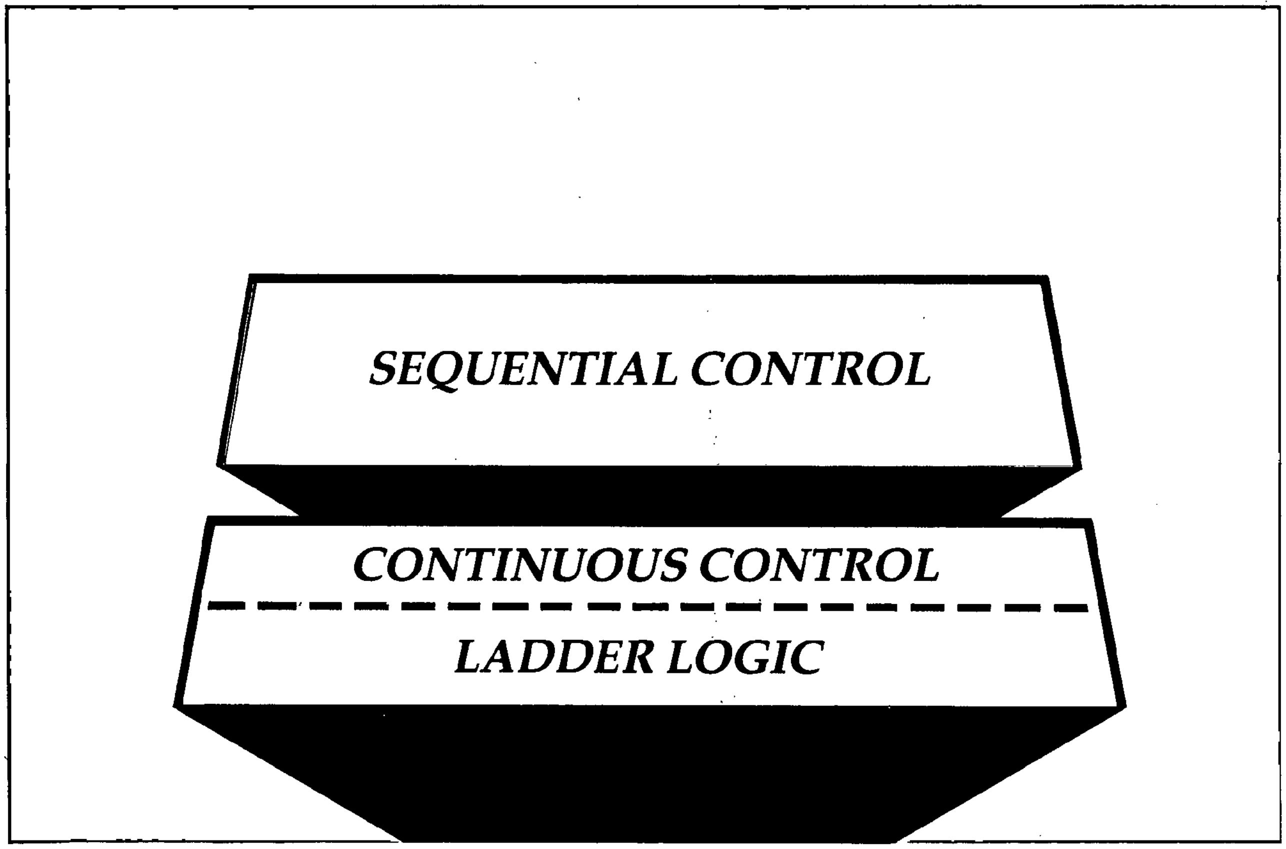
Product Specifications

PSS 21S-3K1 B3

# Intelligent Automation Series Sequential Control

Replace by 1287 Version of PSS 21S-3B1 B3



*Sequential control complements the continuous and ladder logic domains by providing sequencing capabilities. Sequential control can serve either as a stand-alone control strategy or with continuous, ladder logic, and discrete input/output control.*

## Sequential Control

Sequential control is one of the control domains of the Integrated Control Software. Sequence, continuous, and ladder logic blocks can be combined within a single compound structure. The sequence control software will run in any Intelligent Automation Series System station that contains control processor software.

Sequential control fulfills the needs of sequential, feedback-oriented applications at the equipment control level.

FOXBORO

## Concepts

In the continuous control domain, control blocks have fixed algorithms, a fixed number of parameters, and fixed properties. Additionally, continuous block algorithms can refer only to their own parameters.

In the sequential control domain, on the other hand, you can build your own algorithms using a logical set of parameters within the control compound/block structure. And Sequence block algorithms can read and write other compound/block parameters directly.

A high-level Sequence language and a parameter set provide the necessary tools to build Sequence blocks. When built, these blocks can be combined with continuous and/or ladder logic blocks in a compound (refer to Figure 1).

### Block Classes

There are three classes of blocks: Sequence, Monitor, and Timer.

| | |
|---|---|
| Sequence blocks | —Manipulate any compound, block parameter, or shared variable. (A shared variable acts as a linkage between an application and the control data base.) |
| | — Activate other Sequence blocks and Monitor blocks. |
| | — Send messages to historians. |
| Monitor blocks | —Monitor up to 16 process conditions (parameter values and Boolean expressions). |
| Timer blocks | —Keep track of time while control strategies are executing. |

### Block Types

Structurally, all sequential control blocks are the same. They differ only in their interaction within a compound. Their type defines this difference. The types are:

- Dependent Sequence block
- Independent Sequence block
- Exception Sequence block
- Monitor block
- Timer block

A Dependent Sequence block's execution is automatically delayed while any Exception Sequence block that is nested in the same compound is running.

An Exception Sequence block's execution, on the other hand, is never delayed.

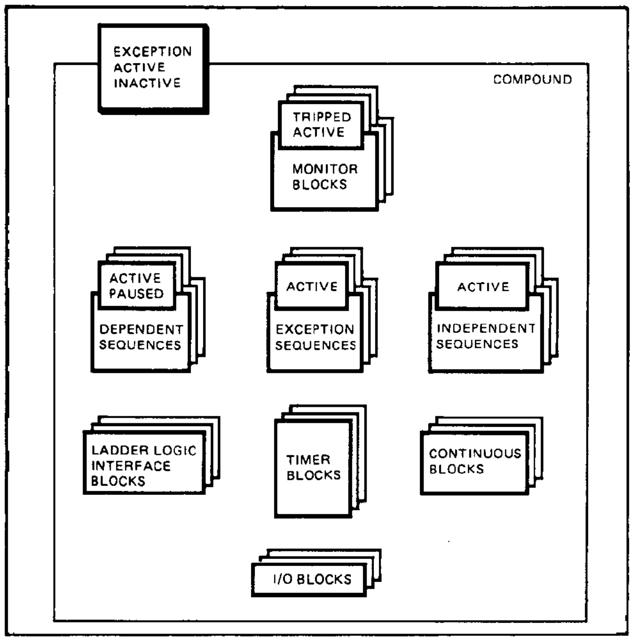


Figure 1.

The Independent Sequence block's execution does not affect the execution of other sequences nor does the execution of other blocks affect the operation of Independent Sequence blocks.

### Block States

The block states are Inactive, Active, or Paused.

The Inactive state means that a Sequence block is not executing statements or that the Monitor block is not evaluating conditions.

The Active state means that a Sequence block is executing statements or that the Monitor block is evaluating conditions.

The Paused state means that a Dependent block is in a suspended condition. Dependent blocks pause whenever an Exception block in the same compound becomes active. The Dependent block becomes active again when the Exception blocks complete their execution.

The Monitor block has a Tripped state (one of its conditions is true). Therefore, a sequence is activated by the Monitor block. All 16 conditions act independently.

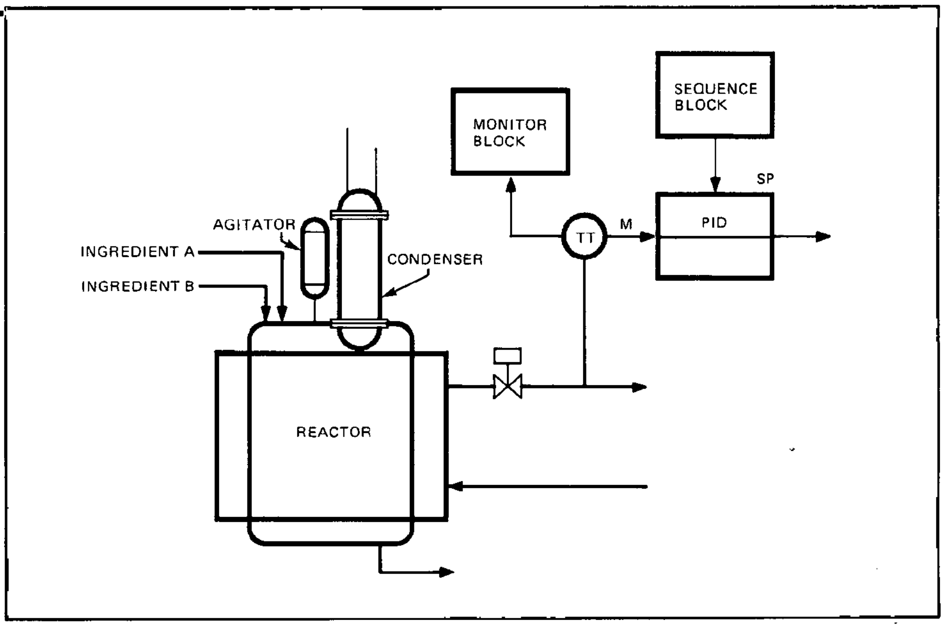


Figure 2. Sample Control Loops

The compound parameter SSTATE shows the operational behavior of the Sequence block states within that compound in one of three values:

*Inactive*—neither the Sequence blocks nor the Monitor blocks nested in the same compound are active.

*Active*—one or more Monitor blocks; and/or one or more Dependent Sequence blocks; and/or one or more Independent Sequence blocks that are nested in the compound are active.

*Exception*—one or more Exception Sequence blocks nested in the compound are active.

## Processing

Sequence blocks can run in parallel with continuous blocks, ladder logic blocks, and each other in that:

- Sequences may be Active concurrently
- Monitor blocks may be Active in parallel with Sequence blocks.

Timing is an independent feature and can run in parallel with other blocks.

Block processing order is:

1. Continuous and ladder logic blocks
2. Monitors and timers
3. Exception Sequence blocks
4. Dependent and Independent Sequence blocks.

## Sequential Control Domain

Sequence blocks contain logic that supervise the control loops. The logic regulates such things as:

- pressure control
- temperature control
- agitator control
- ingredient fills
- gas control, etc.

Figures 2 and 3 illustrate one example of how you might use Sequence blocks to supervise reactor control flow loops. The intent of this example is to show just a few control loops rather than a complete control strategy. Figure 2 shows a reactor having two ingredient inputs, an agitator, a condenser, and a heat jacket.

The Figure 3 flowchart shows Sequence blocks within a compound structure coordinating the continuous control loops in the Figure 2 example.

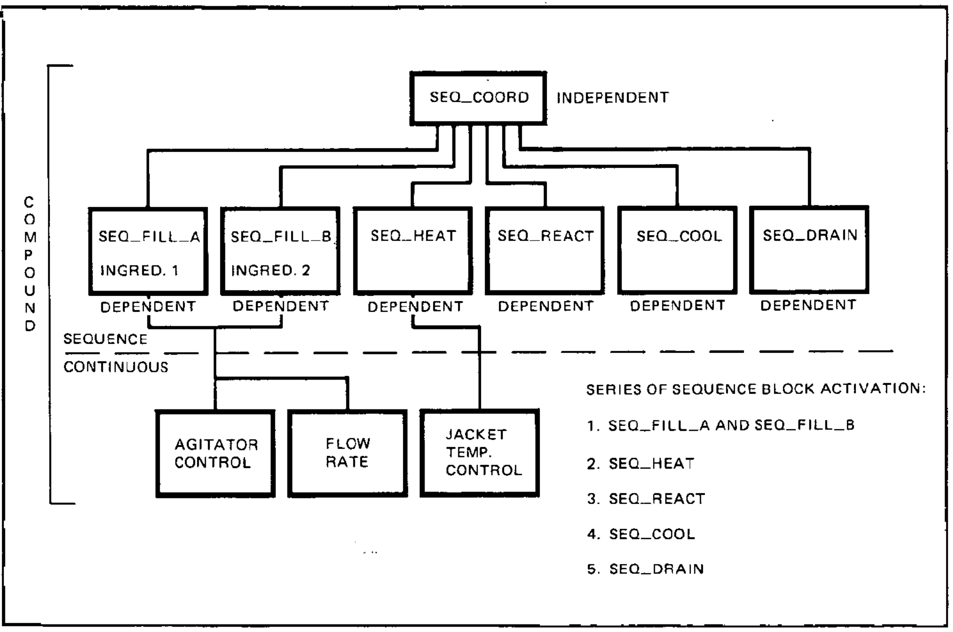


Figure 3. Mixed Compound Samples

In this example:

1. SEQ__COORD, an Independent block, is coordinating activities in the Dependent blocks. The first action it takes is to activate SEQ__FILL__A and SEQ__FILL__B.
2. SEQ__FIll__A and SEQ__FIL__B start to fill the reactor with two ingredients, concurrently.
3. The SEQ__FILL__blocks then send a set point to the PID block, start the jacket temperature control loop, and start the Monitor block(s) to watch the jacket temperature.
4. SEQ__FILL blocks continue adding ingredients to the Reactor. (They no longer need to worry about the jacket temperature alarms since a Monitor block is doing this.)
5. If the jacket temperature exceeds the alarm limits, a Monitor block activates an Exception block to correct the situation.

   When an Exception block is active within a compound, the Dependent blocks within the same compound pause (i.e., SEQ__FILL__A, etc.). However, Independent blocks continue executing.
6. When the SEQ__FILL blocks reach completion, they turn off the monitors associated only with SEQ__FILL.

## Sequence Language

The Sequence block language is a subset of the I/A Series Systems high-level sequential language. It is a structured language somewhat like the programming language, PASCAL. However, its focus is on control applications. The language includes logic flow control statements as well as Boolean and arithmetic functions. Refer to Figure 4 for a sample block built with the Sequence language.

The language statements do not operate the I/O directly. Rather, they make connections between their own parameters and I/O block parameters. They can write the I/O block parameters within continuous, ladder logic, or other Sequence blocks which operate the input/output.

*Logic Flow Control Statements*
These statements determine the flow of control. They may select groups of statements to be executed, skip them, execute them repetitively, or delay their execution. They are:

```
if. . .then. . .elseif. . .else. . .endif
for. . .to. . .do. . .endfor
repeat. . .until
while. . .do. . .endwhile
exitloop
goto
wait. . .time
wait. . .until condition
exit
```

*Data Operation Statements*
There are two kinds of statements that can manipulate data: the Assignment statement and the Procedural statement.

The Assignment statement replaces the current value of some object with a new value that results from evaluating an expression.

The Procedural statements are:

| | |
|---|---|
| ACTIVATE | — Activates a Sequence block or a Monitor. |
| ABORT | — Aborts an active Sequence block or Monitor. |
| START__TIMER | — Starts timers at current value or selected value. |
| STOP__TIMER | — Stops timers. |
| ACTCASES | — Manipulates activity of the 16 Monitor block cases. |
| SENDMSG | — Initiates a message from executing sequence logic. It can address any object that acts like a logical device, such as historians or annunciator keys. It can also assign a message to a string parameter. |

Figure 4. Sample Block Using the Sequence Language

```
EXCEPTION SEQUENCE
    {*****************************************}
    {                                         }
    {          EXCEPTION SEQUENCE             }
    {            CONTROL BLOCK                }
    {             FOR REACTOR                 }
    {                                         }
    {*****************************************}

    {*****************************************}
    { Specify the user-labeled parameters in }
    { one of the following formats:           }
    {    user_name : RInnnn; (real input)     }
    {    user_name : IInnnn; (integer input)  }
    {    user_name : BInnnn; (Boolean input)  }
    {    user_name : ROnnnn; (real output)    }
    {    user_name : IOnnnn; (integer output) }
    {    user_name : BOnnnn; (Boolean output) }
    {    user_name : SNnnnn; (string name)    }
    {*****************************************}

EXP2SS : SN1;

    {*****************************************}
    {  The declaration part finishes here.    }
    {  Now enter the block's statements.      }
    {*****************************************}

    (* EXCEPTION LOGIC FOR JACKET WATER PUMP FAILURE *)

    (* SHUT STEAM VALVES AOV_X15A and AOV_X15B *)

    REACT_CONT:AOV_X15A.CLOSE := TRUE;
    REACT_CONT:AOV_X15B.CLOSE := TRUE;

    (* PUT CONTROLLER TIC_X03 IN MANUAL *)

    REACT_ANLG:TIC_X03.MA := FALSE;

    (* SET OUTPUT OF TIC_X03 TO 0.0 *)

    REACT_ANLG:TIC_X03.OUT := 0.0;

    (* SEND ALARM TO ANNUNCIATOR VIA BOOLEAN BLK *)

    REACT_SEQ:EXCP2_MSG.IN_1 := TRUE;

    (* SEND MESSAGE TO THE OPERATOR *)

    SENDMSG ("WATER PUMP NOT RUNNING UNIT BEING SHUT DOWN") TO EXP2SS;

    (* PROCEED TO SHUT DOWN THE UNIT *)
       .
       .
       .
ENDSEQUENCE
```

N.B.: THE DELIMITERS, { . . . } AND (* . . . *) ENCLOSE COMMENTS AND OPERATOR REMARKS, RESPECTIVELY. COMMENTS ARE USED TO DOCUMENT THE PROGRAM. REMARKS EXPLAIN PROGRAM ACTIONS TO AN OPERATOR.

## Block Creation

Block creation is accomplished through the Control and I/O (CIO) Configurator, much like continuous block configuration. The Sequence Environment is accessed from a menu in the configurator.

The Sequence block ASCII source code is generated from a standard, full-screen Interactive Character Editor (ICE). This environment operates only on the selected block. Its function is to create/edit the source code. After this step, it is compiled and inserted into the control data base by the block configurator. When this process is complete, control is returned to the configurator for block functions.

*Block Functions*
The Sequence block functions include:

| | |
|---|---|
| FILE | — Provides a list of sequence files in sequence library. |
| COPY | — Copies an entire source file into a new source file. |
| EDIT | — Provides editing capabilities to the currently selected file. |
| VIEW | — Provides a search function. |
| COMPILE | — Compiles the currently selected destination file. |
| PRINT | — Prints existing source or list files using a print utility. |
| SAVE | — Saves data from the currently selected source file into a new file in the local sequence library. |
| EXIT | — Exits the Sequence Environment. |

FOXBORO®