PROGRAMMABLE ALARM CLOCK

Having developed a system sensitive to intrusive footsteps, let's now consider a project to turn your micro into a handy programmable alarm clock. Such a system can, of course, be tailored to meet one's exact needs. The program we give (in versions for the Commodore 64 and the BBC Micro) allows the user to enter:

1) the time of day;

2) the number of 'snooze' intervals (periods between the bursts of the buzzer or music) required;

3) for each snooze interval, whether music, alarm or a silent period is required and the interval length;

4) for each interval, whether a light should be switched on;

5) the latest desirable rising time.

The program works on the assumption that the following connections are made to the low voltage output box:

1) A tape recorder is connected to line 0 through a mains relay.

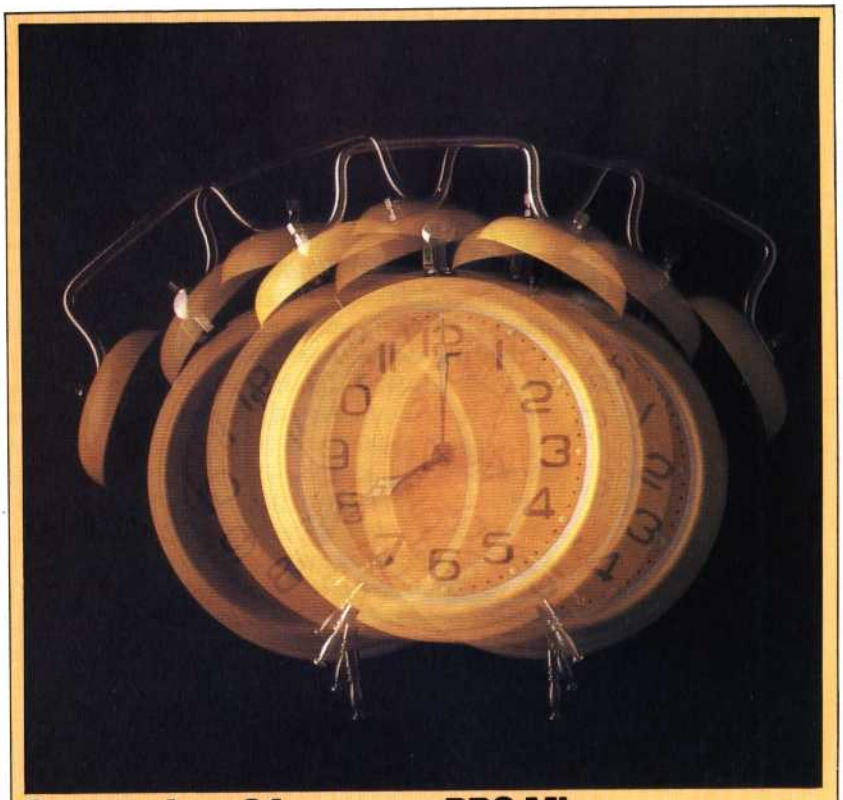
2) A table lamp is connected to line 1 through a mains relay to line 1.

3) A nine-volt electric bell is connected directly to line 3.

The program accepts the latest rising time and works backwards through the programmed intervals to calculate the start time for each interval. Use is made of arrays to store the data that tell us which appliances are to be on during any one period. Note that the array variables are given values that correspond to the bit value required in the data register to turn that particular appliance on. By making use of the logical OR instruction, we can simply find the composite total that must be placed in the data register to turn any combination of the devices on.

Most of our programming effort has been directed towards manipulating string variables to allow numerical calculations to be made. This is particularly true of the Commodore 64 program, as the version of BASIC used by that machine lacks the useful MOD and DIV commands available to programmers of the BBC Micro.

We have now developed a truly flexible input and output system for microcomputer control that allows us to control LEDs, low voltage devices and mains appliances, as well as allowing the micro to accept and interpret data input from a range of sensors.There are many possibilities now open to us to design control systems for our own use. In the examples given here, the micro is used as a sophisticated programmable timer. Other applications could involve turning electric fires on and off in response to a pair of heat sensors, or turning on an electric light at night. There are endless possibilities for experimentation.



IAN McKINNELL

Commodore 64

```
100 REM **** CBM 64 ALARM CLOCK ****
110 DDR=56579:DATREG=56577
120 POKE DDR,255:POKEDATREG,0
130 PRINTCHR$(147):REM CLEAR SCREEN
140 INPUT"NUMBER OF SNOOZE INTERVALS";N
150 M=N+1
160 DIM A(M),M(M),L(M),TS$(M),T(M)
170 :
180 REM **** INPUT INTERVAL DATA ****
190 FOR C=1 TO N
200 PRINT:PRINT"INTERVAL NUMBER";C
210 INPUT"MUSIC,ALARM OR SILENCE (M/A/S)";AN$
215 AN$=LEFT$(AN$,1)
220 IF AN$<>"M"ANDAN$<>"A"ANDAN$<>"S"THEN 210
230 IF AN$="M" THEN M(C)=1:A(C)=0
240 IF AN$="A" THEN A(C)=0:M(C)=0
250 IF AN$="S" THEN A(C)=0:M(C)=0
260 INPUT"LIGHT ON (Y/N)";AL$
270 L$=LEFT$(AL$,1)
280 IF L$<>"Y" AND L$<>"N" THEN 260
290 IF L$="Y" THEN L(C)=2:GOTO310
300 L(C)=0
310 INPUT"TIME INTERVAL (MINS)";T(C)
320 NEXT C
330 :
340 INPUT"LATEST RISING TIME (HHMM)";LT$
350 LT$=LT$+"00":REM ADD SECONDS
360 TS$(N+1)=LT$:REM LAST TIME
370 REM CONVERT LATEST TIME TO MINUTES
380 LM=60*VAL(LEFT$(LT$,2))+VAL(MID$(LT$,3,2))
390 :
400 INPUT"TIME NOW (HHMM)";TN$
410 TI$=TN$+"00":REM START TIMER
420 :
430 REM **** ANALYSE AND CALCULATE ****
440 REM ** CALC INTERVAL START TIMES **
450 FOR C=N TO 1 STEP -1
460 LM=LM-T(C):REM START TIME IN MINS
470 HR=INT(LM/60)
480 MN=INT(60*(LM/60-HR+.000001))
490 HR$=STR$(HR):REM HOURS
500 MN$=STR$(MN):REM MINS
510 MN$=MID$(MN$,2,LEN(MN$))
520 HR$=MID$(HR$,2,LEN(HR$))
530 REM ** ADD LEADING ZEROS **
540 SP$="00"
550 HR$=LEFT$(SP$,2-LEN(HR$))+HR$
560 MN$=LEFT$(SP$,2-LEN(MN$))+MN$
570 TS$(C)=HR$+MN$+"00"
580 NEXT C
590 :
600 REM **** GO ****
610 PRINTCHR$(147)
620 FOR C=1 TO N+1
630 IF TI$<TS$(C)THENGOSUB710:GOTO630
640 DN=M(C) OR A(C) OR L(C):REM DATREG DATA
650 POKE DATREG,DN
670 NEXT C
680 POKE DATREG,0
690 END
700 :
710 REM **** DISPLAY TIMER S/R ****
720 PRINTCHR$(145);:REM CRSR UP
730 PRINTLEFT$(TI$,2)":"MID$(TI$,3,2);
740 PRINT" "RIGHT$(TI$,2)
750 RETURN
```

BBC Micro

```
10 REM BBC ALARM CLOCK
15 MODE7
20 DDR=&FE62:DATREG=&FE60
30 CLS
40 INPUT"NUMBER OF SNOOZE INTERVALS";N
45 M=N+1
50 DIM A(M),M(M),L(M),T(M),TS(M)
70 REM **** INPUT INTERVAL DATA ****
80 FORC=1 TO N
90   PRINT"INTERVAL NUMBER";C
96   REPEAT
100     PRINT "MUSIC,ALARM OR SILENCE";
103     INPUT " (M/A/S)";AN$
105    AN$=LEFT$(AN$,1)
110    UNTIL AN$="M"ORAN$="A"ORAN$="S"
120  IF AN$="M"THEN M(C)=1:A(C)=0
130  IF AN$="A"THEN M(C)=0:A(C)=0
140  IF AN$="S"THEN M(C)=0:A(C)=0
150  REPEAT
160    INPUT"LIGHT ON (Y/N)";AL$
170    AL$=LEFT$(AL$,1)
180    UNTIL AL$="Y" OR AL$="N"
190  IF AL$="Y" THEN L(C)=2 ELSE L(C)=0
200  INPUT"TIME INTERVAL (MINS)";T(C)
210  NEXT C
220 :
230 INPUT"LATEST RISING TIME (HHMM)";LT$
232  TS(N+1)=6000*(60*VAL(LEFT$(LT$,2)))
234   TS(N+1)=TS(N+1)+VAL(RIGHT$(LT$,2))
236 REM CONVERT LATEST TIME TO MINS
237 LM=60*VAL(LEFT$(LT$,2))
239  LM=LM+VAL(RIGHT$(LT$,2))
240 INPUT"TIME NOW (HHMM)";TN$
250  TIME=6000*(60*VAL(LEFT$(TN$,2)))
255   TIME=TIME+VAL(RIGHT$(TN$,2))
260 :
270 REM ANALYSE AND CALCULATE
280 FORC=N TO 1 STEP -1
290   LM=LM-T(C):REM INTERVAL START
300   TS(C)=6000*LM
310    NEXT C
320 :
330  REM **** GO ****
340 CLS
350 FOR C=1 TO N+1
360   REPEAT
370     PROCtimer
380     UNTIL TIME>=TS(C)
390   REGDATA=M(C) OR A(C) OR L(C)
400   ?DATREG=REGDATA
420   NEXT C
430 ?DATREG=0
440 END
999 :
1000 DEF PROCtimer
1020 MIN=(TIME DIV 6000) MOD 60
1030  HR=(TIME DIV 6000) MOD 60
1040 MIN$=STR$(MIN):HR$=STR$(HR)
1042 REM ADD LEADING ZEROS
1043 SP$="00"
1044 HR$=LEFT$(SP$,2-LEN(HR$))+HR$
1045 MIN$=LEFT$(SP$,2-LEN(MIN$))+MIN$
1050 PRINTTAB(18,12)HR$;":";MIN$
1060 ENDPROC
```