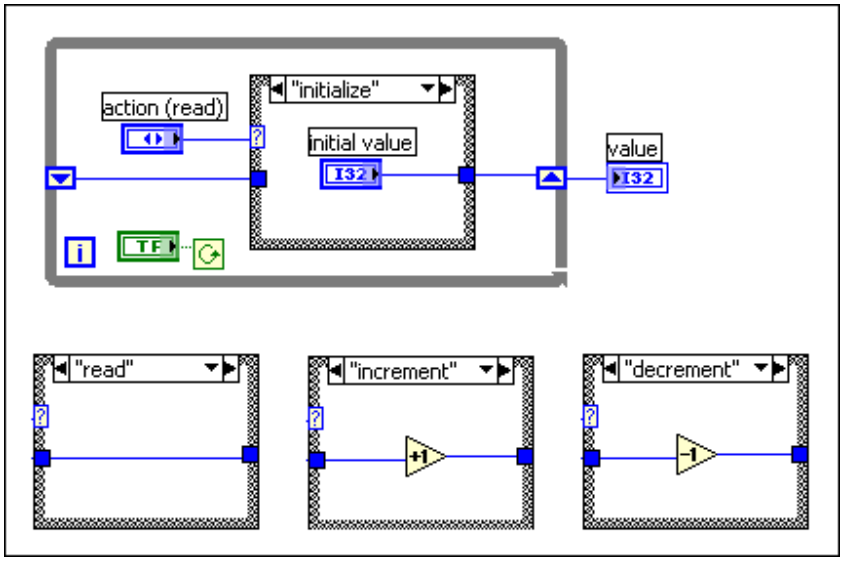
Functional Global Variables

Another way to avoid race conditions associated with global variables is to use **functional global** variables. Functional global variables are VIs that use loops with uninitialized shift registers to hold global data. A functional global variable usually has an **action** input parameter that specifies which task the VI performs. The VI uses an uninitialized shift register in a While Loop to hold the result of the operation. The following illustration shows a functional global variable that implements a simple count global variable. The actions in this example are **initialize**, **read**, **increment**, and **decrement**.



Every time you call the VI, the block diagram in the loop runs exactly once. Depending on the **action** parameter, the case inside the loop initializes, does not change, incrementally increases, or incrementally decreases the value of the shift register.

Although you can use functional global variables to implement simple global variables, as shown in the previous example, they are especially useful when implementing more complex data structures, such as a stack or a queue buffer. You also can use functional global variables to protect access to global resources, such as files, instruments, and data acquisition devices, that you cannot represent with a global variable.