ChasingLEDs – code breakdown

```
R0 = LED pattern
R1 = LED speed
R2 = Console Switch and Display Register address
R3 = clock interrupt counter
R4 = temporary register


line clock interrupt vector
100    001500                      <- address of line clock interrupt handler
102    000340                      <- Processor Status Word, CPU to highest priority for interrupt

initialize
1000   012706   MOV #1000,SP        <- set stack location (stack decrements on push)
1002   001000
1004   012737   MOV #100,@#177546   <- enable line clock, by setting bit 6 to enable interrupt mode
1006   000100
1010   177546
1012   012701   MOV #5,R1           <- set default LED speed
1014   000005
1016   012702   MOV #177570,R2      <- set Console Switch and Display Register address
1020   177570
1022   000426   BR 1100             <- go to main program

main program
1100   011204   MOV (R2),R4         <- copy switches value to temporary register (for LED pattern setting)
1102   100412   BMI 1130            <- if it's a negative value (switch 15 set) do not set LED pattern
1104   020427   CMP R4,#0           <- check if switch value is zero (no switches set)
1106   000000
1110   001003   BNE 1120            <- if not so skip the next part
1112   012700   MOV #1,R0           <- set LED pattern to 1 (at least one LED must be lit)
1114   000001
1116   000404   BR 1130             <- LED pattern is set, go to LED show cycle
1120   010400   MOV R4,R0           <- set LED pattern from temporary register
1122   000402   BR 1130             <- LED pattern is set, go to LED show cycle
1124   004767   JSR PC,1300         <- jump to "show and wait" subroutine before shifting the LEDs left again
1126   000150
1130   006100   ROL R0              <- shift LED pattern to the left
1132   100374   BPL 1124            <- loop left shifts as long as leftmost LED not lit (negative bit is not set)
1134   004767   JSR PC,1300         <- jump to "show and wait" subroutine before shifting the LEDs right (again)
1136   000140
1140   006000   ROR R0              <- shift LED pattern to the right
1142   102374   BVC 1134            <- loop right shifts as long as overflow bit not set (shifts have gone past the first LED)
1144   000755   BR 1100             <- loop to the beginning of the main program to start the next cycle

"show and wait" subroutine
1300   011204   MOV (R2),R4         <- copy switches value to temporary register (for LED speed setting)
1302   010027   MOV R0,#177570      <- show LED pattern on the panel, by updating the Console Switch and Display Register
1304   020000
1306   005704   TST R4              <- set the condition codes of the temporary register value
1310   100014   BPL 1342            <- if it's a positive value (switch 15 not set) do not set LED speed
1312   010401   MOV R4,R1           <- copy temporary register to LED speed register for further processing
1314   012704   MOV #100000,R4      <- set bit clearing mask to temporary register
1316   100000
1320   040401   BIC R4,R1           <- clear negative bit from the LED speed register
1322   020127   CMP R1,#0           <- check if no switches (besides switch 15) are set
1324   000000
1326   001003   BNE 1336            <- if not so skip the next part
1330   012701   MOV #5,R1           <- set LED speed to the default value
1332   000005
1334   000402   BR 1342             <- LED speed is set, go to wait routine
1336   072127   ASH #-5,R1          <- shift LED speed value 5 positions to the right, to keep the wait delay reasonable
1340   177773
1342   005003   CLR R3              <- set the clock interrupt counter to 0
1344   020301   CMP R3,R1           <- wait the set number of interrupts by comparing the clock interrupt counter...
1346   002776   BLT 1344            <- ...and looping if not reached
1350   000207   RTS PC              <- end of subroutine, return to main program

line clock interrupt handler
1500   005203   INR R3              <- on every line clock interrupt increase the value of the counter
1502   000002   RTI                 <- end of interrupt handler
```