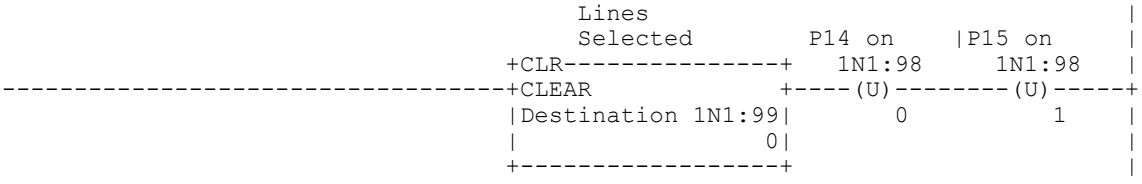
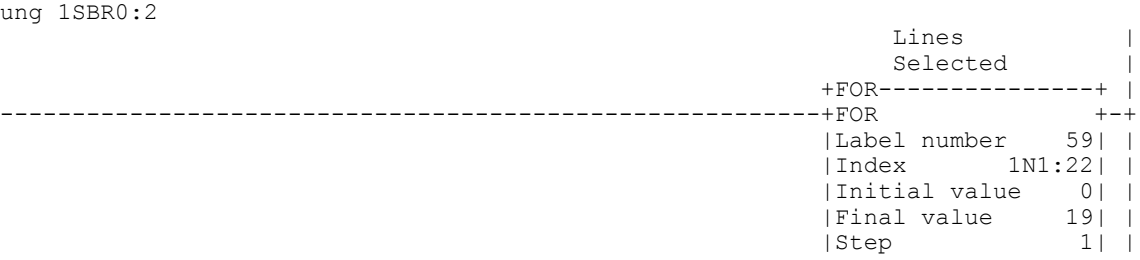
Rung 1SBR0:0
Subroutine with parameters

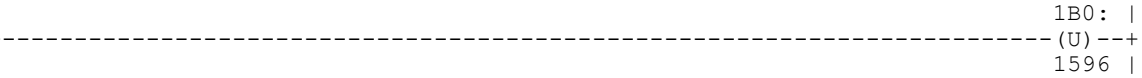
Rung 1SBR0:1



Rung 1SBR0:2

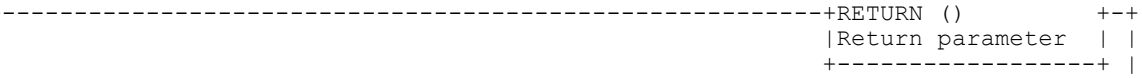


Rung 1SBR0:3

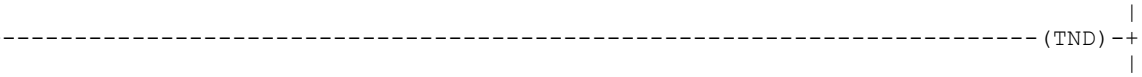


Rung 1SBR0:4

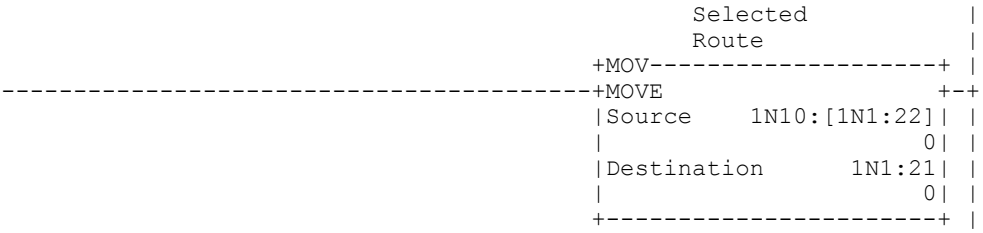
Rung 1SBR0:5



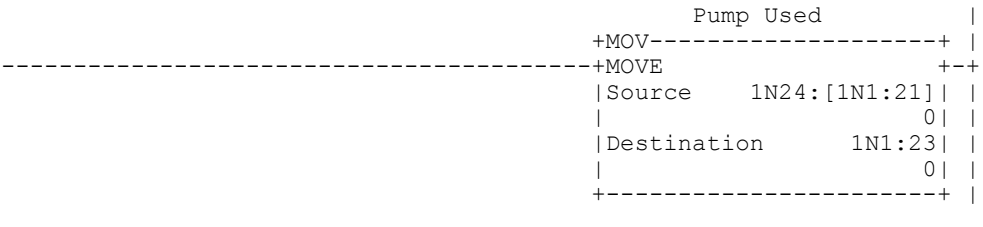
Rung 1SBR0:6



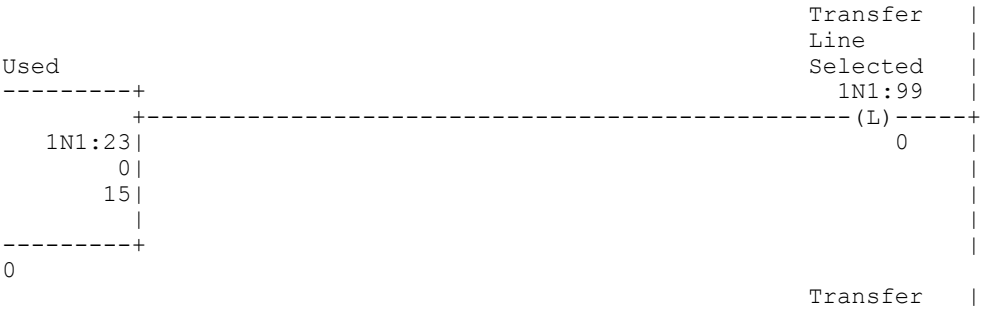
Rung 1SBR0:7



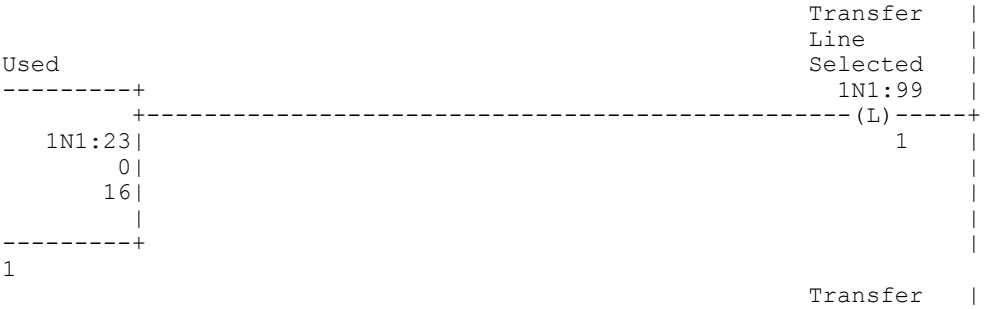
Rung 1SBR0:8



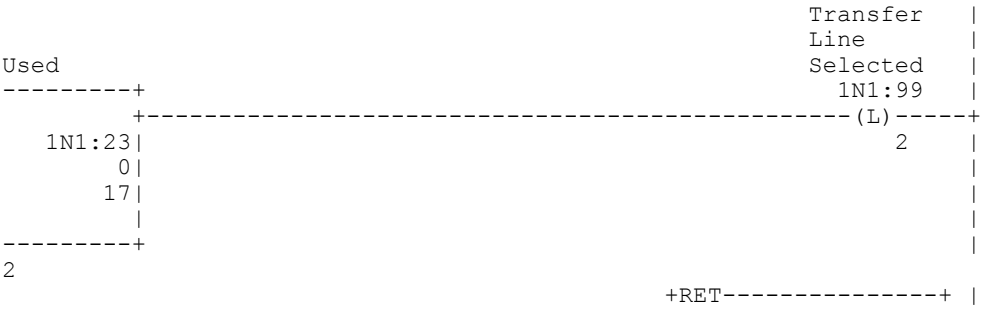
Rung 1SBR0:9



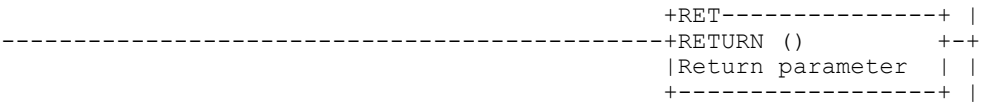
Rung 1SBR0:10



Rung 1SBR0:11



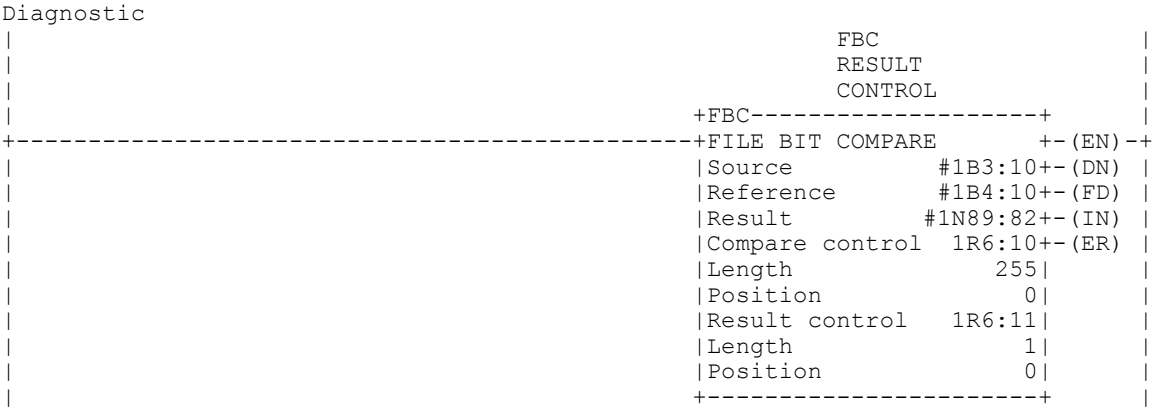
Rung 1SBR0:12



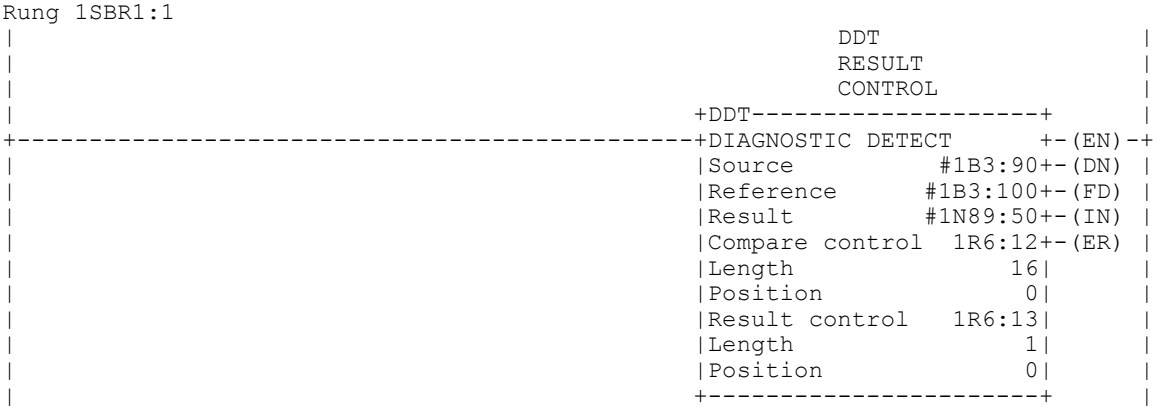
Rung 1SBR0:13

[END OF FILE]

Rung 1SBR1:0
Diagnostic



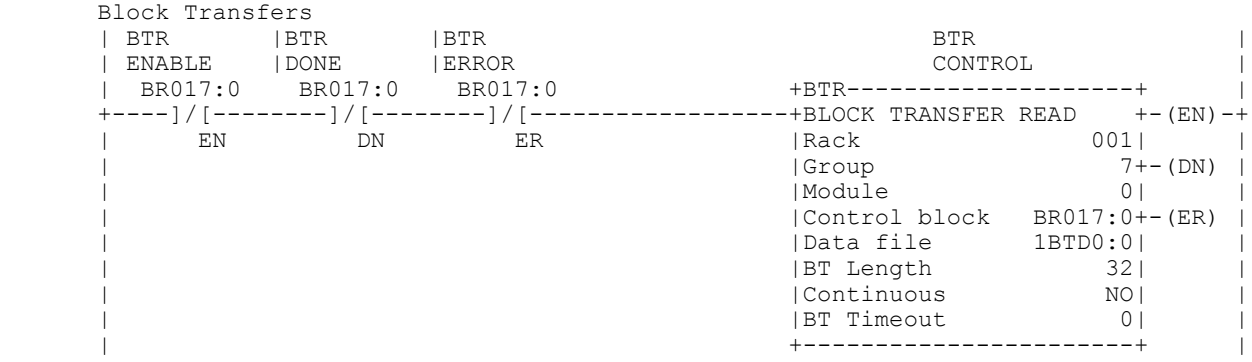
Rung 1SBR1:1



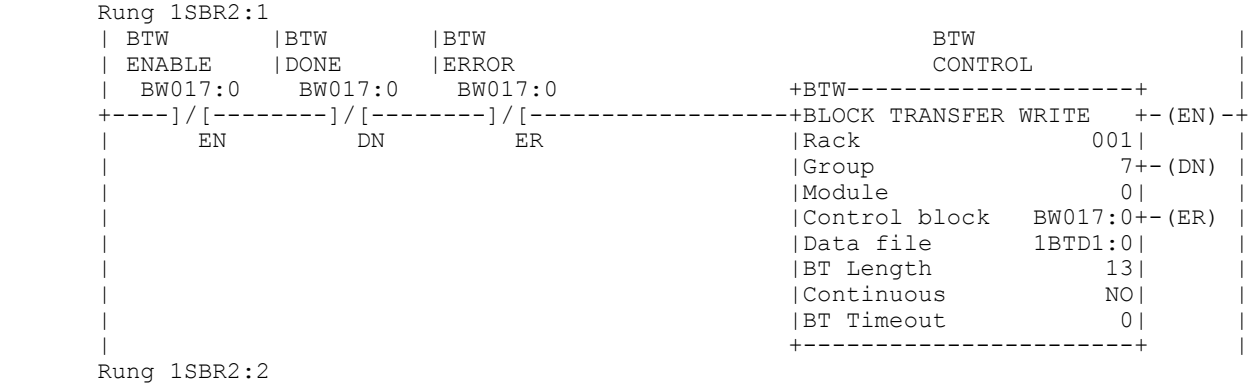
Rung 1SBR1:2

[END OF FILE]

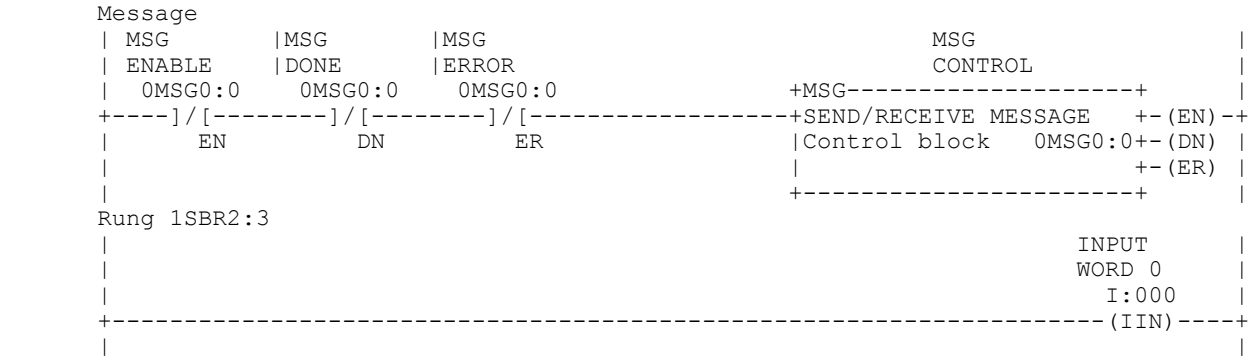
Rung 1SBR2:0
Block Transfers



Rung 1SBR2:1



Rung 1SBR2:2
Message

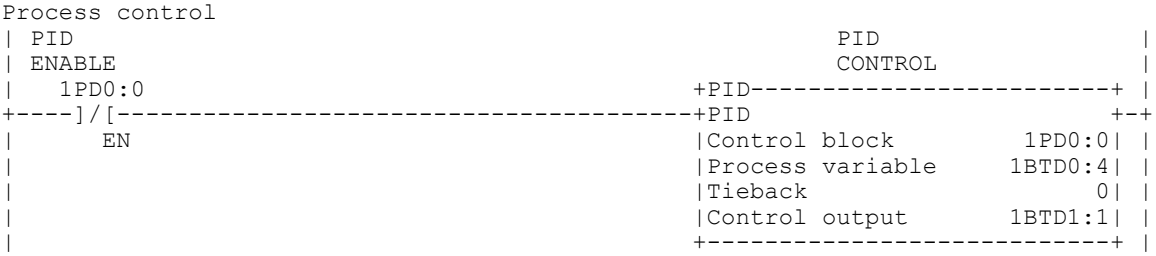


Rung 1SBR2:3

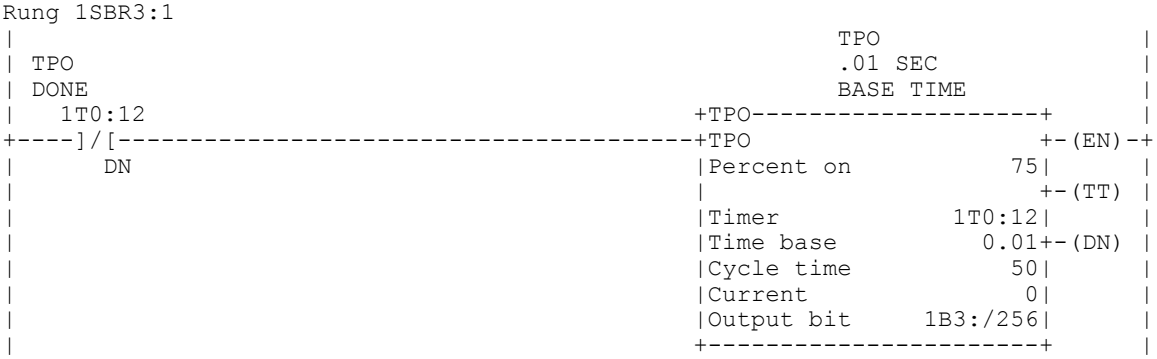
Rung 1SBR2:4

[END OF FILE]

Rung 1SBR3:0
Process control



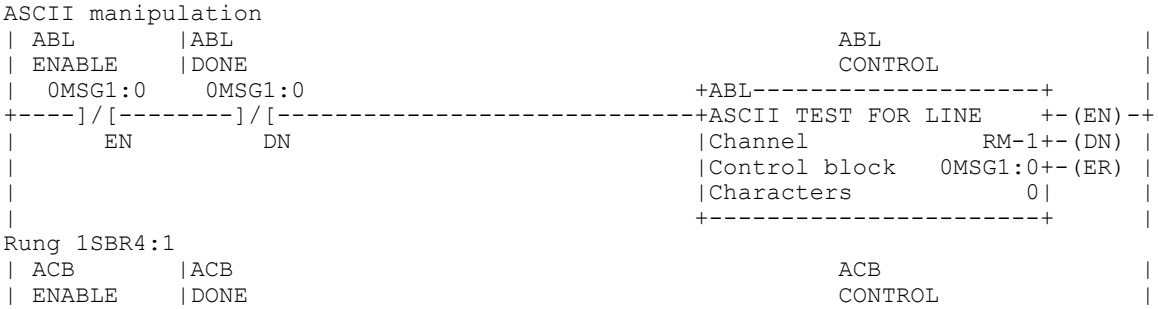
Rung 1SBR3:1



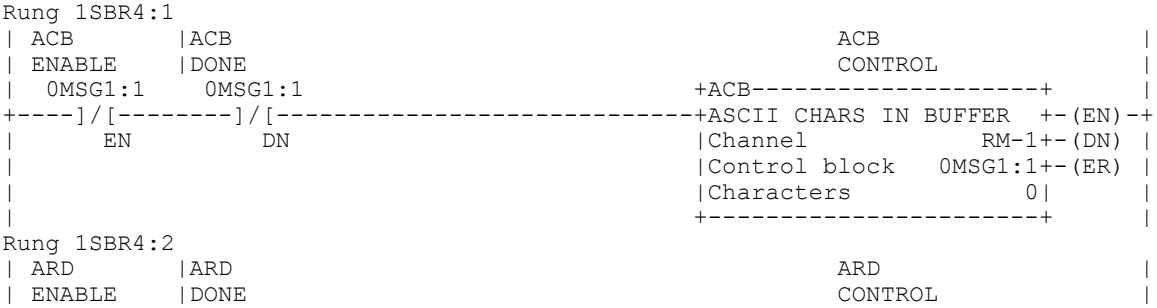
Rung 1SBR3:2

[END OF FILE]

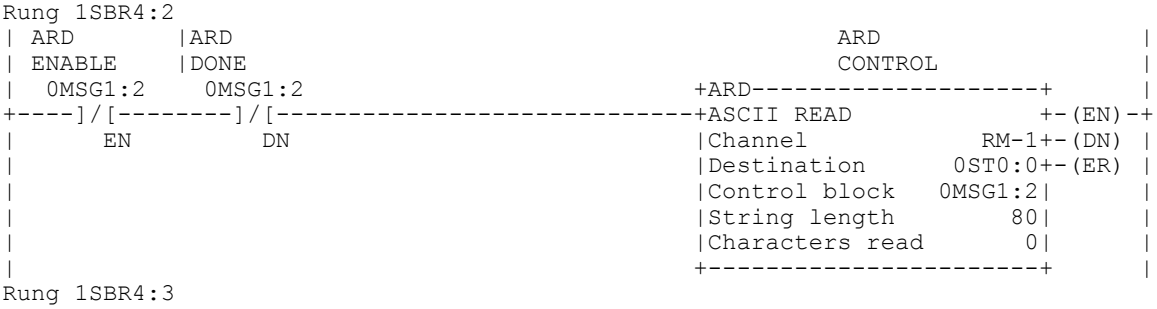
Rung 1SBR4:0
ASCII manipulation



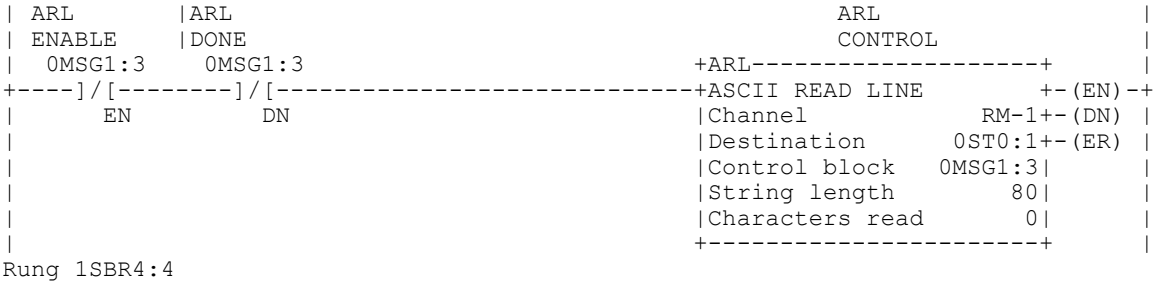
Rung 1SBR4:1



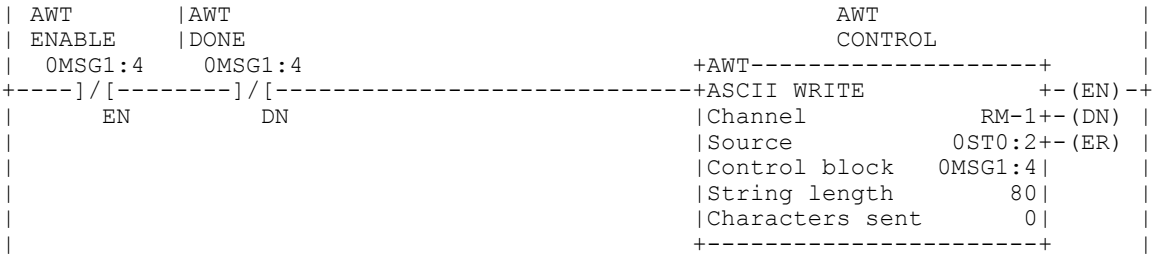
Rung 1SBR4:2



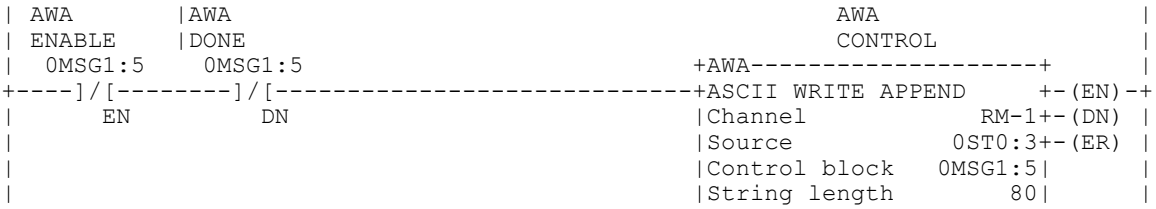
Rung 1SBR4:3



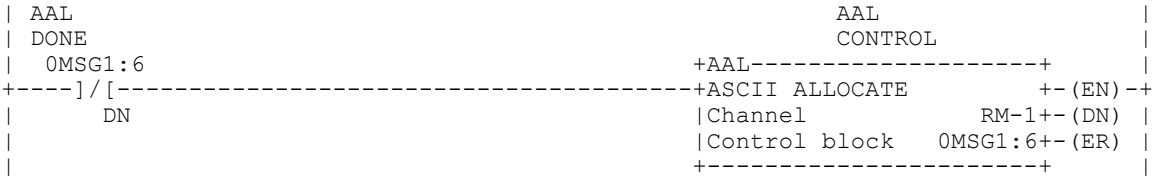
Rung 1SBR4:4



Rung 1SBR4:5

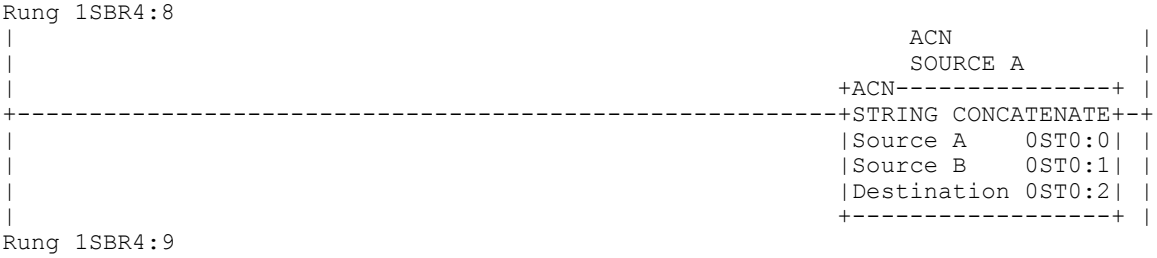


Rung 1SBR4:6



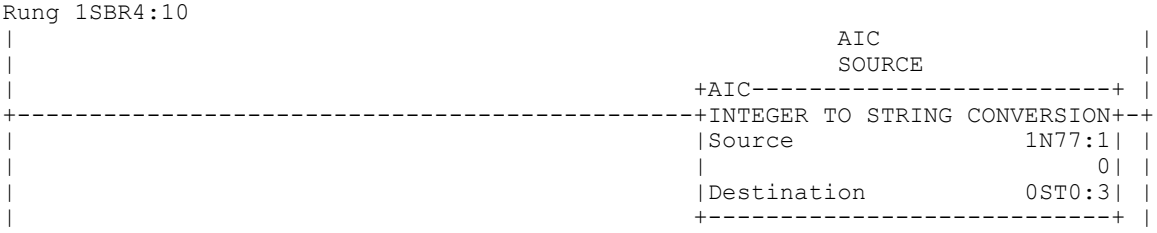
Rung 1SBR4:7

Rung 1SBR4:8

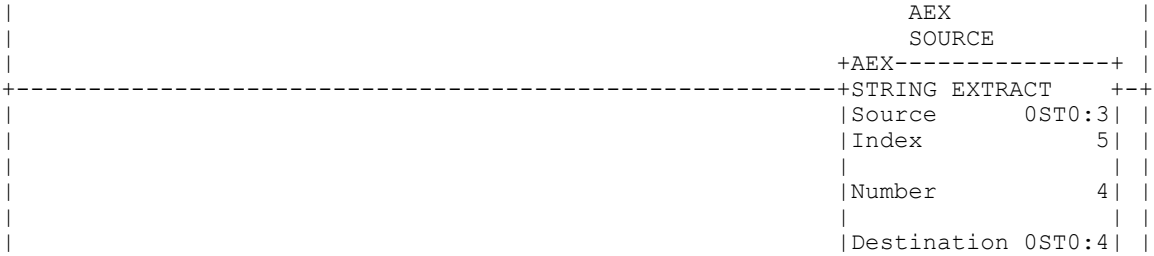


Rung 1SBR4:9

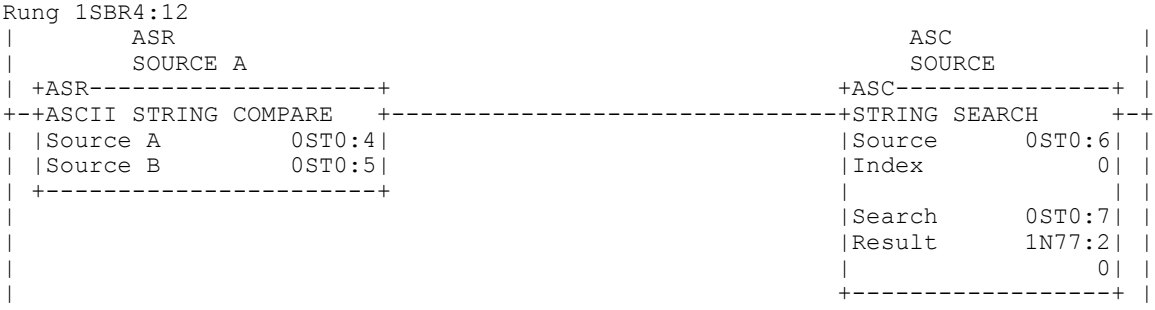
Rung 1SBR4:10



Rung 1SBR4:11



Rung 1SBR4:12



Rung 1SBR4:13

[END OF FILE]

```
Rung 1TRN0:0
|                                                                              |
+-[AFI]-------------------------------------------------------------------[EOT]-+
|                                                                              |
Rung 1TRN0:1
|                                                                              |
+--------------------------------[END OF FILE]-----------------------------------+
|                                                                              |
```

Rung 1STEP0:0
This is a rung comment.

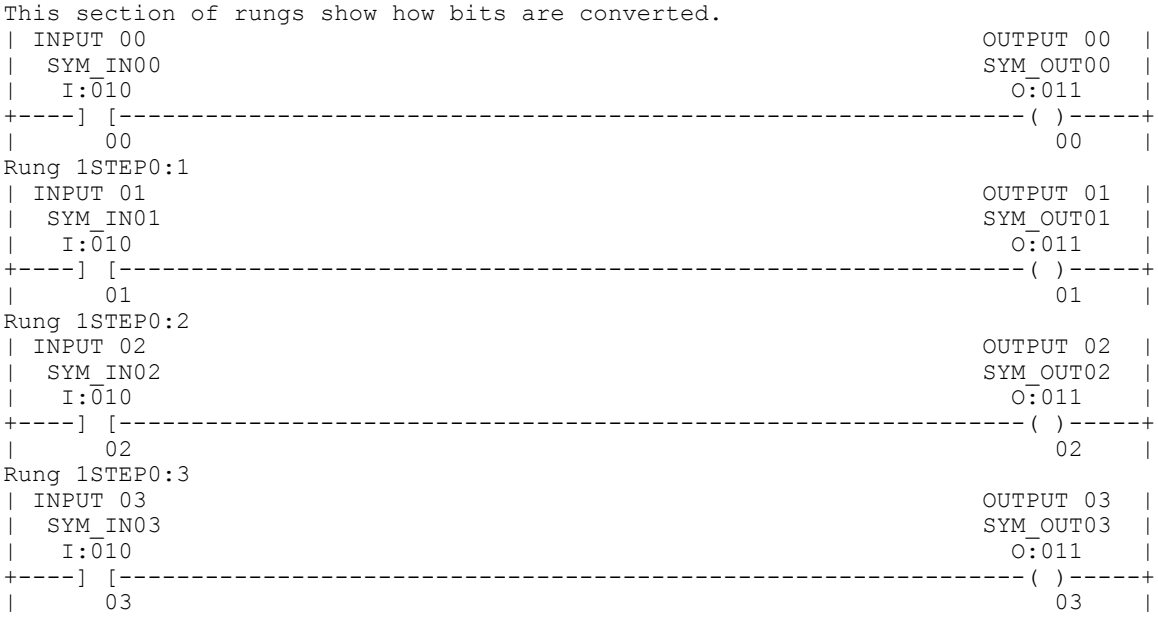
This program is totally useless...

as far as controlling any process. Now on the other hand if you
would like to see how different instructions are converted read on.

This program was converted using the DHRIO option. The racks get converted to
full racks on a universal Remote I/O. Bits in the rack range will be remapped
to I/O. Bits outside of this range will point to the INT array.

I/O bit addresses use the format: _RRS:I.Data[M].B
where:
RR = rack in octal
S = Starting module group
I = Input (or O for output)
M = Module group offset
B = Terminal number in decimal

This section of rungs show how bits are converted.



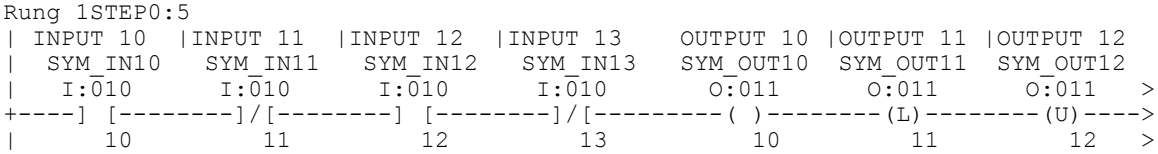
Rung 1STEP0:1

Rung 1STEP0:2

Rung 1STEP0:3

Rung 1STEP0:4

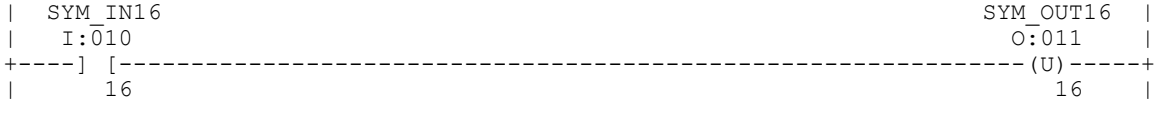
Rung 1STEP0:5



Rung 1STEP0:6

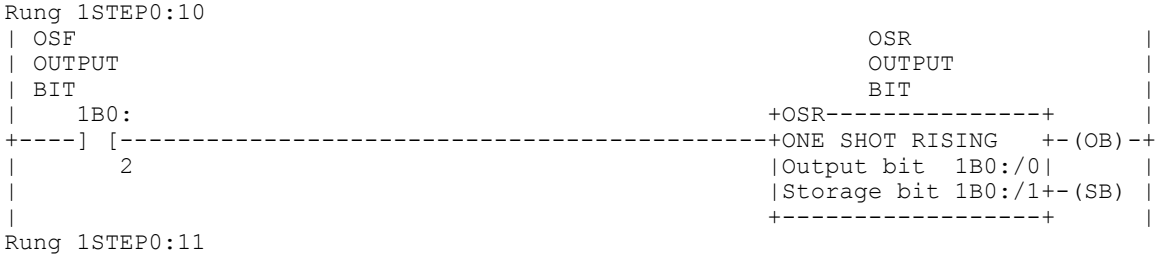
Rung 1STEP0:7

Rung 1STEP0:8

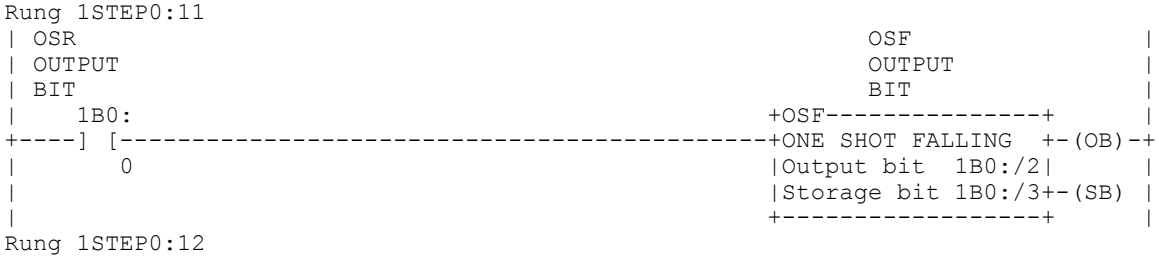


Rung 1STEP0:9

Rung 1STEP0:10

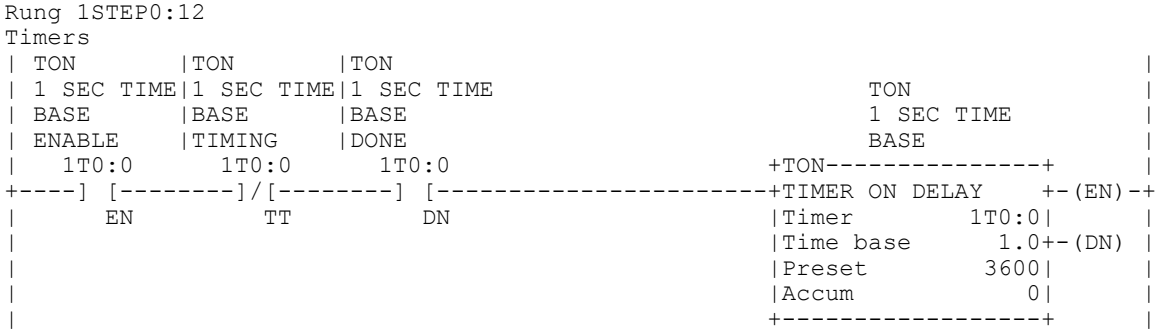


Rung 1STEP0:11

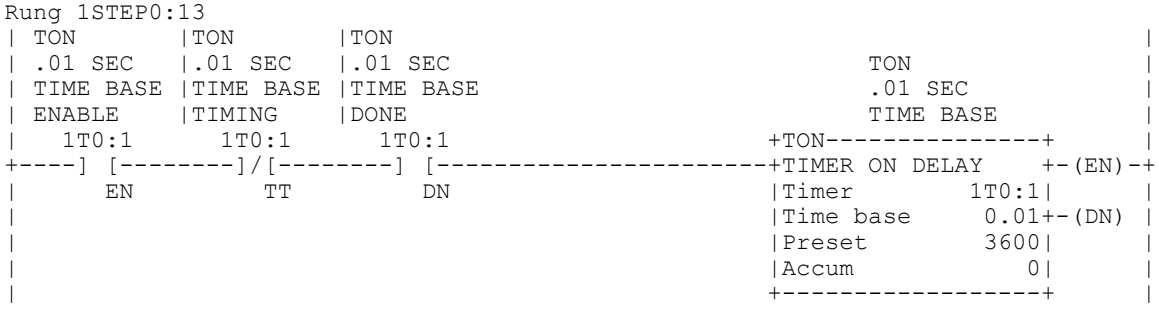


Rung 1STEP0:12

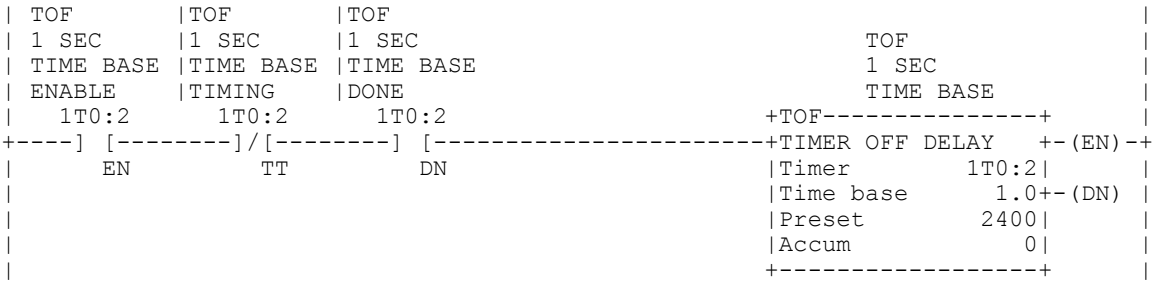
Timers



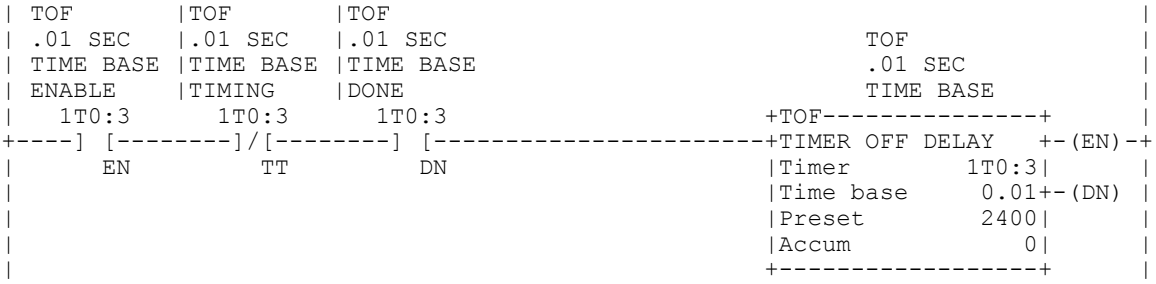
Rung 1STEP0:13



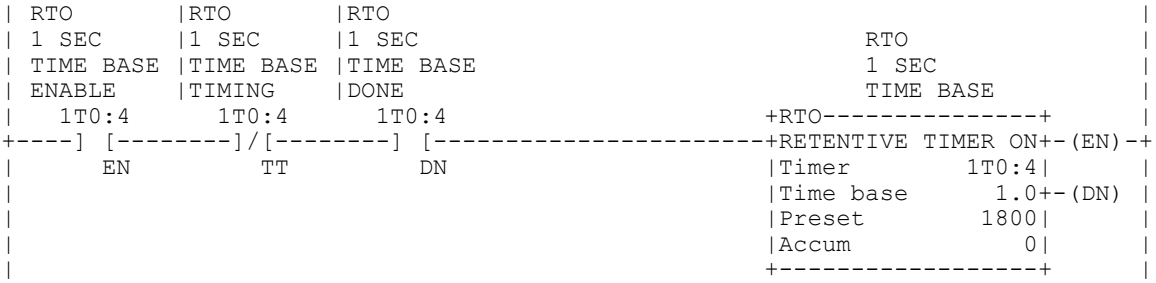
Rung 1STEP0:14



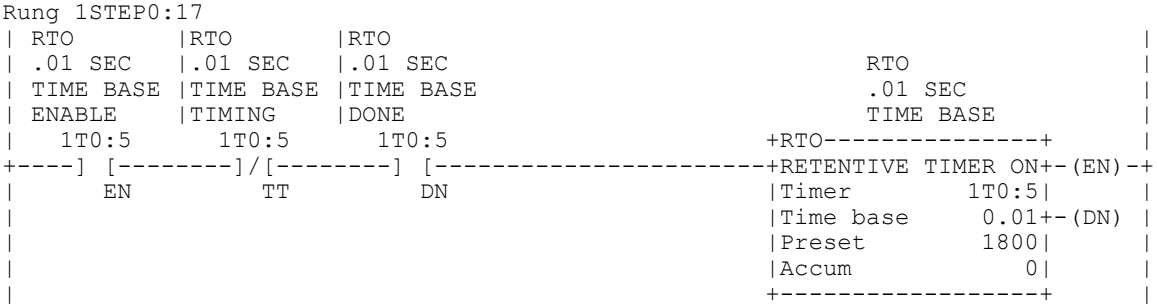
Rung 1STEP0:15



Rung 1STEP0:16

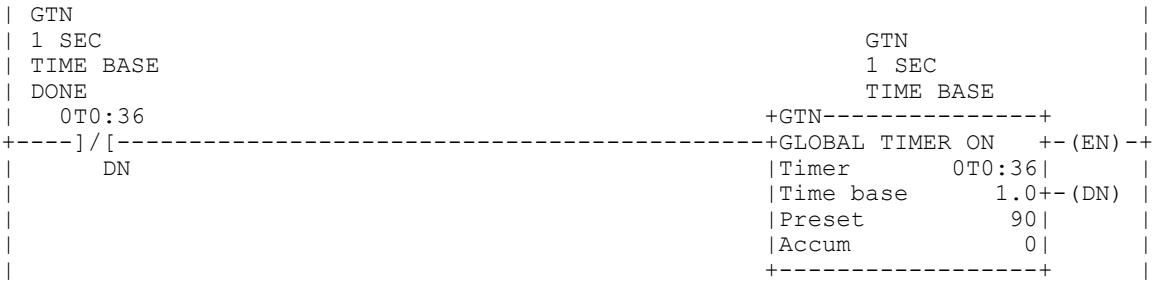


Rung 1STEP0:17

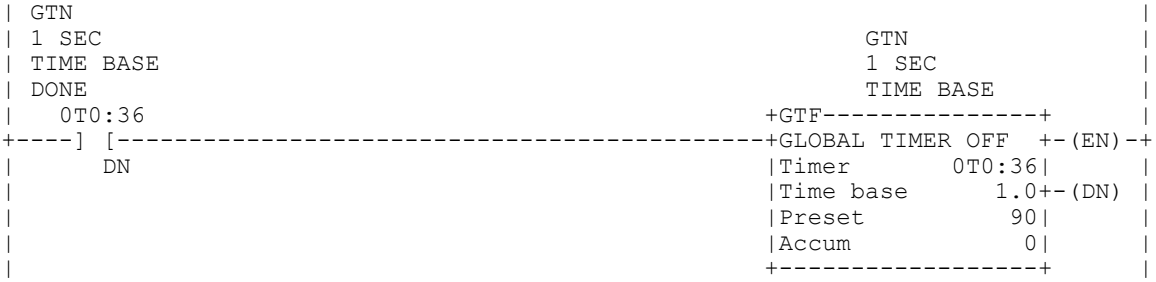


Rung 1STEP0:18

Rung 1STEP0:19

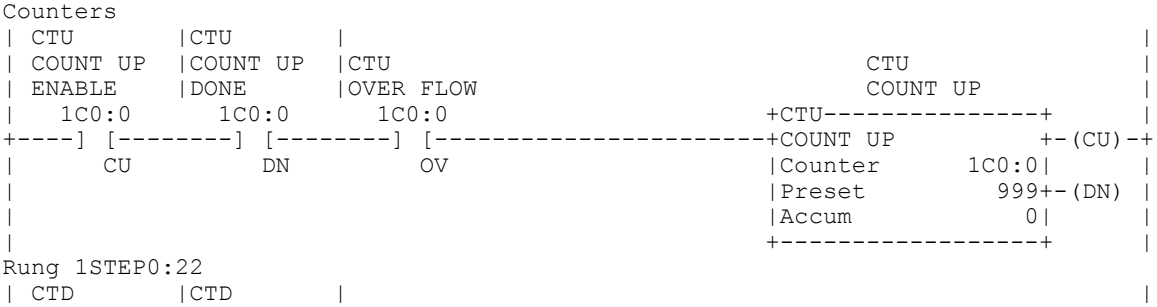


Rung 1STEP0:20

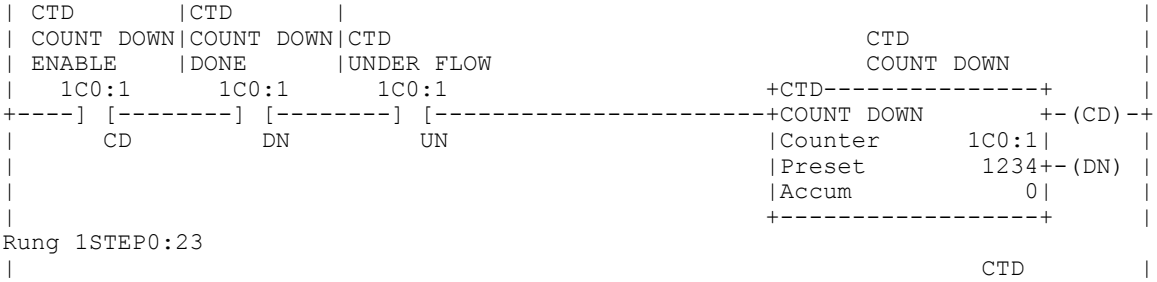


Rung 1STEP0:21

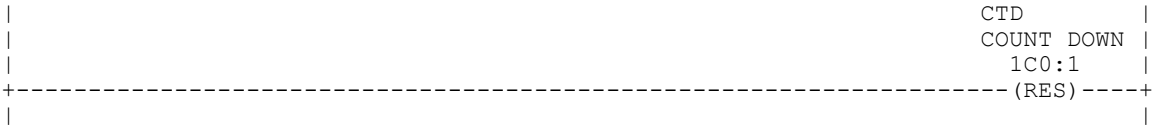
Counters



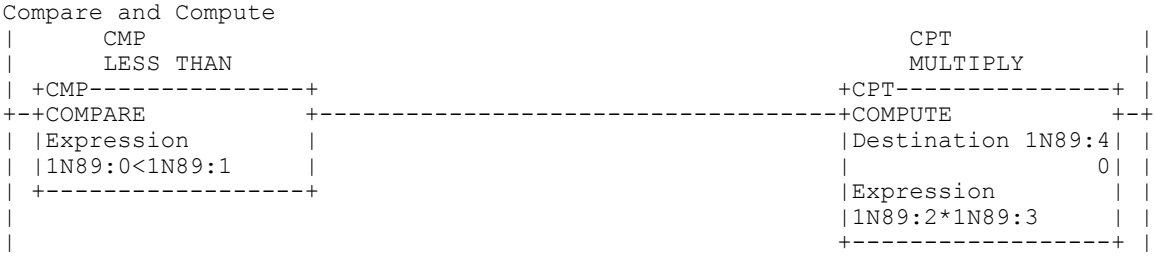
Rung 1STEP0:22



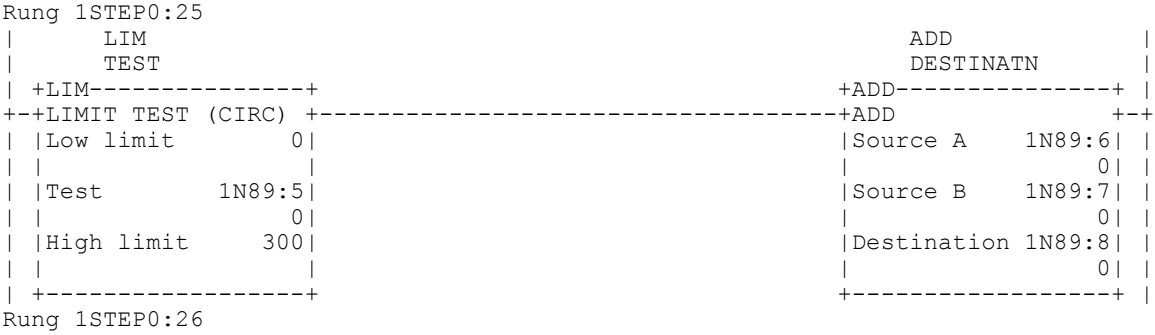
Rung 1STEP0:23



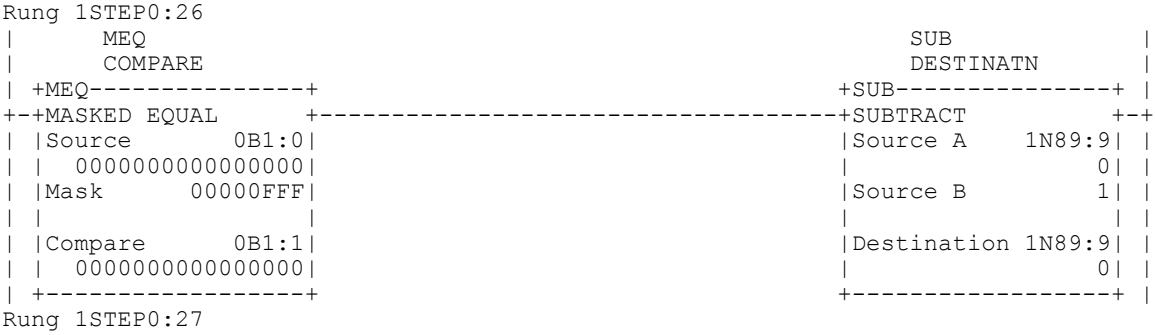
Rung 1STEP0:24
Compare and Compute



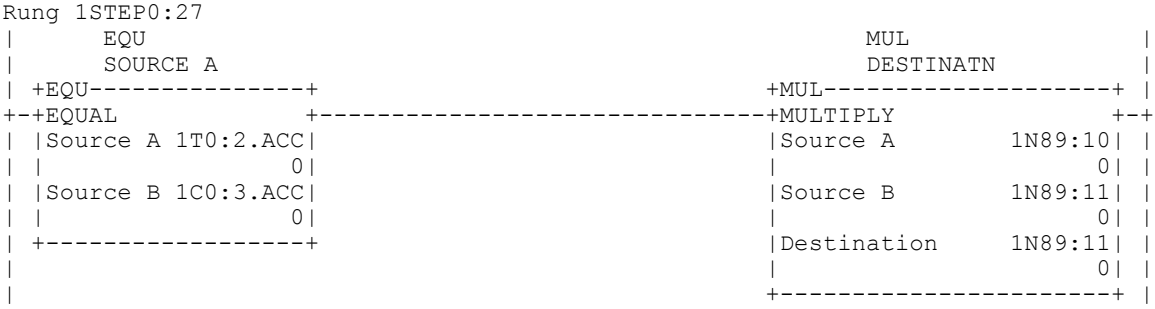
Rung 1STEP0:25



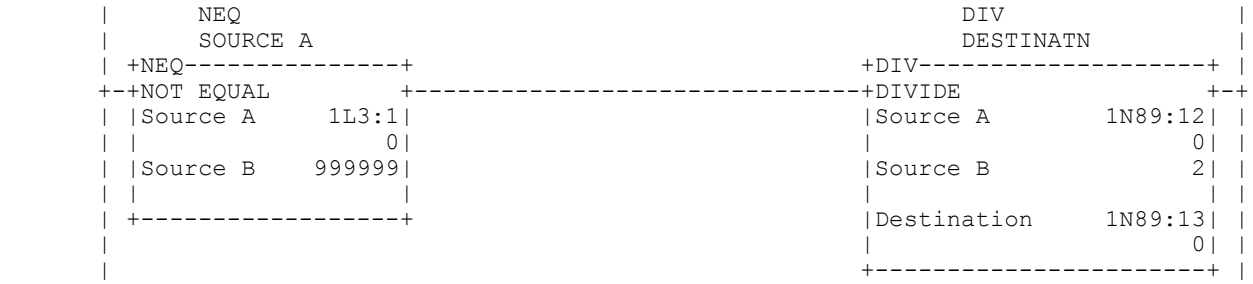
Rung 1STEP0:26



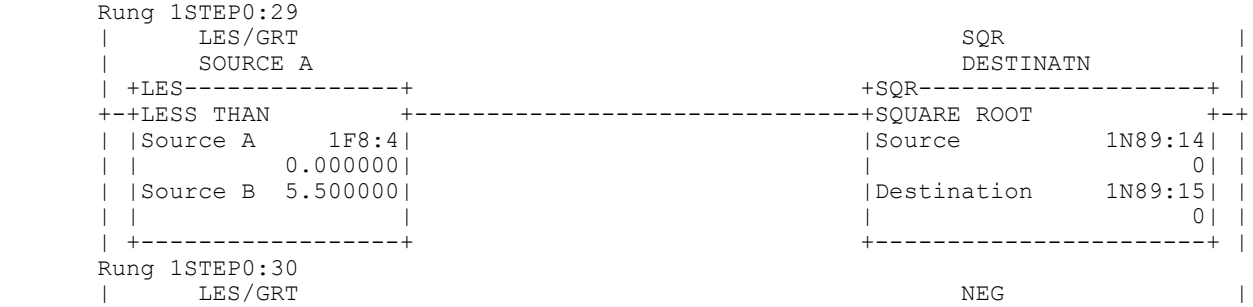
Rung 1STEP0:27



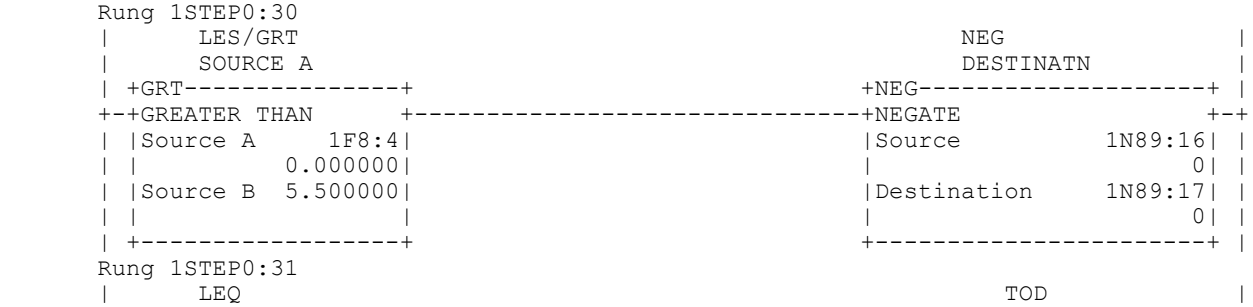
Rung 1STEP0:28



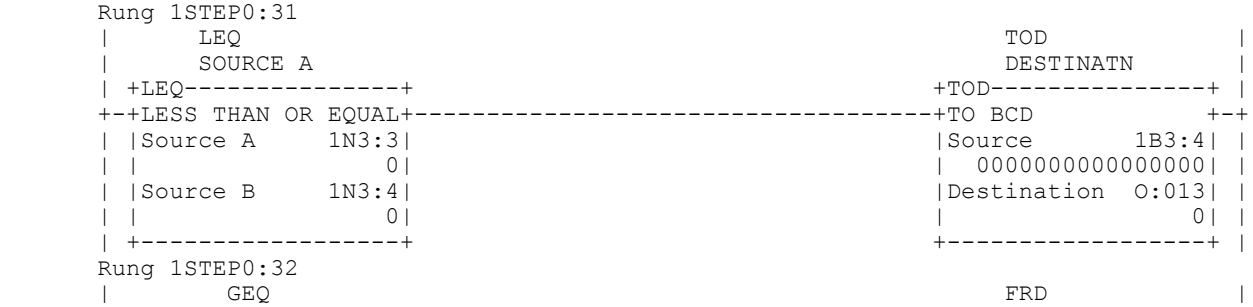
Rung 1STEP0:29



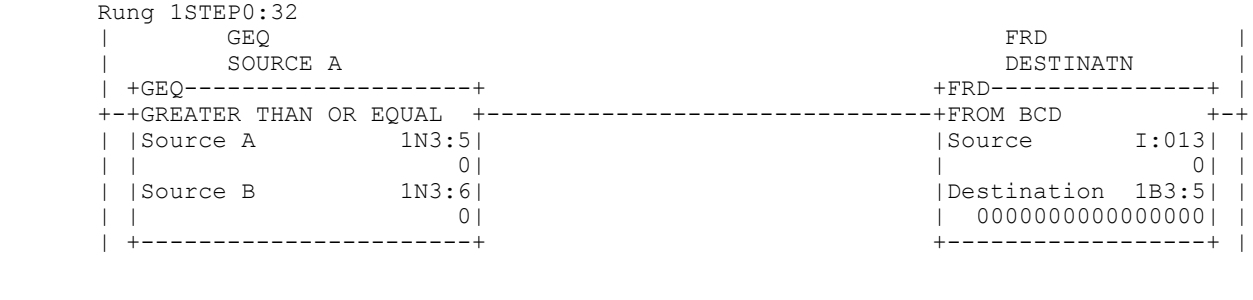
Rung 1STEP0:30



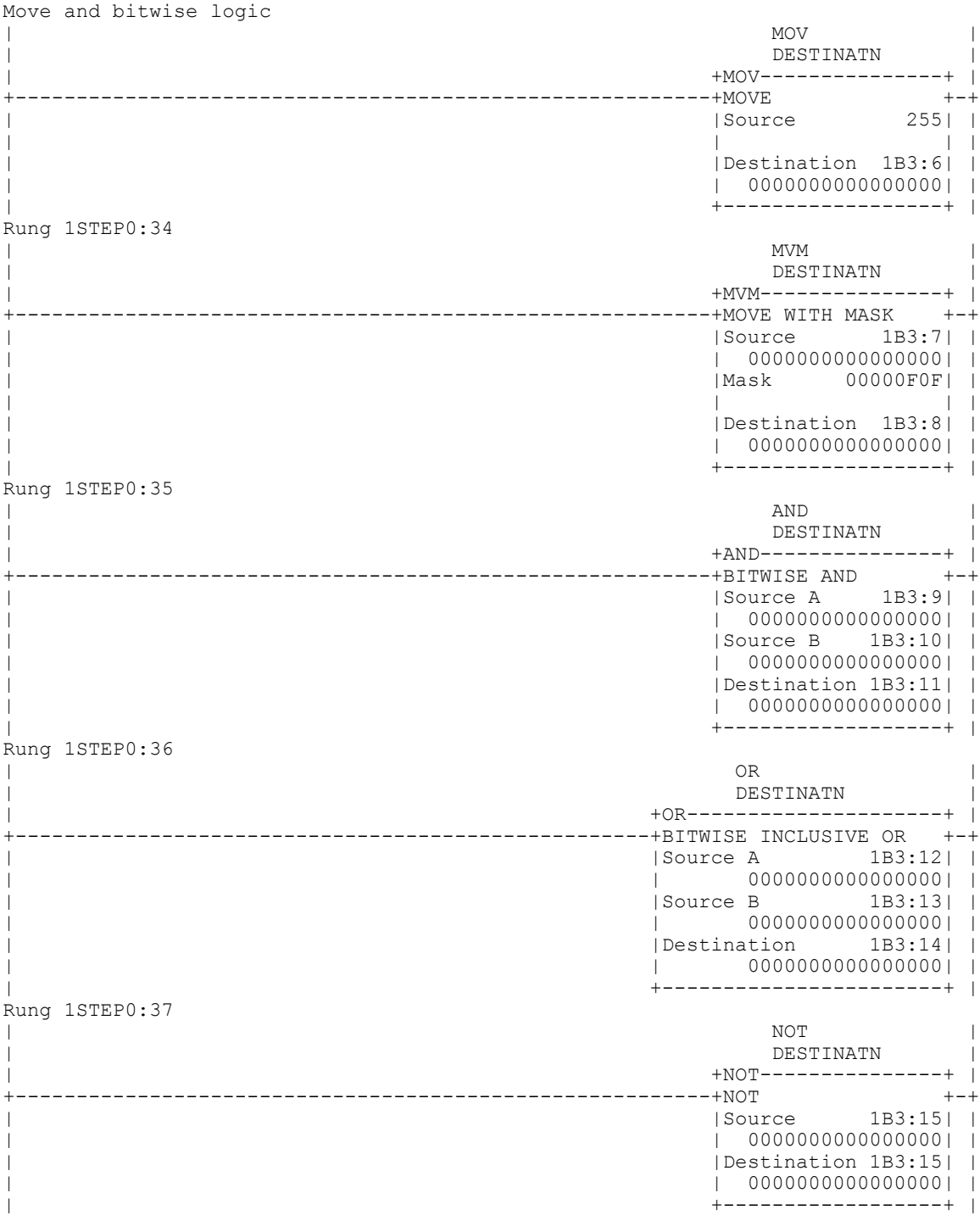
Rung 1STEP0:31



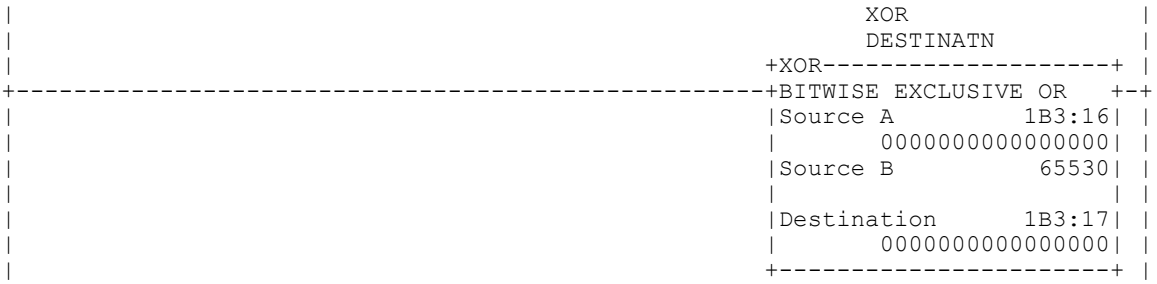
Rung 1STEP0:32



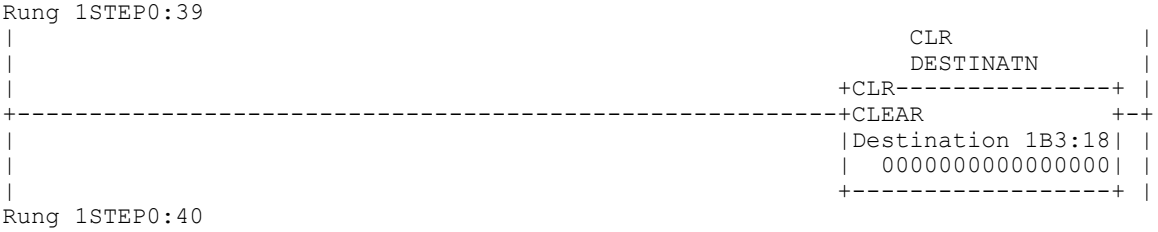
Rung 1STEP0:33
Move and bitwise logic



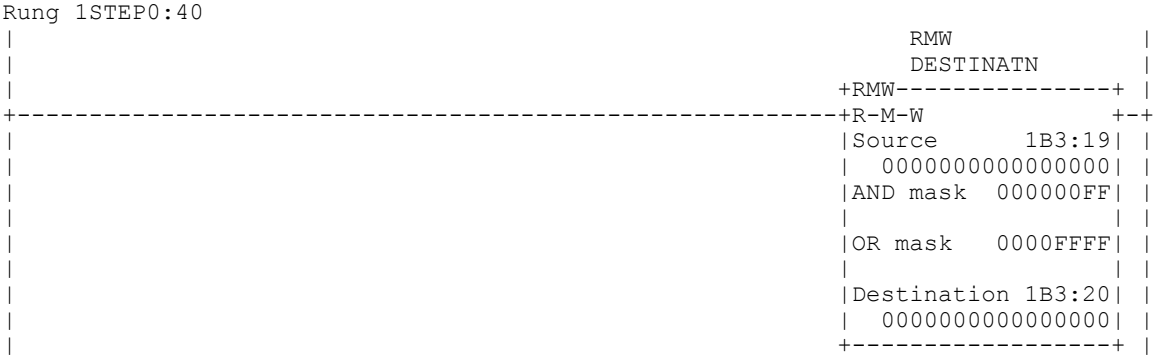
Rung 1STEP0:38



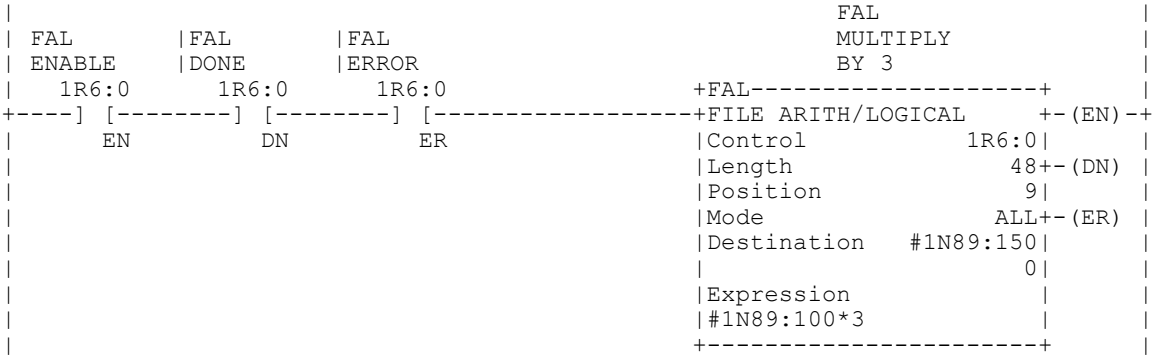
Rung 1STEP0:39



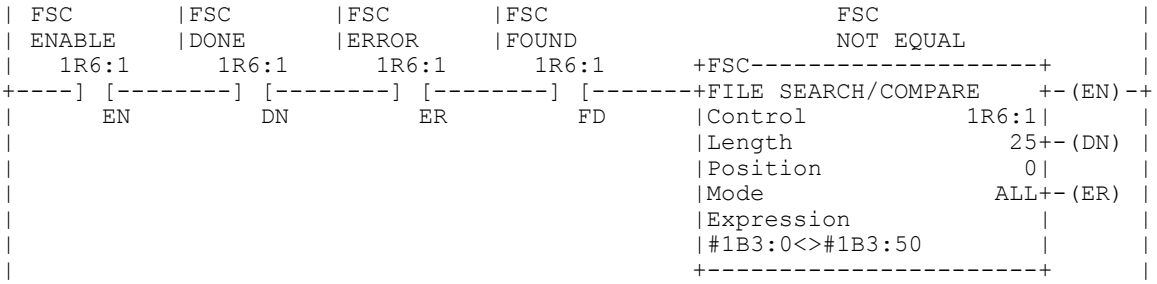
Rung 1STEP0:40



Rung 1STEP0:41
File instructions

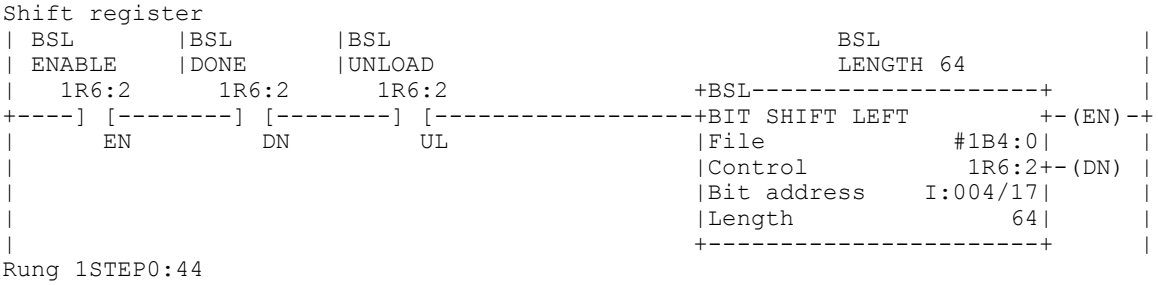


Rung 1STEP0:42

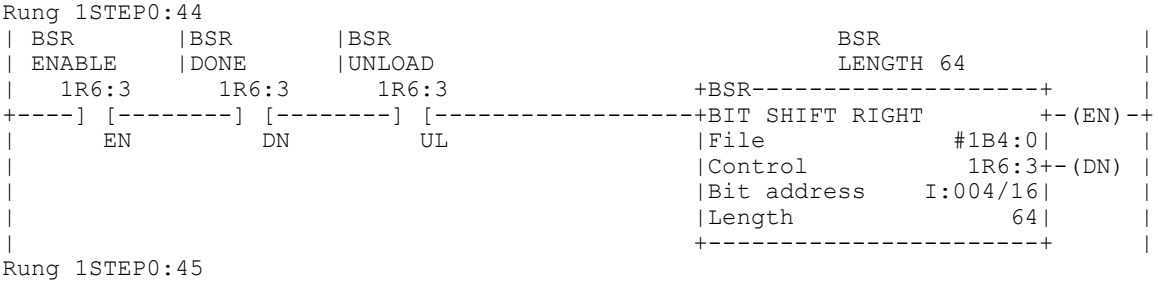


Rung 1STEP0:43

Shift register

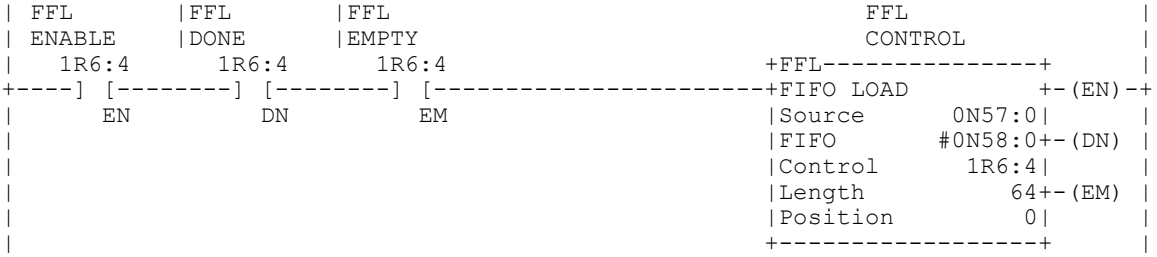


Rung 1STEP0:44

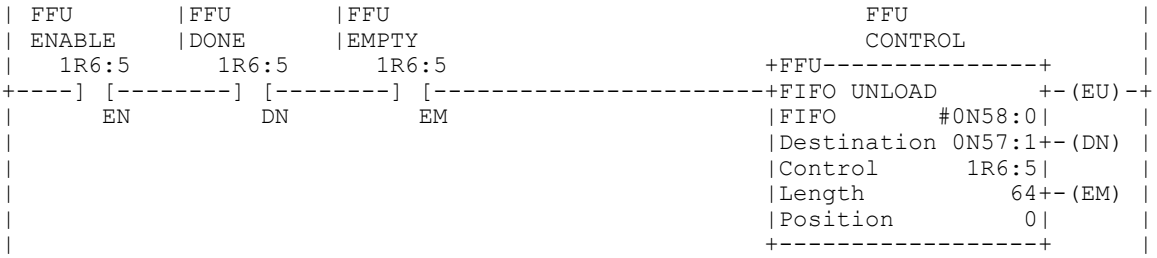


Rung 1STEP0:45

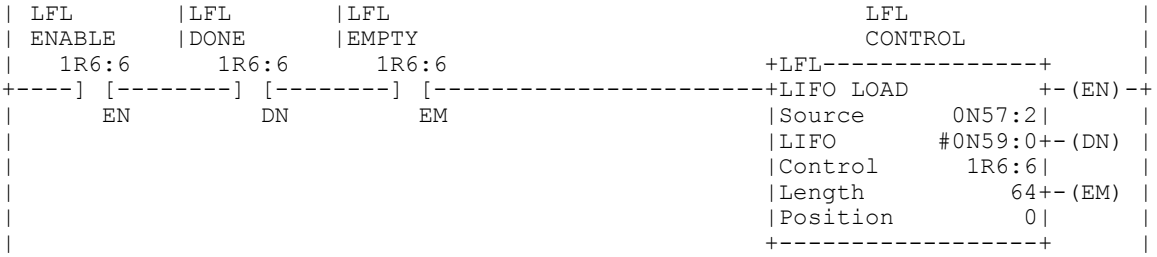
Rung 1STEP0:46



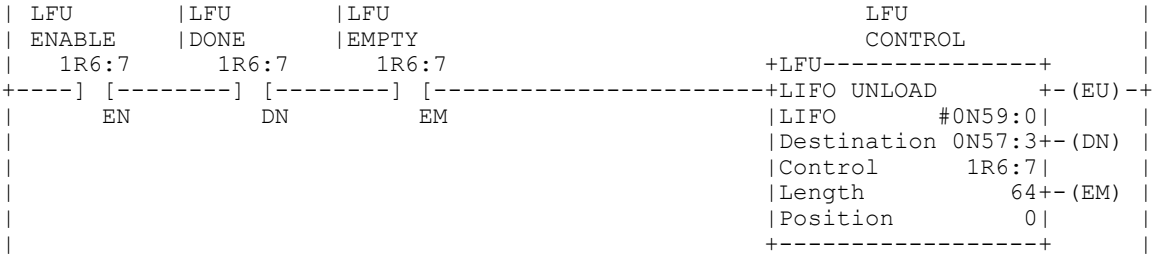
Rung 1STEP0:47



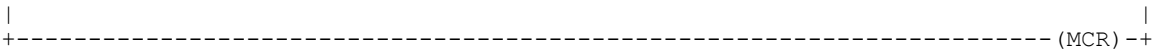
Rung 1STEP0:48



Rung 1STEP0:49

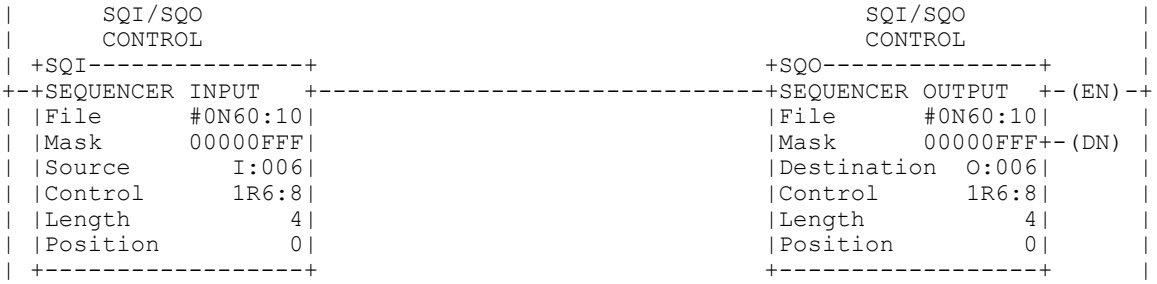


Rung 1STEP0:50

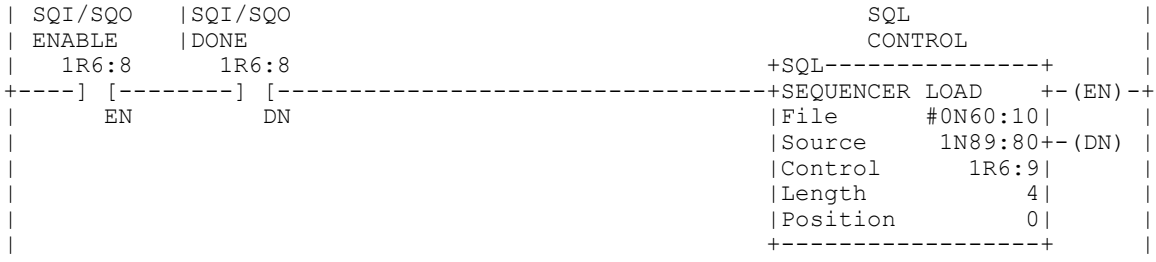


Rung 1STEP0:51

Sequencer

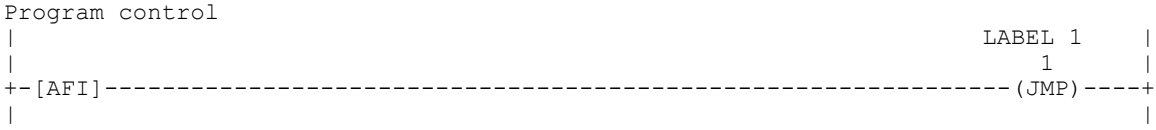


Rung 1STEP0:52

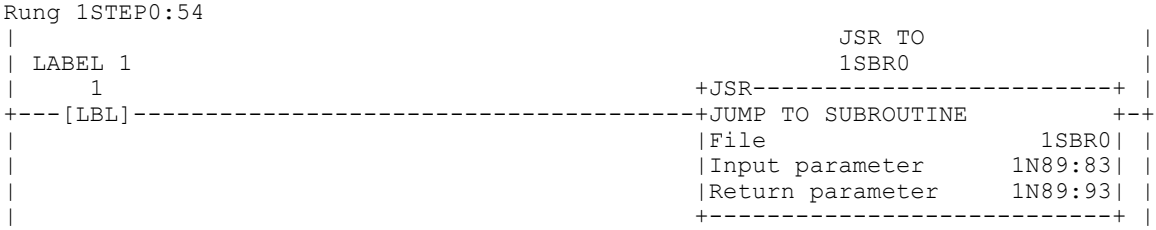


Rung 1STEP0:53

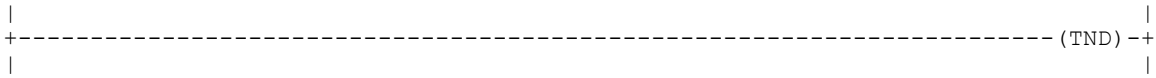
Program control



Rung 1STEP0:54



Rung 1STEP0:55

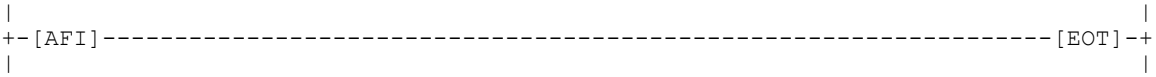


Rung 1STEP0:56

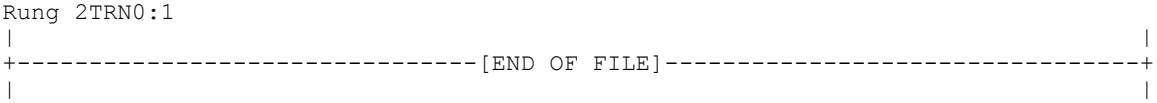
NO FILES IN SECTION 2IBP

NO FILES IN SECTION 2SBR

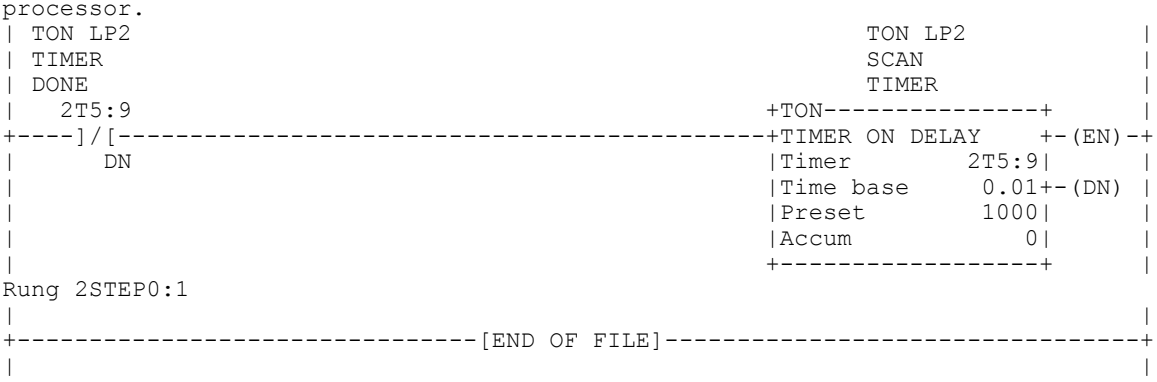
Rung 2TRN0:0



Rung 2TRN0:1



Rung 2STEP0:0
This step is in LP2. It shows what the addresses look like in a different processor.



Rung 2STEP0:1

[END OF FILE]