XDEF stepper  
 XREF hexkeypad, disp, display\_string , pushpress, Alternator  
 XREF port\_t, port\_p, port\_s, locknoise, unlocknoise  
 XREF LUT, dispF, gen1off, gen2off, gen3off, elevator, delayont, delaytimert  
 XREF delaytimer, delayon, delayons, delaytimers, speedup, switchchange, switchstatus  
 XREF sum, TON, pot\_value, real\_value, power\_output, read\_pot, gen1cap, ledvalue, delayonm, delaytimerm  
 XREF LEDTABLE, ledswitches, waithold, instepper, ledm1, gen2cap, gen3cap, wait, shutoff, shutdown  
 XREF gen1timer, gen2timer, gen3timer  
   
   
; The stepper is what unlocks, locks, and moves coal into the plant.  
; We use the RTI as a timer for each step so we can control how fast it goes  
; To make it speed up or slow down we simly set it at a value and incrament it when its down   
; and run it agin   
   
   
   
 stepper:  
 MOVB #1, instepper ; let other program functions know that it is in the stepper

; routine

waitforF: LDAA $450 ; memory address contains generator that was selected  
 CMPA #1 ; The next 6 lines check and branch to specific generator   
 ; routine

BEQ generator1  
 CMPA #2  
 BEQ generator2   
 CMPA #3  
 BEQ generator3  
   
   
generator1: LDAA port\_t ; generator can’t be filled if it is operating. These next 15   
 ; lines are checking if the generator needs to be turned off

; not

ANDA #$1  
 CMPA #$1  
 BEQ turnoffgen1  
 JMP opendoor  
   
generator2: LDAA port\_t  
 ANDA #$2  
 CMPA #$2  
 BEQ turnoffgen2   
 JMP opendoor2   
   
generator3: LDAA port\_t  
 ANDA #$4  
 CMPA #$4  
 BEQ turnoffgen3  
 JMP opendoor3   
   
   
   
   
turnoffgen1: ; The turnoffgen routines first let the rest of the program   
 ; know the generator is now off, checks if a shutdown is

; initiated, and then verifies the user flipped the right

; switch

LDAA gen1off  
 LDAA #$1  
 STAA gen1off  
 LDD #dispF  
 JSR display\_string  
 LDAA shutoff  
 CMPA #1  
 BNE skipshut1  
 JSR shutdown  
 MOVB #1, instepper  
skipshut1: LDAA port\_t  
 ANDA #$1  
 CMPA #$0  
 BNE turnoffgen1  
 JMP opendoor  
   
   
turnoffgen2:   
 LDAA gen2off  
 LDAA #$1  
 STAA gen2off  
 LDD #dispF  
 JSR display\_string  
 LDAA shutoff  
 CMPA #1  
 BNE skipshut2  
 JSR shutdown  
 MOVB #1, instepper  
skipshut2: LDAA port\_t  
 ANDA #$2  
 CMPA #$0  
 BNE turnoffgen2  
 JMP opendoor2   
   
  
 turnoffgen3:   
 LDAA gen3off  
 LDAA #$1  
 STAA gen3off  
 LDD #dispF  
 JSR display\_string  
 LDAA shutoff  
 CMPA #1  
 BNE skipshut3  
 JSR shutdown  
 MOVB #1, instepper  
skipshut3: LDAA port\_t  
 ANDA #$4  
 CMPA #$0  
 BNE turnoffgen3  
 JMP opendoor3

; File is broken up into 3 separate but very similar routines for each individual generator. They all

; begin with the opendoor process which actively checks if a switch has been changed and then

; ‘opens’ the generator doors a specific amount of time relative to the capacity of the generators.

; It then saves the value of the next led that needs to flash when the filling begins.

opendoor:   
 LDAA switchstatus  
 ANDA #$06  
 STAA switchstatus  
 LDX #LEDTABLE ;Open doors  
look: LDAA 1, x+  
 DECA ;Start of Part one  
 CMPA gen1cap  
 BNE look  
 INCA  
 STAA ledvalue  
 MOVB #1, waithold  
 MOVB ledvalue, ledswitches ; make leading LED flash as the filling proceeds  
 LDAA ledvalue  
 DECA  
 STAA ledm1   
 LDAA gen1cap  
 CMPA #$FF  
 BEQ FULL ; Don’t do any filling if generator is already filled  
 MOVB #1, unlocknoise ; make unlock music commence

movb #'W',disp+3  
 movb #'o',disp+4  
 movb #'r',disp+5  
 movb #'k',disp+6  
 movb #'i',disp+7  
 movb #'n',disp+8  
 movb #'g',disp+9  
 movb #'.',disp+10  
 movb #'.',disp+11   
 LDD #disp  
 JSR display\_string  
   
  
 LDX #LUT ; LUT used to move stepper motor

LDY #0  
  
again: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon ; delay specifies speed of stepper motor  
 MOVB #0, delaytimer  
   
stepdelay1: LDAA delaytimer  
 CMPA #2  
 BNE stepdelay1  
 MOVB #0, delayon ; turn delay off  
 CPX #LUT+4 ; verify LUT has been spanned  
 BNE Skip  
 INY  
 CPY #12  
 BEQ onerot ; keep track of how long motor has rotated  
 LDX #LUT  
Skip: JMP again   
   
   
 FULL: MOVB #0, switchchange  
 movb #'F',disp+3  
 movb #'U',disp+4  
 movb #'L',disp+5 ;If the generator is full  
 movb #'L',disp+6  
 movb #' ',disp+7  
 movb #' ',disp+8  
 movb #' ',disp+9  
 movb #' ',disp+10  
 movb #' ',disp+11   
 LDD #disp  
vroom: JSR display\_string  
 JSR hexkeypad  
 CMPA #$F  
 BNE vroom ; verify user understands generator is already full

JMP leave   
  
 onerot:   
 LDX #LUT+3 ; similar to previous sequence except motor needs to spin

; counterclockwise and move very fast

LDY #0  
again1: JSR updateton ; updates speed of DC motor if pot is adjusted   
 ; during the motor turning process

LDAA 1 ,x-  
 STAA port\_p  
 MOVB #1, delayons  
 MOVB #0, delaytimers  
stepdelay2: LDAA delaytimers  
 CMPA #150  
 BNE stepdelay2  
 MOVB #0, delayons  
 CPX #LUT-1   
 BNE Skip1  
 INY  
 CPY #3 ; only rotate a small amount  
 BEQ FILL  
 LDX #LUT+3  
Skip1: JMP again1   
  
  
FILL: MOVB #0, unlocknoise  
 MOVB #0, delayont  
 MOVB #0, delaytimert  
 MOVB #0, Alternator  
 MOVB #1, elevator ; want filling music to play  
 LDY #0   
 LDX #LUT ; motor moves clockwise   
 LDAA gen1cap ; display LED capacity  
 STAA port\_s  
 MOVB #9, speedup ; motor needs to gradually speed up  
again3: JSR updateton ; update DC motor speed  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay3: LDAA delaytimer  
 CMPA speedup  
 BNE stepdelay3  
 LDAB speedup ; speeding up  
 SUBB #1  
 STAB speedup  
 MOVB #0, delayon  
 CPX #LUT+4  
 BNE Skip3  
 INY  
 CPY #2 ; wait till motor reaches max speed  
 BEQ CON  
 LDX #LUT  
Skip3:   
 JMP again3  
  
SlowJmp: JMP Slow ;Jump to slow  
  
CON: LDAA gen1cap  
 CMPA #$7F ; if one dots are empty  
 BEQ SlowJmp  
   
 LDAA gen1cap  
 CMPA #$3F ; if two dots are empty  
 BEQ TDOT   
   
   
 LDY #0  
 LDX #LUT  
again4: LDAA gen1cap ; next 11 lines show generator filling with leds  
 LSLA  
 ADDA #1  
 STAA gen1cap  
 STAA port\_s  
 LDAA ledvalue  
 LSLA  
 STAA ledvalue  
 DECA  
 STAA ledm1  
 MOVB ledvalue, ledswitches  
 JSR updateton ; update DC motor speed  
redo: LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay4: LDAA delaytimer  
 CMPA #1  
 BNE stepdelay4  
 MOVB #0, delayon  
 CPX #LUT+4  
 BNE Skip4  
 LDX #LUT  
Skip4: INY  
 CPY #47  
 BLO redo  
 LDY #0  
 LDAA gen1cap ; check that generator isnt almost full  
 CMPA #$7F  
 BEQ Slow ; slow down filling process after all but one led is on

JMP again4  
  
TDOT: LSLA ; if only two leds need filled, generator should immediately  
 ; slow down after reaching top speed

INCA  
 STAA gen1cap ;Two dots need this to work  
 STAA port\_s  
 LDAA ledvalue  
 LSLA  
 STAA ledvalue  
 DECA  
 STAA ledm1  
 MOVB ledvalue, ledswitches  
 BRA Slow  
  
 Slow:  
 LDY #0  
 LDX #LUT  
 LDAB #1  
 MOVB #1, speedup ; motor at max speed  
again5: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay5: LDAA delaytimer  
 CMPA speedup  
 BNE stepdelay5  
 MOVB #0, delayon  
 LDAB speedup  
 INCB ; slowing motor down  
 STAB speedup  
 CPX #LUT+4  
 BNE Skip5  
 INY  
 CPY #3  
 BEQ LOCK ; once last led filled and motor stopped, initiate lock  
 LDX #LUT  
Skip5:  
 JMP again5  
   
   
   
 LOCK: MOVB #0, elevator  
 MOVB #0, delayont  
 MOVB #0, delaytimert  
 MOVB #0, ledswitches  
 MOVB #0, waithold  
 MOVB #0, wait  
 LDAA #$FF  
 STAA port\_s  
 LDAA gen1cap  
 LSLA  
 ADDA #1  
 STAA gen1cap ; let program know that generator capacity is full  
 STAA port\_s  
 movb #'I',disp  
 movb #'n',disp+1  
 movb #'i',disp+2  
 movb #'t',disp+3  
 movb #'i',disp+4  
 movb #'a',disp+5  
 movb #'t',disp+6  
 movb #'e',disp+7  
 movb #' ',disp+8  
 movb #'L',disp+9  
 movb #'o',disp+10  
 movb #'c',disp+11  
 movb #'k',disp+12  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28   
 LDD #disp  
 JSR display\_string  
waitpress: LDAA port\_p ; wait for push button to initiate lock  
 ANDA #$20  
 CMPA #0  
 BNE waitpress  
 MOVB #1, locknoise ; play locking music  
 MOVB #0, waithold  
 MOVB #0, ledswitches  
 LDX #LUT+3  
 LDY #0  
  
again6: JSR updateton  
 LDAA 1 ,x-  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
   
stepdelay6: LDAA delaytimer  
 CMPA #2  
 BNE stepdelay6  
 MOVB #0, delayon  
 CPX #LUT-1  
 BNE Skip6  
 INY  
 CPY #12  
 BEQ lastrot  
   
 LDX #LUT+3  
Skip6: JMP again6  
  
  
lastrot: LDX #LUT  
 LDY #0  
again7: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayons  
 MOVB #0, delaytimers  
stepdelay7: LDAA delaytimers  
 CMPA #150  
 BNE stepdelay7  
 MOVB #0, delayons  
 CPX #LUT+4  
 BNE Skip7  
 INY  
 CPY #3  
 BEQ leave  
 LDX #LUT  
Skip7: JMP again7   
   
done1jmp: JMP done1   
   
leave: MOVB #0, locknoise  
 MOVB #0, delayont  
 MOVB #0, delaytimert  
try1: LDAA port\_t  
 ANDA #$07  
 CMPA switchstatus ; verify switches are in state they were in before stepper   
 ; started

BEQ done1jmp  
 movb #' ',disp+16  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28   
 movb #'R',disp  
 movb #'e',disp+1  
 movb #'s',disp+2  
 movb #'e',disp+3  
 movb #'t',disp+4  
 movb #' ',disp+5  
 movb #'S',disp+6  
 movb #'w',disp+7  
 movb #'i',disp+8  
 movb #'t',disp+9  
 movb #'c',disp+10  
 movb #'h',disp+11  
 movb #'e',disp+12  
 movb #'s',disp+13  
 LDD #disp  
 JSR display\_string  
 JMP try1  
done1: movb #' ',disp  
 movb #' ',disp+1  
 movb #' ',disp+2  
 movb #' ',disp+3  
 movb #' ',disp+4  
 movb #' ',disp+5  
 movb #' ',disp+6  
 movb #' ',disp+7  
 movb #' ',disp+8  
 movb #' ',disp+9  
 movb #' ',disp+10  
 movb #' ',disp+11  
 movb #' ',disp+12  
 movb #' ',disp+13  
 movb #' ',disp+14  
 movb #' ',disp+16  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28  
 MOVB #0, switchchange  
 MOVB #0, instepper ; - End of part 1 -  
 MOVB #0, port\_s  
 MOVB #0, ledswitches  
 MOVB #0, waithold  
 MOVB #0, wait  
 MOVB #0, ledvalue  
 MOVB #0, ledm1  
 MOVB #0, gen1timer  
 MOVB #$FF, gen1cap  
 RTS

; opendoor2 and opendoor3 are same as opendoor for the second and third generators ; refer to generator 1 commenting for details   
   
opendoor2:   
 LDAA switchstatus  
 ANDA #$8D  
 STAA switchstatus  
 LDX #LEDTABLE ;Open doors  
look2: LDAA 1, x+   
 DECA ;Start of Part Two  
 CMPA gen2cap  
 BNE look2  
 INCA  
 STAA ledvalue  
 MOVB #1, waithold  
 MOVB ledvalue, ledswitches   
 LDAA ledvalue  
 DECA  
 STAA ledm1   
 LDAA gen2cap  
 CMPA #$FF  
 BEQ FULL2  
 MOVB #1, unlocknoise  
  
   
 movb #'W',disp+3  
 movb #'o',disp+4  
 movb #'r',disp+5  
 movb #'k',disp+6  
 movb #'i',disp+7  
 movb #'n',disp+8  
 movb #'g',disp+9  
 movb #'.',disp+10  
 movb #'.',disp+11   
 LDD #disp  
 JSR display\_string  
   
  
 LDX #LUT  
 LDY #0  
  
again2: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
   
stepdelay12: LDAA delaytimer  
 CMPA #2  
 BNE stepdelay12  
 MOVB #0, delayon  
 CPX #LUT+4  
 BNE Skip2  
 INY  
 CPY #12  
 BEQ onerot2  
 LDX #LUT  
Skip2: JMP again2   
   
   
 FULL2: MOVB #0, switchchange  
 movb #'F',disp+3  
 movb #'U',disp+4  
 movb #'L',disp+5 ;If the generator is full  
 movb #'L',disp+6  
 movb #' ',disp+7  
 movb #' ',disp+8  
 movb #' ',disp+9  
 movb #' ',disp+10  
 movb #' ',disp+11   
 LDD #disp  
vroom2: JSR display\_string  
 JSR hexkeypad  
 CMPA #$F  
 BNE vroom2  
 JMP leave   
 onerot2: LDX #LUT+3  
 LDY #0  
again12: JSR updateton  
 LDAA 1 ,x-  
 STAA port\_p  
 MOVB #1, delayons  
 MOVB #0, delaytimers  
stepdelay22:LDAA delaytimers  
 CMPA #150  
 BNE stepdelay22  
 MOVB #0, delayons  
 CPX #LUT-1  
 BNE Skip12  
 INY  
 CPY #3  
 BEQ FILL2  
 LDX #LUT+3  
Skip12: JMP again12   
  
  
FILL2: MOVB #0, unlocknoise  
 MOVB #0, delayont  
 MOVB #0, delaytimert  
 MOVB #0, Alternator  
 MOVB #1, elevator  
 LDY #0  
 LDX #LUT  
 LDAA gen2cap  
 STAA port\_s  
 MOVB #9, speedup  
again32: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay32: LDAA delaytimer  
 CMPA speedup  
 BNE stepdelay32  
 MOVB #0, delayon  
 LDAB speedup  
 SUBB #1  
 STAB speedup  
 CPX #LUT+4  
 BNE Skip32  
 INY  
 CPY #2  
 BEQ CON2  
 LDX #LUT  
Skip32:   
 JMP again32  
  
SlowJmp2: JMP Slow2 ;Jump to slow  
  
CON2: LDAA gen2cap  
 CMPA #$7F ; if one dots are empty  
 BEQ SlowJmp2  
   
 LDAA gen2cap  
 CMPA #$3F ; if two dots are empty  
 BEQ TDOT2  
   
   
 LDY #0  
 LDX #LUT  
again42: LDAA gen2cap  
 LSLA  
 ADDA #1  
 STAA gen2cap  
 STAA port\_s  
 LDAA ledvalue  
 LSLA  
 STAA ledvalue  
 DECA  
 STAA ledm1  
 MOVB ledvalue, ledswitches  
 JSR updateton  
redo2: LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay42: LDAA delaytimer  
 CMPA #1  
 BNE stepdelay42  
 MOVB #0, delayon  
 CPX #LUT+4  
 BNE Skip42  
 LDX #LUT  
Skip42: INY  
 CPY #47  
 BLO redo2  
 LDY #0  
 LDAA gen2cap  
 CMPA #$7F  
 BEQ Slow2  
 JMP again42  
  
TDOT2: LSLA  
 INCA  
 STAA gen2cap ;Two dots need this to work  
 STAA port\_s  
 LDAA ledvalue  
 LSLA  
 STAA ledvalue  
 DECA  
 STAA ledm1  
 MOVB ledvalue, ledswitches  
 BRA Slow2  
  
 Slow2:  
 LDY #0  
 LDX #LUT  
 LDAB #1  
 MOVB #1, speedup  
again52: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay52: LDAA delaytimer  
 CMPA speedup  
 BNE stepdelay52  
 MOVB #0, delayon  
 LDAB speedup  
 INCB  
 STAB speedup  
 CPX #LUT+4  
 BNE Skip52  
 INy  
 CPY #3  
 BEQ LOCK2  
 LDX #LUT  
Skip52:  
 JMP again52  
   
   
   
 LOCK2: MOVB #0, elevator  
 MOVB #0, delayonm  
 MOVB #0, delaytimerm  
 MOVB #0, ledswitches  
 MOVB #0, waithold  
 MOVB #0, wait   
 LDAA gen2cap  
 LSLA  
 ADDA #1  
 STAA gen2cap  
 STAA port\_s   
 MOVB #$FF, port\_s   
 movb #'I',disp  
 movb #'n',disp+1  
 movb #'i',disp+2  
 movb #'t',disp+3  
 movb #'i',disp+4  
 movb #'a',disp+5  
 movb #'t',disp+6  
 movb #'e',disp+7  
 movb #' ',disp+8  
 movb #'L',disp+9  
 movb #'o',disp+10  
 movb #'c',disp+11  
 movb #'k',disp+12  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28   
 LDD #disp  
 JSR display\_string  
waitpress2: LDAA port\_p  
 ANDA #$20  
 CMPA #0  
 BNE waitpress2  
 MOVB #1, locknoise  
 MOVB #0, waithold  
 MOVB #0, ledswitches  
 LDX #LUT+3  
 LDY #0  
  
again62: JSR updateton  
 LDAA 1 ,x-  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
   
stepdelay62: LDAA delaytimer  
 CMPA #2  
 BNE stepdelay62  
 MOVB #0, delayon  
 CPX #LUT-1  
 BNE Skip62  
 INY  
 CPY #12  
 BEQ lastrot2  
   
 LDX #LUT+3  
Skip62: JMP again62  
  
  
lastrot2:   
 LDX #LUT  
 LDY #0  
again72: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayons  
 MOVB #0, delaytimers  
stepdelay72: LDAA delaytimers  
 CMPA #150  
 BNE stepdelay72  
 MOVB #0, delayons  
 CPX #LUT+4  
 BNE Skip72  
 INY  
 CPY #3  
 BEQ leave2  
 LDX #LUT  
Skip72: JMP again72   
   
done2jmp: JMP done2   
   
leave2: MOVB #0, locknoise  
 MOVB #0, delayont  
 MOVB #0, delaytimert  
try2: LDAA port\_t  
 ANDA #7  
 CMPA switchstatus  
 BEQ done2jmp  
 movb #' ',disp+16  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28   
 movb #'R',disp  
 movb #'e',disp+1  
 movb #'s',disp+2  
 movb #'e',disp+3  
 movb #'t',disp+4  
 movb #' ',disp+5  
 movb #'S',disp+6  
 movb #'w',disp+7  
 movb #'i',disp+8  
 movb #'t',disp+9  
 movb #'c',disp+10  
 movb #'h',disp+11  
 movb #'e',disp+12  
 movb #'s',disp+13  
 LDD #disp  
 JSR display\_string  
 JMP try2  
done2: movb #' ',disp  
 movb #' ',disp+1  
 movb #' ',disp+2  
 movb #' ',disp+3  
 movb #' ',disp+4  
 movb #' ',disp+5  
 movb #' ',disp+6  
 movb #' ',disp+7  
 movb #' ',disp+8  
 movb #' ',disp+9  
 movb #' ',disp+10  
 movb #' ',disp+11  
 movb #' ',disp+12  
 movb #' ',disp+13  
 movb #' ',disp+14  
 movb #' ',disp+16  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28  
 MOVB #0, switchchange  
 MOVB #0, instepper  
 MOVB #0, port\_s  
 MOVB #0, ledswitches  
 MOVB #0, waithold  
 MOVB #0, wait ;End of part 2  
 MOVB #0, ledvalue  
 MOVB #0, ledm1  
 MOVB #0, gen2timer  
 MOVB #$FF, gen2cap  
 RTS   
   
   
opendoor3:   
 LDAA switchstatus  
 ANDA #$8B  
 STAA switchstatus   
 LDX #LEDTABLE ;Open doors  
look3: LDAA 1, x+   
 DECA ;Start of Part 3  
 CMPA gen3cap  
 BNE look3  
 INCA  
 STAA ledvalue  
 MOVB #1, waithold  
 MOVB ledvalue, ledswitches  
 LDAA ledvalue  
 DECA  
 STAA ledm1   
 LDAA gen3cap  
 CMPA #$FF  
 BEQ FULL3  
 MOVB #1, unlocknoise  
   
 movb #'W',disp+3  
 movb #'o',disp+4  
 movb #'r',disp+5  
 movb #'k',disp+6  
 movb #'i',disp+7  
 movb #'n',disp+8  
 movb #'g',disp+9  
 movb #'.',disp+10  
 movb #'.',disp+11   
 LDD #disp  
 JSR display\_string  
   
  
 LDX #LUT  
 LDY #0  
  
again33: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
   
stepdelay13: LDAA delaytimer  
 CMPA #2  
 BNE stepdelay13  
 MOVB #0, delayon  
 CPX #LUT+4  
 BNE Skip33  
 INY  
 CPY #12  
 BEQ onerot3  
 LDX #LUT  
Skip33: JMP again33   
   
   
 FULL3: MOVB #0, switchchange  
 movb #'F',disp+3  
 movb #'U',disp+4  
 movb #'L',disp+5 ;If the generator is full  
 movb #'L',disp+6  
 movb #' ',disp+7  
 movb #' ',disp+8  
 movb #' ',disp+9  
 movb #' ',disp+10  
 movb #' ',disp+11   
 LDD #disp  
vroom3: JSR display\_string  
 JSR hexkeypad  
 CMPA #$F  
 BNE vroom3  
 JMP leave3   
  
 onerot3:   
 LDX #LUT+3  
 LDY #0  
again13: JSR updateton  
 LDAA 1 ,x-  
 STAA port\_p  
 MOVB #1, delayons  
 MOVB #0, delaytimers  
stepdelay23: LDAA delaytimers  
 CMPA #150  
 BNE stepdelay23  
 MOVB #0, delayons  
 CPX #LUT-1  
 BNE Skip13  
 INY  
 CPY #3  
 BEQ FILL3  
 LDX #LUT+3  
Skip13: JMP again13   
  
  
FILL3: MOVB #0, unlocknoise  
 MOVB #0, delayont  
 MOVB #0, delaytimert  
 MOVB #0, Alternator  
 MOVB #1, elevator  
 LDY #0  
 LDX #LUT  
 LDAA gen3cap  
 STAA port\_s  
 MOVB #9, speedup  
again34: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay33: LDAA delaytimer  
 CMPA speedup  
 BNE stepdelay33  
 MOVB #0, delayon  
 LDAB speedup  
 SUBB #1  
 STAB speedup  
 CPX #LUT+4  
 BNE Skip34  
 INY  
 CPY #2  
 BEQ CON3  
 LDX #LUT  
Skip34:   
 JMP again34  
  
SlowJmp3: JMP Slow3 ;Jump to slow  
  
CON3: LDAA gen3cap  
 CMPA #$7F ; if one dots are empty  
 BEQ SlowJmp3  
   
 LDAA gen3cap  
 CMPA #$3F ; if two dots are empty  
 BEQ TDOT3  
   
   
 LDY #0  
 LDX #LUT  
again43: LDAA gen3cap  
 LSLA  
 ADDA #1  
 STAA gen3cap  
 STAA port\_s  
 LDAA ledvalue  
 LSLA  
 STAA ledvalue  
 DECA  
 STAA ledm1  
 MOVB ledvalue, ledswitches  
 JSR updateton  
redo3: LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay43: LDAA delaytimer  
 CMPA #1  
 BNE stepdelay43  
 MOVB #0, delayon  
 CPX #LUT+4  
 BNE Skip43  
 LDX #LUT  
Skip43: INY  
 CPY #47  
 BLO redo3  
 LDY #0  
 LDAA gen3cap  
 CMPA #$7F  
 BEQ Slow3  
 JMP again43  
  
TDOT3: LSLA  
 INCA  
 STAA gen3cap ;Two dots need this to work  
 STAA port\_s  
 LDAA ledvalue  
 LSLA  
 STAA ledvalue  
 DECA  
 STAA ledm1  
 MOVB ledvalue, ledswitches  
 BRA Slow3  
  
 Slow3:  
 LDY #0  
 LDX #LUT  
 LDAB #1  
 MOVB #1, speedup  
again53: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
stepdelay53: LDAA delaytimer  
 CMPA speedup  
 BNE stepdelay53  
 MOVB #0, delayon  
 LDAB speedup  
 INCB  
 STAB speedup  
 CPX #LUT+4  
 BNE Skip53  
 INY  
 CPY #3  
 BEQ LOCK3  
 LDX #LUT  
Skip53:  
 JMP again53  
   
   
   
 LOCK3: MOVB #0, elevator  
 MOVB #0, delayont  
 MOVB #0, delaytimert  
 MOVB #0, ledswitches  
 MOVB #0, waithold  
 MOVB #0, wait   
 LDAA gen3cap  
 LSLA  
 ADDA #1  
 STAA gen3cap  
 STAA port\_s  
 MOVB #$FF, port\_s  
 movb #'I',disp  
 movb #'n',disp+1  
 movb #'i',disp+2  
 movb #'t',disp+3  
 movb #'i',disp+4  
 movb #'a',disp+5  
 movb #'t',disp+6  
 movb #'e',disp+7  
 movb #' ',disp+8  
 movb #'L',disp+9  
 movb #'o',disp+10  
 movb #'c',disp+11  
 movb #'k',disp+12  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28   
 LDD #disp  
 JSR display\_string  
waitpress3: LDAA port\_p  
 ANDA #$20  
 CMPA #0  
 BNE waitpress3  
   
 MOVB #1, locknoise  
 MOVB #0, waithold  
 MOVB #0, ledswitches  
 LDX #LUT+3  
 LDY #0  
  
again63: JSR updateton  
 LDAA 1 ,x-  
 STAA port\_p  
 MOVB #1, delayon  
 MOVB #0, delaytimer  
   
stepdelay63: LDAA delaytimer  
 CMPA #2  
 BNE stepdelay63  
 MOVB #0, delayon  
 CPX #LUT-1  
 BNE Skip63  
 INY  
 CPY #12  
 BEQ lastrot3  
   
 LDX #LUT+3  
Skip63: JMP again63  
  
  
lastrot3: LDX #LUT  
 LDY #0  
again73: JSR updateton  
 LDAA 1 ,x+  
 STAA port\_p  
 MOVB #1, delayons  
 MOVB #0, delaytimers  
stepdelay73: LDAA delaytimers  
 CMPA #150  
 BNE stepdelay73  
 MOVB #0, delayons  
 CPX #LUT+4  
 BNE Skip73  
 INY  
 CPY #3  
 BEQ leave3  
 LDX #LUT  
Skip73: JMP again73   
   
done3jmp: JMP done3   
   
leave3: MOVB #0, locknoise   
 MOVB #0, delayont  
 MOVB #0, delaytimert  
try3: LDAA port\_t  
 ANDA #7  
 CMPA switchstatus  
 BEQ done3jmp  
 movb #' ',disp+16  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28   
 movb #'R',disp  
 movb #'e',disp+1  
 movb #'s',disp+2  
 movb #'e',disp+3  
 movb #'t',disp+4  
 movb #' ',disp+5  
 movb #'S',disp+6  
 movb #'w',disp+7  
 movb #'i',disp+8  
 movb #'t',disp+9  
 movb #'c',disp+10  
 movb #'h',disp+11  
 movb #'e',disp+12  
 movb #'s',disp+13  
 LDD #disp  
 JSR display\_string  
 JMP try3  
done3: movb #' ',disp  
 movb #' ',disp+1  
 movb #' ',disp+2  
 movb #' ',disp+3  
 movb #' ',disp+4  
 movb #' ',disp+5  
 movb #' ',disp+6  
 movb #' ',disp+7  
 movb #' ',disp+8  
 movb #' ',disp+9  
 movb #' ',disp+10  
 movb #' ',disp+11  
 movb #' ',disp+12  
 movb #' ',disp+13  
 movb #' ',disp+14  
 movb #' ',disp+16  
 movb #' ',disp+17  
 movb #' ',disp+18  
 movb #' ',disp+19  
 movb #' ',disp+20  
 movb #' ',disp+21  
 movb #' ',disp+22  
 movb #' ',disp+23  
 movb #' ',disp+24  
 movb #' ',disp+25  
 movb #' ',disp+26  
 movb #' ',disp+27  
 movb #' ',disp+28  
 MOVB #0, switchchange  
 MOVB #0, instepper ; - End of part 3 -  
 MOVB #0, port\_s  
 MOVB #0, ledswitches  
 MOVB #0, waithold  
 MOVB #0, wait  
 MOVB #0, ledvalue  
 MOVB #0, ledm1  
 MOVB #0, gen3timer  
 MOVB #$FF, gen3cap  
 RTS   
   
   
   
   
   
 Updateton: ; verifies DC motor is still responding to user input  
   
 PSHD  
 PSHX  
 PSHY  
 JSR read\_pot ; same process from hex keypad routine  
 LDD pot\_value  
 STD power\_output  
 LDX #3  
 IDIV  
 TFR x, b  
 STAB real\_value  
 LDAA gen1off  
 CMPA #$1  
 BEQ skip10  
 LDAA sum  
 ADDA real\_value  
 STAA sum   
skip10: LDAA gen2off  
 CMPA #$1  
 BEQ skip20  
 LDAA sum  
 ADDA real\_value  
 STAA sum  
skip20: LDAA gen3off  
 CMPA #$1  
 BEQ skip30  
 LDAA sum  
 ADDA real\_value  
 STAA sum   
skip30: LDAA sum  
 STAA TON  
 MOVB #0, sum  
 LDAA shutoff ; check if a shutdown has been initiated

CMPA #1  
 BNE endrts  
 JSR shutdown  
 MOVB #1, instepper  
 MOVB #1, waithold  
 LDD #disp  
 JSR display\_string  
endrts PULY  
 PULX  
 PULD  
 RTS