

Appendix C

Using the Pre-Defined (Reserved) Ladder Language Variables

The Editor contains pre-defined variables that you can use in an individual ladder program to:

- execute logic based on a Processor scan
- specify how to handle error conditions
- check scan time execution

The pre-defined ladder language variables are local variables, which you can use for ladder instruction parameters. Their names are reserved and appear in the choices offered by the Variable Smart Matching option.

Using the Pre-Defined Program Scan Variables

Use the following Boolean variables to execute logic based on the Processor's scan. Only use these variables for input parameters (read-only):

• first_scan	Use this variable to execute logic during a program's first scan. This variable is true during the initial scan of the ladder rung and false during all other scans.
• second_scan	Use this variable to execute logic during a program's second scan. This variable is true during the second scan of the ladder rung and false during all others.
• last_scan	Use this variable to execute logic during a program's last scan. This variable is true on the final pass of the ladder rung after you have selected TASK STOP. This variable is not set when a STOP ALL error occurs. To use the last_scan variable, your program scan must be less than 0.5 seconds. IMPORTANT Do not use the last_scan variable in an event-driven program. Because these programs are based on the occurrence of an event, the last_scan variable may never be executed when used in an event-driven program.

Using the Pre-Defined Error Handling Variables

Use the following Boolean variables to help you handle error conditions. Use `error_eno` and `no_error_log` for output (read and write) parameters. You can use `task_error` as either an input (read-only) or output (read and write) parameter:

● <code>task_error</code>	This variable is set true whenever an error is found. Monitor the bit to see if an error occurs during execution and clear it by using the ladder logic. This bit is set true even if errors are not being logged.
● <code>error_eno</code>	Use this bit to determine the value ENO outputs will have if the instruction has an error. The default value is false, which disables any instructions that are connected to the ENO output, possibly making math expressions incomplete. When you set <code>error_eno</code> true, you can continue the execution of the logic connected to the ENO output even if the instruction block had an error. This variable can be changed during ladder logic program execution.
● <code>no_error_log</code>	Set this variable true to prevent errors from being entered into the program error log or being seen by the rung monitor. Only one error is logged for each instruction per program scan; however, you may want to prevent the errors encountered on certain instructions from being entered into the error log. The default state for the <code>no_error_log</code> is false. You can suppress error messages for a group of rungs by changing this variable during program execution. STOP ALL errors and parameter limit errors for AR1, AR2, and ARC instructions are still inserted into the error log when the <code>no_error_log</code> variable is true.

Using the Pre-Defined Ladder Execution Time Variables

Use the following double integer variables to help you check and monitor the program's execution time. Only use these variables as input parameters.

● <code>task_usec_max</code>	Use this variable to monitor the maximum execution time (in μs) for the current program.
● <code>task_usec_now</code>	Use this variable to monitor the latest execution time (in μs) of the current program. The execution time is the real "clock" time it took the program to run from start to finish. It includes the execution time for higher priority programs and interrupt service routines if they run while your program is running.

To reset these times, write a value of 0 into the variable.