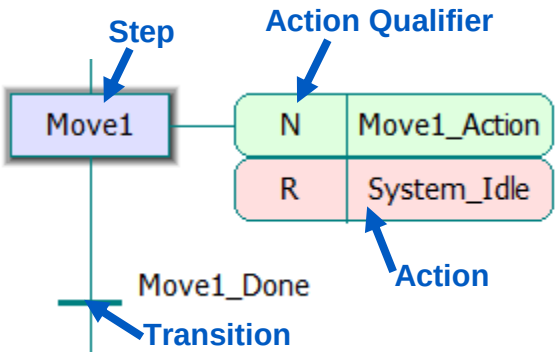
FAQ

Title: Sequence Management in SFC using Step.X variables	
Product(s): MotionWorks IEC	Doc. No. TN.MWIEC.03

Execution of SFC

Sequential Function Chart (SFC) is an IEC-61131 language that allows the user to organize the application code in an easy to follow flow diagram. The diagram shows a series of **steps** separated by **transitions**. A step will execute until one of the transitions that follow becomes TRUE. At this point, the program will move to the next step.



Each step triggers the execution of one or more **actions**. Each action has an **action qualifier** that determines when the action is executed (see Appendix for a complete list of action qualifiers).

The Step.X Variable

Behind the scenes, an SFC keeps track of which step it is on using a special variable referred to as the 'Step.X' (read Step-dot-X) variable. For example, if the step is called YASKAWA, the execution status bit is YASKAWA.X. This is a Boolean variable that indicates that a particular step is active. This status flag is not a declared variable and will not be found in any variable worksheets, but is available for use by the IEC program to make entry and exit conditions for any actions.

When the transition for a step becomes active (evaluates as TRUE), the Step.X variable for the previous step is set FALSE and the Step.X for the next step is set to TRUE. The previous step and its actions are executed one more time. This allows any clean-up that the Step may require. It is important to realize that this clean-up step happens for all actions when the step transitions.

Note: Much like the Step.X variables, there are also execution variables for actions (Action.X) and transitions (Transition.X). This document will only focus on Step.X but the other execution variables work in a similar way.

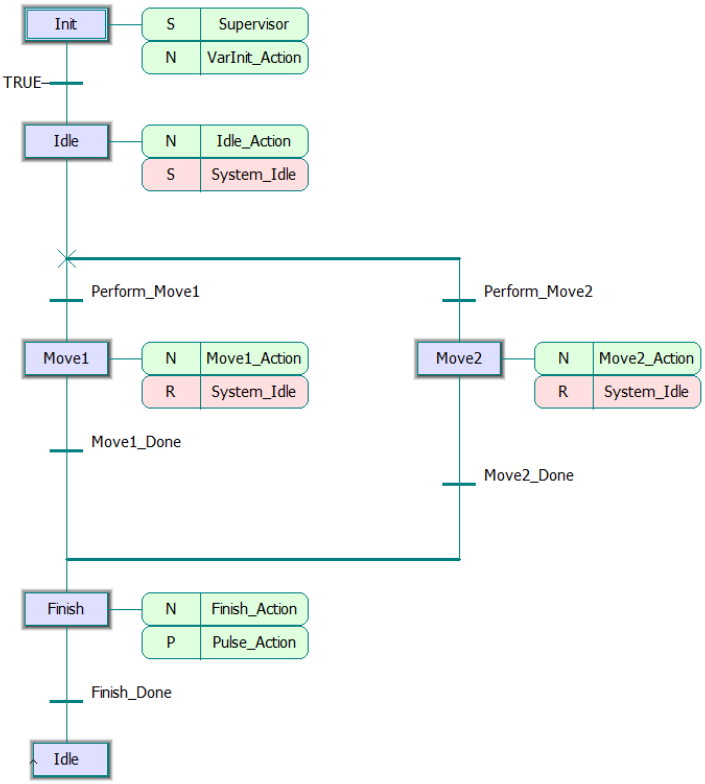
Example:

Here is an example of how the Step.X variables work for the SFC code shown below.

Title: Sequence Management in SFC using Step.X variables

Product(s): MotionWorks IEC	Doc. No. TN.MWIEC.03

1. When the code is executed, `S_Init.X` is set TRUE because it is the Start Step. The actions `Supervisor` and `VarInit_Action` are executed.
2. The transition is evaluated, and because it is TRUE the `S_Init.X` bit is set to FALSE.
3. The `Supervisor` and `VarInit_Action` actions are executed again in the next scan. (`A_Supervisor` will continue to execute each scan because the 'S' (stored) action qualifier is specified.)
4. `Idle.X` is set to TRUE and `Idle_Action` is executed each scan until either `Perform_Move1` or `Perform_Move2` become TRUE. The boolean variable `System_Idle` is set to TRUE.



5. When `Perform_Move1` becomes TRUE, `S_Idle.X` is set to FALSE and `Move1.X` is set to TRUE. `Idle_Action` is executed one more time and `Move1_Action` is executed.
6. The execution of the code continues in the same way...

Using Step.X to Initialize Variables

SFC actions can initialize variables when a step is activated by evaluating the rising edge of the Step.X variable for that step. For example, if there is a step called `Step1`, the internal variable `Step1.X` will be set to TRUE when the step is activated. By using an R_TRIG function block to evaluate the rising edge of this status variable, it is possible to initialize variables on the first pass through the action.

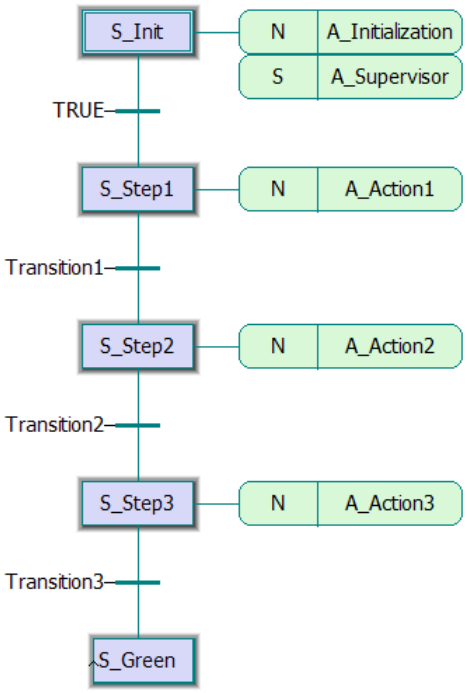
```
(* ENTRY CODE - This section only executes one time
when the step is first executed. *)

R_TRIG_1(CLK:=Step1.X);

IF R_TRIG_1.Q THEN
    Var1:=0;
    Var2:=10;
    Running:=TRUE
END_IF;


(* MAIN CODE - This section runs every scan while the
step is active. *)

IF Step1.X THEN
    Var1 := Var1 + 1;
    (* Code goes here *)
END_IF;


(* EXIT CODE - This section runs for only one scan
during the clean-up. *)

F_TRIG_1(CLK:=Step1.X);

IF F_TRIG_1.Q THEN
    Running := FALSE;
END_IF;
```

It is also possible to use the F_TRIG function block

along with the Step.X variable to create code that only executes for one scan during the clean-up of the action.

Resetting SFC

To handle a power failure, E-Stop, alarm, or other unexpected interruption to an SFC sequence, Yaskawa recommends executing a supervisory routine to reset the Step.X bits of each Step and therefore reset the SFC. The simplest place to add this is in the initialization Step of the SFC as an Action with an "S" (stored) action qualifier, see attached.



The A_Supervisor Action runs each scan because of the "S" (stored) action qualifier. This code simply checks for a Boolean reset bit to set the each Step.X bit to FALSE, and then setting the first step's Step. X bit to TRUE.

Care must be taken that any critical variables in the SFC are re-initialized during the SFC reset process.

```
1    (* Code from A_Supervisor action *)
2
3    (* If the SFC is interrupted abnormally by
4    an unexpected or non-sequential event, this
5    code will abort the current SFC step and start
6    again from the beginning of the SFC sequence.
7
8    Be sure to re-initialize all variables in the
9    start step to ensure proper sequence operation. *)
10
11   IF SFC_Reset THEN
12       S_Step1.X:=FALSE;
13       S_Step2.X:=FALSE;
14       S_Step3.X:=FALSE;
15       S_Init.X:=TRUE;
16   END_IF;
```

Title: Sequence Management in SFC using Step.X variables	
Product(s): MotionWorks IEC	Doc. No. TN.MWIEC.03

Appendix: Action Qualifiers

- **N**: non-stored - The action code body is executed or the Boolean variable is set as long as the step is active.
- **R**: overriding reset – When the step has previously been executed with the S (including DS, DS, and SL) qualifier, the R qualifier will stop the execution of the code or reset the Boolean variable.
- **S**: set (stored) - The action code body is executed or the Boolean variable is set. This state is stored as soon as the step becomes active. It can only be reset explicitly by associating the same action to a different step using the qualifier 'R'.
- **L**: time limited - The action code body is executed or the Boolean variable is set as long as the step is active but maximal for the fixed time interval.
- **D**: time delayed - The action code body is executed or the Boolean variable is set after the fixed delay time has elapsed. The action remains active as long as the step is active. If the step is active shorter than the fixed delay time the action does not become active.
- **P**: pulse - As soon as the step is active the action code body is executed or the Boolean variable is set for one operating cycle. (Note: The code body will then execute for one additional operating cycle with the Step.X variable FALSE.)
- **SD**: stored and time delayed - the action code body is executed or the Boolean variable is stored and set when the fixed delay time has elapsed after the step activation, even if the step becomes inactive. The action remains active until it is reset. If the step is active shorter than the fixed delay time the action becomes active anyway.
- **DS**: delayed and stored - The action code body is executed or the Boolean variable is set when the fixed delay time has elapsed after the step activation. The action remains active until it is reset. If the step is active shorter than the fixed delay time the action does not become active.
- **SL**: stored and time limited - The action code body is executed or the Boolean variable is set and stored for a fixed time interval as soon as the step is active. If the step is active shorter than the time interval the action is active for the whole time interval anyway. If the action is reset during the time interval the action becomes inactive as soon as the action is reset.