

OpenBSI Utilities provides facilities to collect this Audit and Archived data, on a scheduled or demand basis, and present them in useful formats including .CSV and ODBC Access formats.

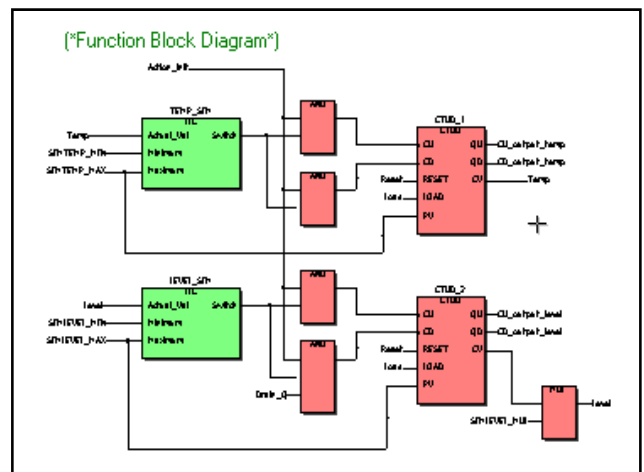
### Language Support - Why settle for only one

While many IEC 61131-3 programming tools offer only Ladder Diagram or a limited set of languages, **ControlWave** supports all five languages available in the IEC 61131-3 standard. The first three are graphical programming languages; Ladder Diagram, Sequential Function Chart, and Function Block Diagram. The latter two, Structures Text and Instruction List are text-based languages. Any or all five languages may be used to implement a process control scheme.

**Ladder Diagram (LD)** employs the elements common to traditional PLCs' such as normally open & normally closed contacts and coils. Ladder is typically used for sequential logic, interlocks and on/off control applications.



**Function Block Diagram (FBD)** is a graphical programming language that resembles the P&I drawings or circuit diagrams commonly found in process control applications. The graphical representation makes analog control loops visually easy to understand and auto-documents the control scheme.



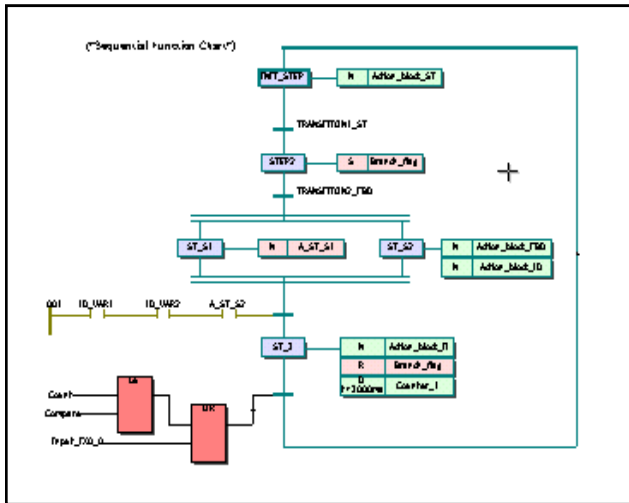
**Structured Text (ST)** is a high level text based language containing all the elements of a modern programming language. As well as providing a convenient method of including the library of function blocks and creating custom function blocks, it allows IF-THEN-ELSE and other conditional branching statements.

```

Heater* Generate Text for Intake *)
2  IF Intake THEN
3     TEXT:=CONCAT(TEXT_1, ' Intake: OPEN ');
4  ELSE
5     TEXT:=CONCAT(TEXT_1, ' Intake: CLOSE ');
6  END_IF;
7  R_TRIG_1(CLK:=( * BOOL * ));
8  ( * BOOL * ):=R_TRIG_1.Q;
9
10 (* Generate Text for Heater *)
11 IF Heater THEN
12    TEXT:=CONCAT(TEXT, ' Heater: ON ');
13 ELSE
14    TEXT:=CONCAT(TEXT, ' Heater: OFF ');
15 END_IF;
16

```

**Sequential Function Chart (SFC)** allows sequential operations to be programmed in a graphical manner similar to a flow chart. The steps represent the actions, which can be performed in sequence or in parallel, and the transitions represent the conditions that must be completed in order to advance to the next step.



**Instruction List (IL)** is also a text-based language similar to assembly language and as such it is somewhat less popular than the other four languages.

```

1 LD    %IX0.2    (*direct variables*)
2 AND   %IX0.3
3 OR    Action_INIT
4 ST    IL_VAR
5
6 LD    Input_IX0_0
7 JMP   MANUAL
8
9 (* Timer FB TON *)
10 LD   Timer_start
11 ST   TON_IL.IN
12 LD   PT_TON_IL
13 ST   TON_IL.PT
14 CAL  TON_IL
15 LD   TON_IL.Q
16 ST   Action_INIT

```

**User Defined Function Blocks – Reusable applications**

With **ControlWave Designer**, once a program containing one or more Functions or Function blocks is created and tested, it can be saved to a User Defined Function Block Library. Then, this new Function Block can be reused any number of times and can even be transported to other projects. The User Defined Function Blocks have reassignable inputs and outputs so each instance of the function block can have different variables attached to the terminals. Each new project can draw on previous development to significantly reduce implementation time.

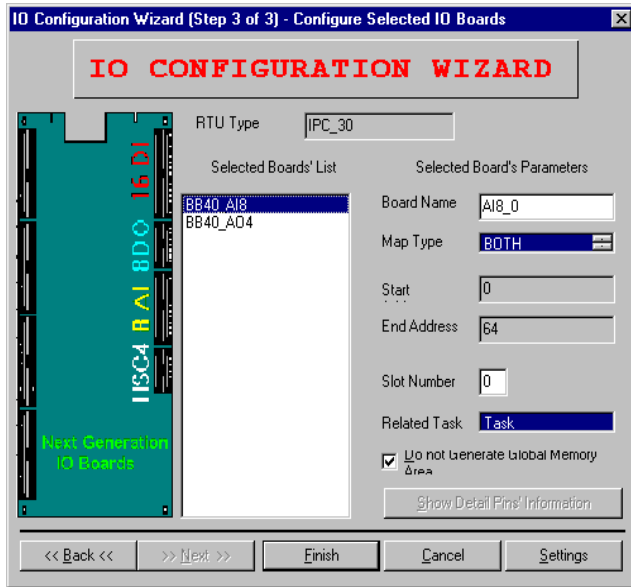
### Multitasking Operation

**ControlWave** runs in a Multitasking environment where program execution can be scheduled and prioritized. While **ControlWave** scan times are extremely fast, multitasking capability gives you control over critical task execution.

### Program Security

For program security the entire program, all associated variables, and graphics can be zipped into a single file. This project file can then be downloaded, stored in **ControlWave** memory, and later uploaded for recovery or modification by the Designer software.

The **ControlWave Designer I/O Configuration Wizard** simplifies the connection between the IEC 61131-3 application program and the physical **ControlWave** local I/O and Remote I/O modules. While I/O assignment can also be accomplished in the I/O\_Configuration section of IEC 61131-3, the **ControlWave Designer I/O Configuration Wizard** provides a self-prompting point and click menu system to simplify programming and eliminate syntax errors.



## On-line Program Debug Tools

No matter how experienced the engineer, there is always a need for powerful software debug tools. **ControlWave Designer** offers a comprehensive set of on-line tools to help you test, analyze and troubleshoot your application program.

The **Variables Cross Reference** list contains all variables, function blocks, actions, transitions, steps, jumps, labels and connectors which are used within the current project. It is a helpful tool for debugging and fault isolation.

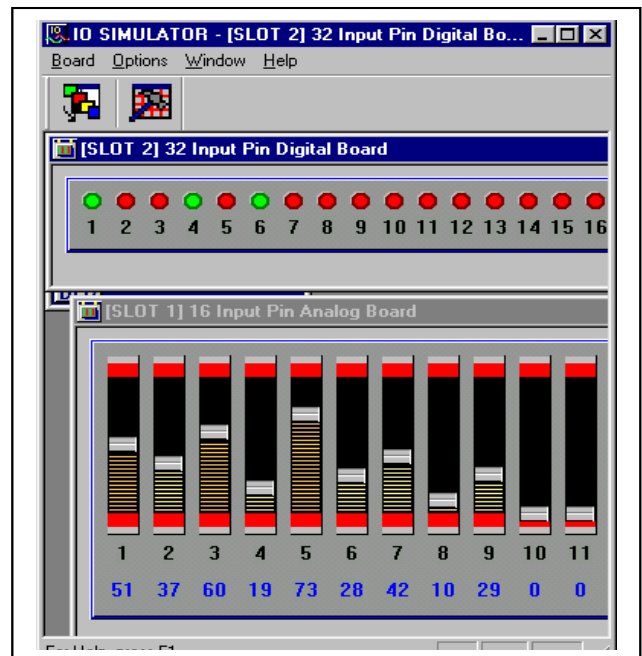
The **Logic Analyzer** is a powerful tool for recording values of variables over a certain time interval. They are displayed graphically in the Logic Analyzer window. All recorded values and settings of the Logic Analyzer are stored automatically with the project.

The **Watch Window** can be used to collect variables from different worksheets to get an appreciation about how these variables work together. In addition you can use the watch window to force and overwrite variables.

**Breakpoint, Single Step and Single Cycle** program execution provides a step and trace function that is extremely useful for program debugging. These functions allow you to continue the program execution line by line after a breakpoint has been reached.

For graphical worksheets, IL and ST worksheets, you can switch from the **Variable Status** to the address status with **Powerflow** and vice versa. Powerflow displays which program parts are actually executed and which ones not. This is useful for debugging worksheets with conditional jumps.

The **ControlWave Designer I/O Simulator** allows the Application load program to be tested on a PC, with simulated analog and digital inputs and outputs. The I/O Simulator utilizes the identical ProConOS real time system used in the **ControlWave PAC** unit; this allows initial I/O testing and debugging to be performed in a safe, isolated environment, without the need for a running **ControlWave PAC** unit and process I/O boards.





### OPC Server Database Open Connectivity

Variables in **ControlWave** Process Automation Controllers can be accessed using the Bristol Babcock OpenBSI OPC Server (Object Linking and Embedding for Process Control). All variables marked as "CSV" in the variable declaration page are included in a ASCII file generated by the OpenBSI Signal Extractor utility. This means that you only create one database, using **ControlWave Designer**, then automatically construct the OPC Server database from that source. Now any OPC compliant client HMI or SCADA application has access to **ControlWave** variables and alarms. By adhering to open standards, **ControlWave** simplifies this process and your life.

### Specifications

CPU: 133 MHz Pentium class PC – Minimum  
 RAM: 32 Mb  
 Disk space: 60 Mb  
 Comm RS 232 or 10/100 MHz Ethernet  
 Display: 800 X 600 – 256 color  
 Drives: CD-ROM  
 Operation system: Windows 95/98/NT

### Project Limits

Global variables	10000
Local Variables	15000
Resources in the project tree	100
Tasks per resource	15
Program instances per task	500
Embedded libraries	15
Programs per library	1000
Programs per project	2000
Code body worksheets per program	64
Different Function/Function Blocks per program	620
Same type Function/Function Blocks per program	1024
Steps/Transitions per program	110
Actions per program	165
Contacts/coils per program	3000
Nesting Functions/Function Blocks	128
Nesting levels for user-defined data types	8
Elements in a structure	200
Elements in an array	32766
FOR/CASE/IF nesting levels	200
Open On-line windows	30

**Bristol Babcock**