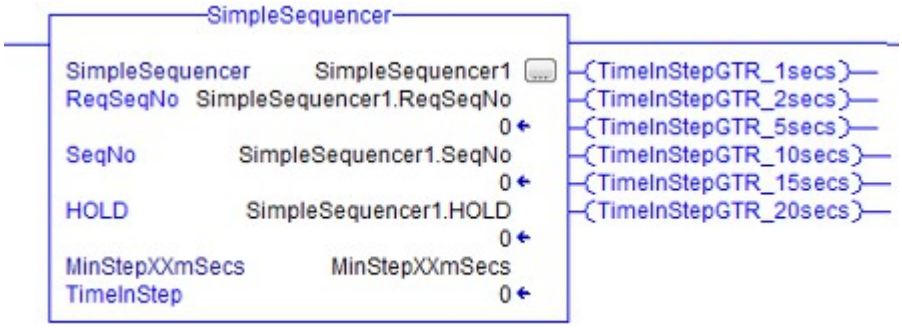
SimpleSequencer

from

www.AllertonEPS-Software.co.uk

The word “Simple” is used to indicate that **SimpleSequencer** is SIMPLE to understand and use, but is flexible and provides just the right amount of control of the sequence to be able to handle small to medium sized sequences.



At it's heart the **SimpleSequencer** controls a register called SeqNo (Step Number).

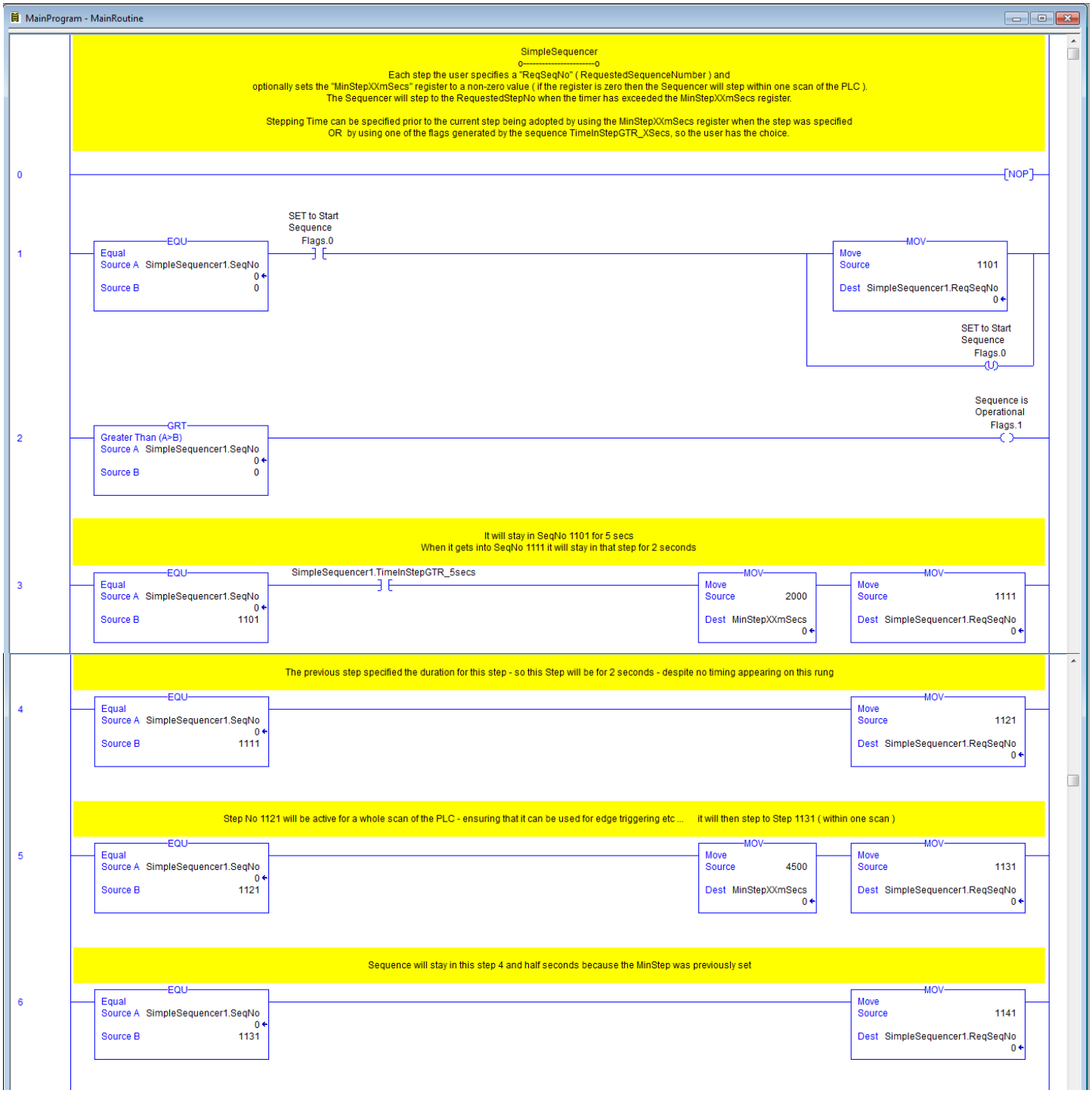
There are two ways to change SeqNo :-

- The normal (controlled) way is to **request** to change the SeqNo (using **ReqSeqNo**) The change of SeqNo will then be performed on the next scan, providing the sequence isn't in HOLD, and the **MinStepXXmSecs** (time) **has been satisfied.**
- **The abnormal way is to change the SeqNo immediately (external to the block) - this is then adopted as the new SeqNo without waiting for MinStepXXmSecs and is independent of HOLD. This is used for CRITICAL functions – Emergency Shutdowns.** Once the sequence has jumped to the Emergency Shutdown section of the sequence, then this can be controlled through the normal operation of the sequence (or it can done externally to this sequence.)

  For Critical functions it is important that the SeqNo is not locked and is free to be changed externally.

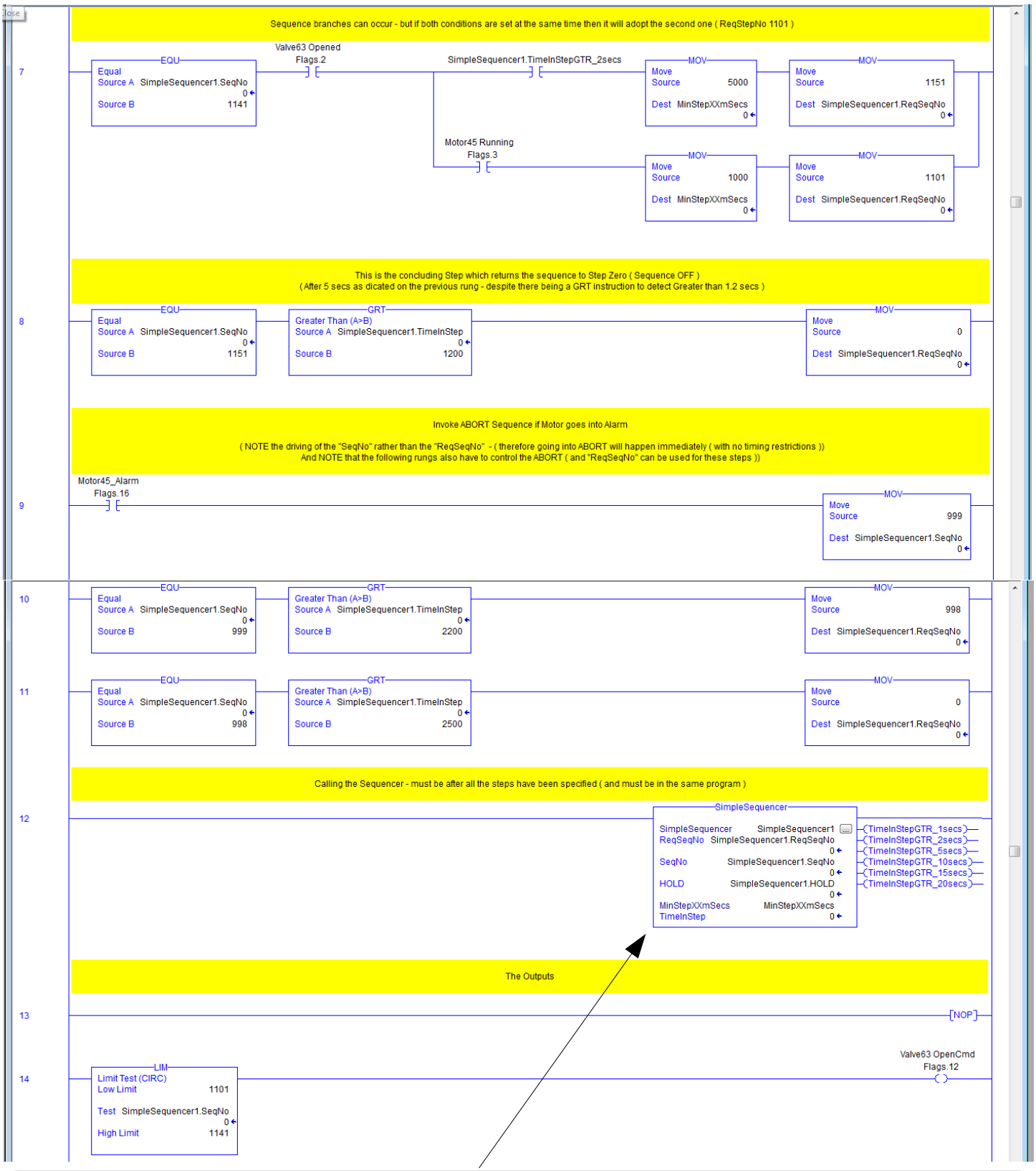
AllertonEPS-Software.co.uk
63 The Rock, Helsby
Cheshire WA6 9AS
Greg +44 7944 296136

This is typical code that could exist outside the Function block



AllertonEPS-Software.co.uk
63 The Rock, Helsby
Cheshire WA6 9AS
Greg +44 7944 296136

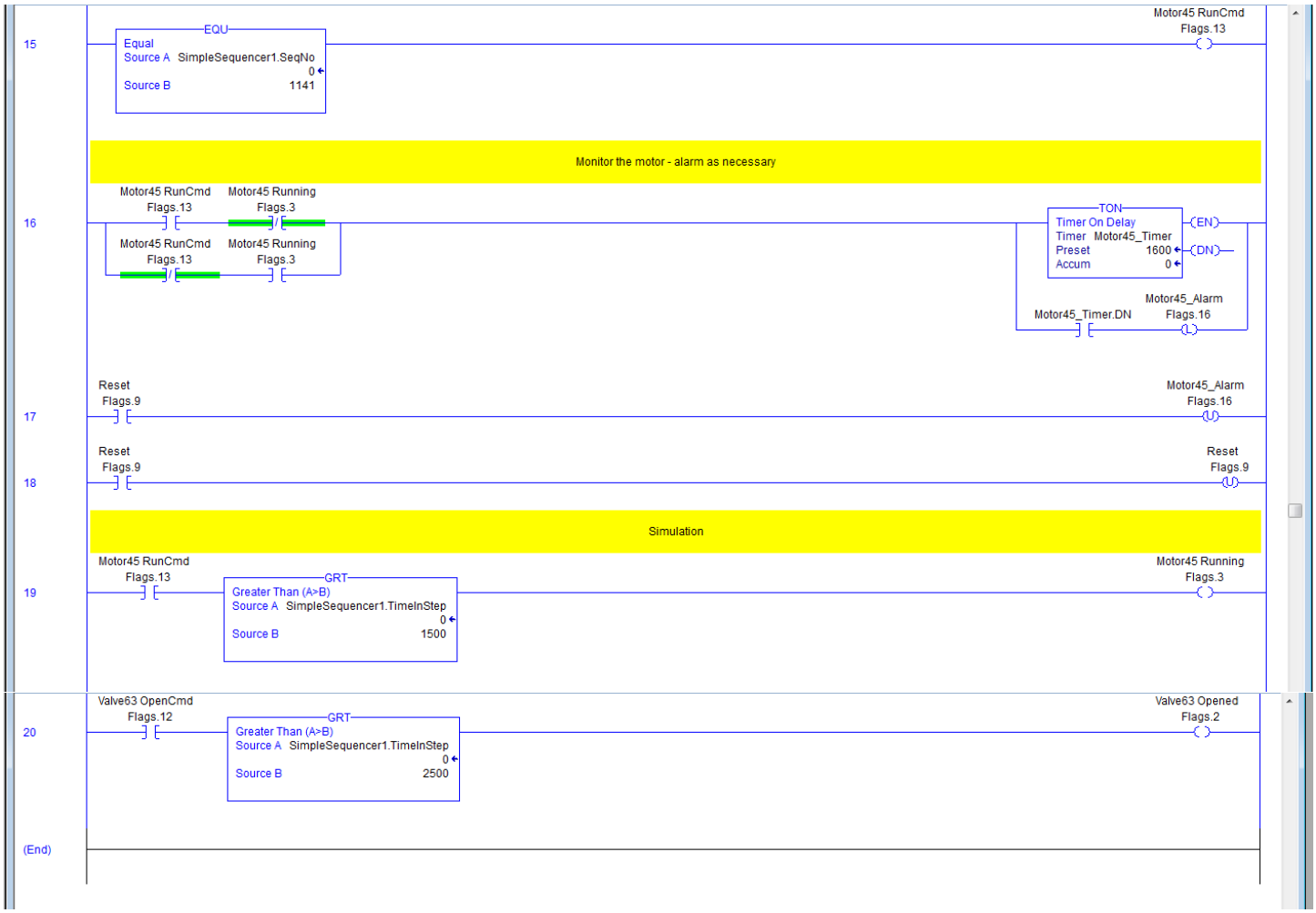
This is typical code that could exist outside the Function block (continued)



Calling the function block has to be at the end of the sequencing

Then the outputs – putting the rungs in this order means that the outputs can respond immediately to the change of the SeqNo (providing the SimpleSequencer function is processed synchronously within the main code)

And after the sequencer has been called – the Outputs are defined – describing what action to take on each step (or collection of steps (as in Rung 14) **).**



This the code in Rockwell statement form (Siemens would need translating)

NOP(); EQU(SimpleSequencer1.SeqNo,0)XIC(Flags.0)
[MOV(1101,SimpleSequencer1.ReqSeqNo),OTU(Flags.0)]; GRT(SimpleSequencer1.SeqNo,0)OTE(Flags.1);
EQU(SimpleSequencer1.SeqNo,1101)XIC(SimpleSequencer1.TimeInStepGTR_5secs)MOV(2000,MinStepXXmSecs)MOV(1111,SimpleSequencer1.ReqSeqNo);
EQU(SimpleSequencer1.SeqNo,1111)MOV(1121,SimpleSequencer1.ReqSeqNo);
EQU(SimpleSequencer1.SeqNo,1121)MOV(4500,MinStepXXmSecs)MOV(1131,SimpleSequencer1.ReqSeqNo);
EQU(SimpleSequencer1.SeqNo,1131)MOV(1141,SimpleSequencer1.ReqSeqNo);
EQU(SimpleSequencer1.SeqNo,1141)XIC(Flags.2)
[XIC(SimpleSequencer1.TimeInStepGTR_2secs)MOV(5000,MinStepXXmSecs)MOV(1151,SimpleSequencer1.ReqSeqNo),XIC(Flags.3)MOV(1000,MinStepXXmSecs)MOV(1101,SimpleSequencer1.ReqSeqNo)];
EQU(SimpleSequencer1.SeqNo,1151)GRT(SimpleSequencer1.TimeInStep,1200)MOV(0,SimpleSequencer1.ReqSeqNo); XIC(Flags.16)MOV(999,SimpleSequencer1.SeqNo);
EQU(SimpleSequencer1.SeqNo,999)GRT(SimpleSequencer1.TimeInStep,2200)MOV(998,SimpleSequencer1.ReqSeqNo);
EQU(SimpleSequencer1.SeqNo,998)GRT(SimpleSequencer1.TimeInStep,2500)MOV(0,SimpleSequencer1.ReqSeqNo);
SimpleSequencer(SimpleSequencer1,SimpleSequencer1.ReqSeqNo,SimpleSequencer1.SeqNo,SimpleSequencer1.HOLD,MinStepXXmSecs); NOP(); LIM(1101,SimpleSequencer1.SeqNo,1141)OTE(Flags.12);
EQU(SimpleSequencer1.SeqNo,1141)OTE(Flags.13); [XIC(Flags.13)XIO(Flags.3),XIO(Flags.13)XIC(Flags.3)]
[TON(Motor45_Timer,?,?),XIC(Motor45_Timer.DN)OTL(Flags.16)]; XIC(Flags.9)OTU(Flags.16);
XIC(Flags.9)OTU(Flags.9); XIC(Flags.13)GRT(SimpleSequencer1.TimeInStep,1500)OTE(Flags.3);
XIC(Flags.12)GRT(SimpleSequencer1.TimeInStep,2500)OTE(Flags.2);

Inside the Sequencer block …

Below is the contents of the Rockwell Comments - basically repeating what has already been described earlier ...

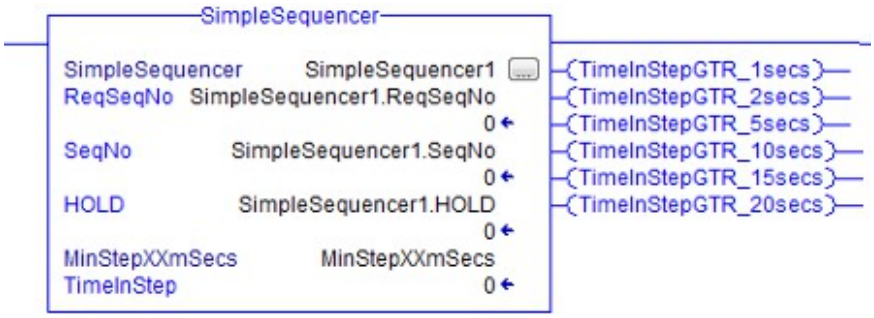
For Each step the user specifies a "ReqSeqNo" (RequestedSequenceNumber)
and optionally sets the "MinStepXXmSecs" register to a non-zero value (if the register is zero then the Sequencer will step within one scan of the PLC).
The Sequencer will step to the Requested (ReqSeqNo) Sequence number when the timer has exceeded the MinStepXXmSecs register.

Stepping Time can be specified prior to the current step being adopted by using the MinStepXXmSecs register when the step was specified
OR by using one of the flags generated by the sequence TimeInStepGTR_XSecs in the rung controlling the stepping (outside this routine), so the user has the choice.

User can change "SeqNo" instead of "ReqSeqNo" when the step has to change immediately without waiting for any timing (like ABORT conditions).

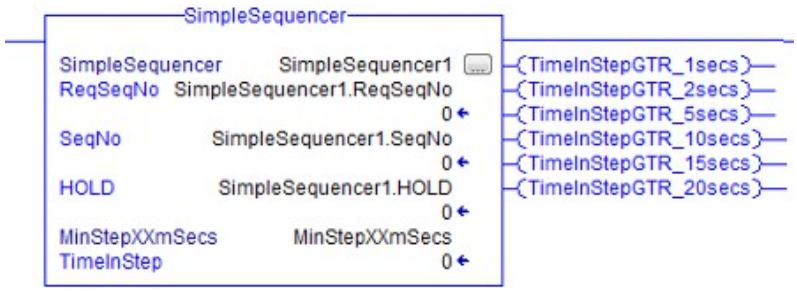
"Hold" will freeze the Step timer (at all times) and stop stepping (using the standard "ReqSeqNo") - So use with caution.

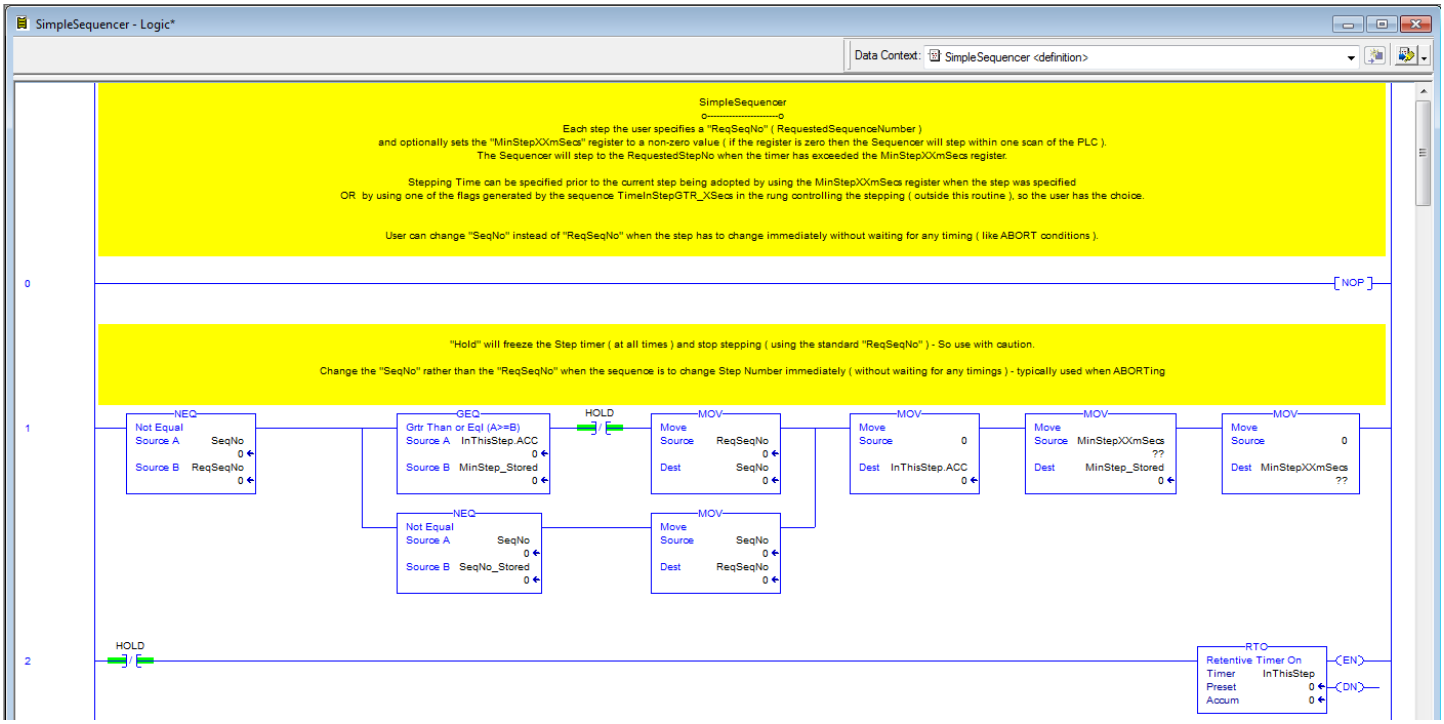
Change the "SeqNo" rather than the "ReqSeqNo" when the sequence is to change Step Number immediately (without waiting for any timings) - typically used when ABORTing.



Inside the Sequencer block …

Rung 1 is the most important rung in the block.





Rung 1 only operates if the ReqSeqNo and the SeqNo are different.

There are then two routes …

The normal route (branch) includes the Time and the HOLD,

Whilst the second detects an external change of the SeqNo and acts immediately.

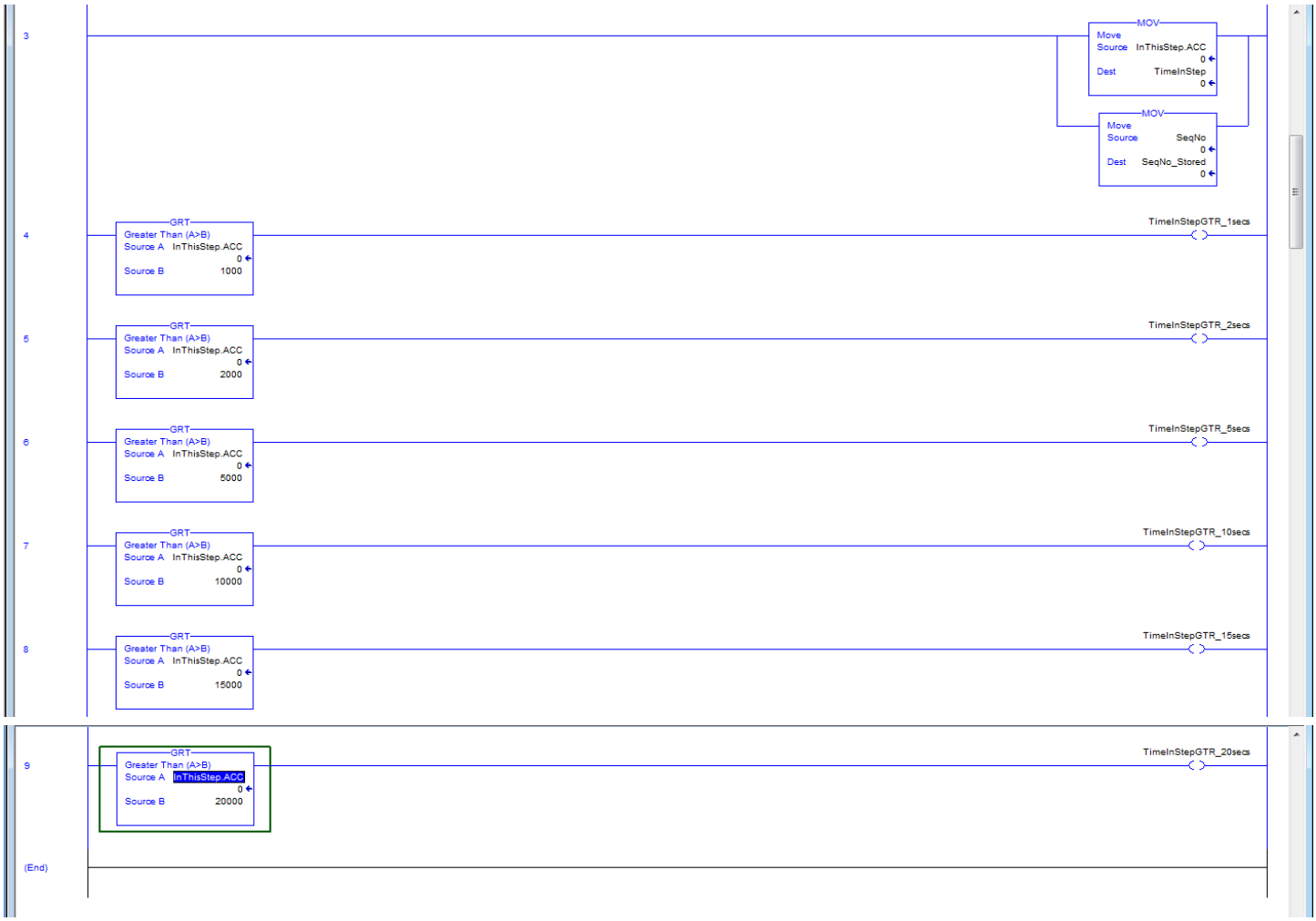
If both routes are valid – the Abnormal route will be adopted – thus is an important design feature.

At this point it is probably worth stating …

“Whilst every care has been afforded to the accuracy of this code, the author cannot be held liable for any onwards cost associated with using this code. The risk to ensure that it operates correctly under all circumstances lies with the user of the code.”

Inside the Sequencer block (continued) …

Rung 4 onwards just generates useful flags to aid control of the sequence



Inside the Sequencer block …

NOP(); NEQ(SeqNo,ReqSeqNo)
[GEQ(InThisStep.ACC,MinStep_Stored)XIO(HOLD)MOV(ReqSeqNo,SeqNo),NEQ(SeqNo,SeqNo_Stored)MOV(SeqNo,ReqSeqNo)]MOV(0,InThisStep.ACC)MOV(MinStepXXmSecs,MinStep_Stored)MOV(0,MinStepXXmSecs);
XIO(HOLD)RTO(InThisStep,?,?); [MOV(InThisStep.ACC,TimeInStep),MOV(SeqNo,SeqNo_Stored)];
GRT(InThisStep.ACC,1000)OTE(TimeInStepGTR_1secs);
GRT(InThisStep.ACC,2000)OTE(TimeInStepGTR_2secs);
GRT(InThisStep.ACC,5000)OTE(TimeInStepGTR_5secs);
GRT(InThisStep.ACC,10000)OTE(TimeInStepGTR_10secs);
GRT(InThisStep.ACC,15000)OTE(TimeInStepGTR_15secs);
GRT(InThisStep.ACC,20000)OTE(TimeInStepGTR_20secs);

The key rung of That's It – That's all there is to it

Most other versions of a sequencer make a Dog's Dinner of it.

If you want an aborting sequence – then create a branch of the sequence – SimpleSequencer doesn't need to know that it is aborting.

There is no need for RUN as that is assumed to be the opposite of HOLD.