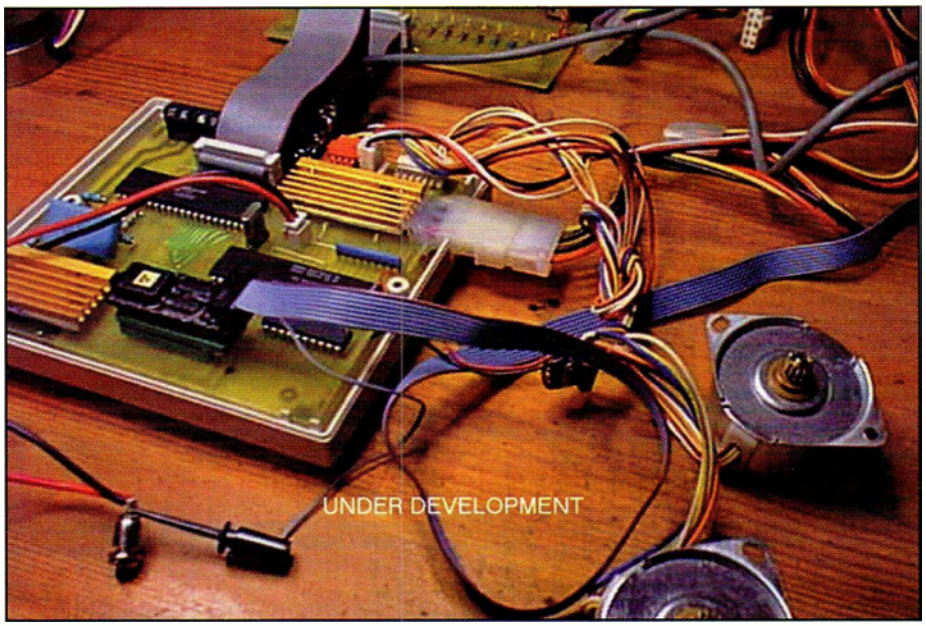
RS232 IO Control Board

PART 2

by R.Grodzik



As with any project of any complexity, 'bugs' both hardware and software do arise, so it is not surprising that a few have been found. These are:

A 10μF capacitor should be connected across the reset switch to ensure reset of the 8031 processor on power on (+ve side to the Vcc). This is due to the behaviour of my PC and the subsequent powering up of the external PSU that supplies the power to the RS232IO board. i.e. during power on, the PC's plug and play software switches on the COM port to check if an external device is connected - so the PSU switches on briefly. If the hardware handshake lines on the COM port are connected together, plug and play assigns the com port exclusively to another device and so preventing other devices operating. So ensure that the handshake line - DTR/DSR and CTS/RTS are open and not strapped together.

The AC on/off circuit comprising of IC8 (MOC3041) and the triac (IC9),worked fine upto a 100 watt load - my hairdryer-motor only with the heating element switched off. When the load was increased by switching on the heat it was found necessary to reduce the value of R6 (the LED voltage dropping resistor) to 220R. Always use a load to test this circuit, otherwise it will not work. And remember that the small heatsink was designed for small lighting/AC. motor circuits so don't go connecting it to your washer/dryer!

The limiting resistor R3 which feeds alarm switch transistor BS170 may be omitted, and the RTS line (pin 7) of the D-type connector strapped directly to the transistors gate with a small wire link.

As to the software bugs, they were too numerous to mention since the PC software driver (written in assembler) and the RS232IO board control software (written in assembler) were of some complexity. I am now happy to report that all of them have been ironed out.

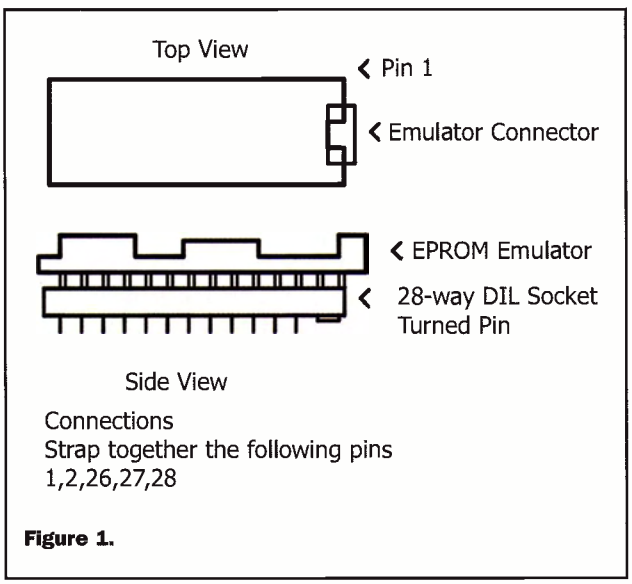


Figure 1.

LED Monitoring and interface board.

This board consists of a series of low current LED's connected directly to the 8155 PIO lines, monitoring Port A and Port B. After consulting the chip's data sheet it was found that each line could sink/source 2.5mA which is sufficient to light a low current led without the need for a cmos/ttl driver. This board also has additional monitoring functions:

- A green led power indicator connected directly to the 5-volt supply via a limiting resistor.
- A red led alarm indicator connected via the ribbon cable and plug3 to the PC0 line of the 8155.This is activated and flashes whenever the alarm circuit comprising of a microswitch (N.C.), the INT0 interrupt pin of the 8031 and a 12K resistor is broken - switch operates. Normally the INT0 line (pin 12) is held at ground by R5. When the microswitch operates, a logic level transition from low to high causes the 8031 to interrupt, and service an interrupt service routine i.e. sends an alarm code back to the PC and also flashes the red led on the monitoring board.

The interface board is also provided with a 5 volt power supply and a prototyping area allowing relay circuits etc. to be driven by the 16 output lines via transistor buffers or open collector circuits.

RS232IO Software

This is of considerable size, filling the 2048 byte capacity of the 2716 EPROM with a couple of bytes to spare, so at 1200 lines of source code is too large to be published in these pages. However I shall attempt to describe the algorithmatic structure of the software and provide software examples to test different parts of the board.

Codes

The entire project works on the fact that codes (control bytes) are sent from the PC's serial COM1 port when different parts of the menu are clicked on by the mouse. These codes are used in a look up table (in the EPROM) to select subroutines by the 8031 dependent on the function required. The lookup table is of considerable size since they have to accommodate the various permutations of driving 2 stepping

motors simultaneously - stop/go/fast/slow/forward/reverse. Simple maths shows that there are 64 possible combinations and so 64 subroutines had to be written! However, this means that the PC can be dispensed with and the ASCII string generator may be used to send pre-programmed codes to the RS232IO board and control say the motors. (use a spare pin on the D-type connector to supply a 5-volt supply from the RS232IO board to the string generator). Hopefully a joystick (PIC-based) will be designed to perform the same functionality, So included here are a full list of codes: (9600,n,8,1)RS232)

CODE LIST:
==========

ON	PORTA0	AD	PORTB0 A3
OFF	PORTA0	91	PORTB0 87
ON	PORTA1	AF	PORTB1 A5
OFF	PORTA1	93	PORTB1 89
ON	PORTA2	B1	PORTB2 A7
OFF	PORTA2	95	PORTB2 8B
ON	PORTA3	B3	PORTB3 A9
OFF	PORTA3	97	PORTB3 8D
ON	PORTA4	B5	PORTB4 AB
OFF	PORTA4	99	PORTB4 8F
ON	PORTA5	B7	PORTB5 AE
OFF	PORTA5	9B	PORTB5 92
ON	PORTA6	B9	PORTB6 B0
OFF	PORTA6	9D	PORTB6 94
ON	PORTA7	BB	PORTB7 B2
OFF	PORTA7	9F	PORTB7 96

A.C. MAINS
ON 072H
OFF 056H

RESET 00

PORTA OFF 02
PORTA ON 01
PORTB OFF 04
PORTB ON 03

MOTOR A
FAST AA SLOW C6
FWD AC REV C8
GO 28 STOP 0C

MOTOR B
FAST 80 SLOW 9C
FWD 82 REV 9E
GO C2 STOP A6

Embedded Software Development.

There is a Forum on the Electronics and Beyond Website, which I hope will be freely used by readers, since it helps contributors to the magazine to gauge what kind of projects should be included. Software development for embedded controllers such as the PIC and the 8031 is easier today than it was when I started on this 'white

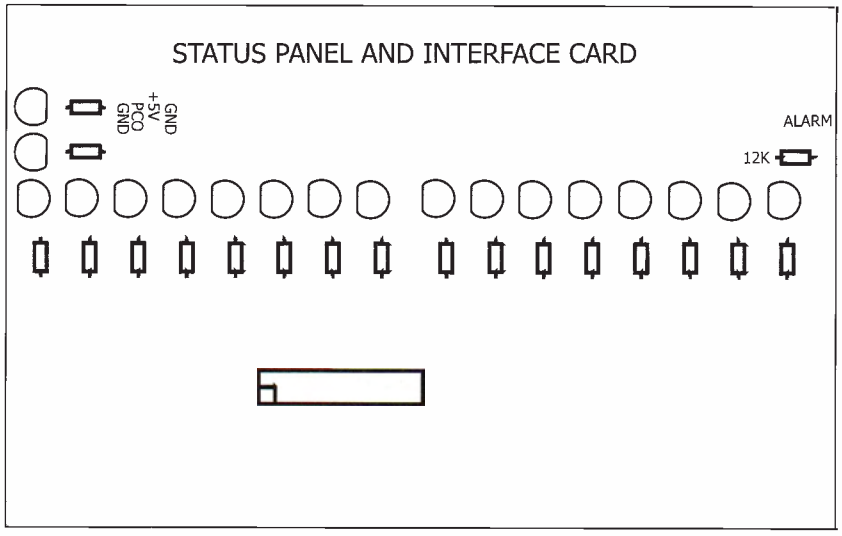


Figure 2. Led panel indicator board layout

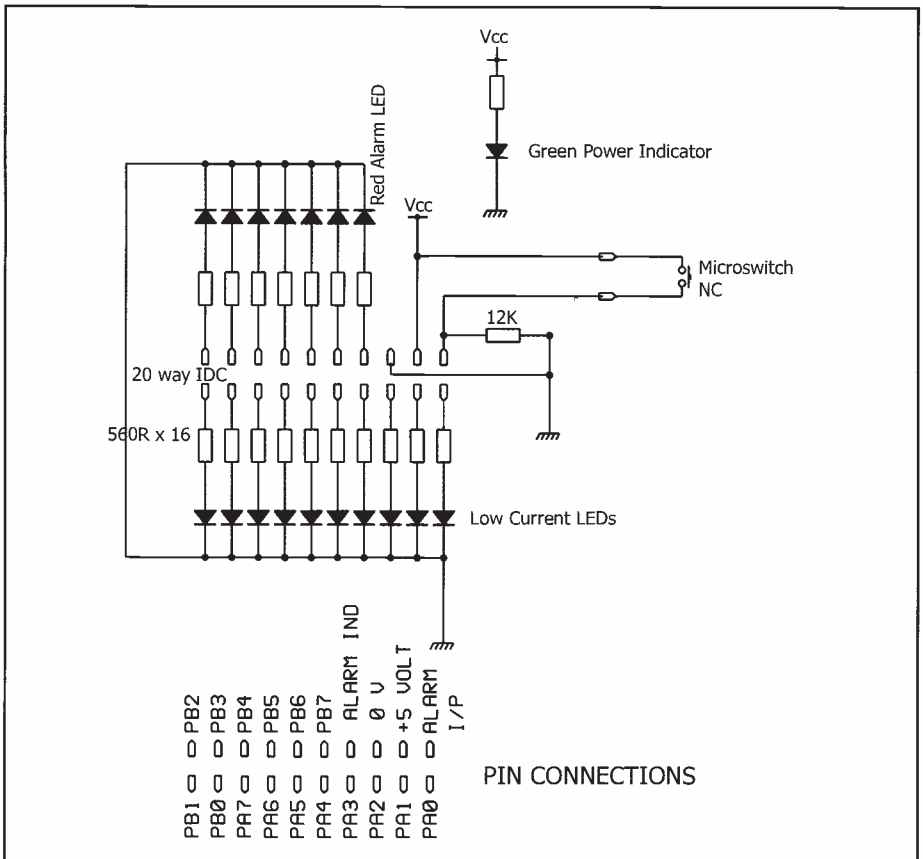
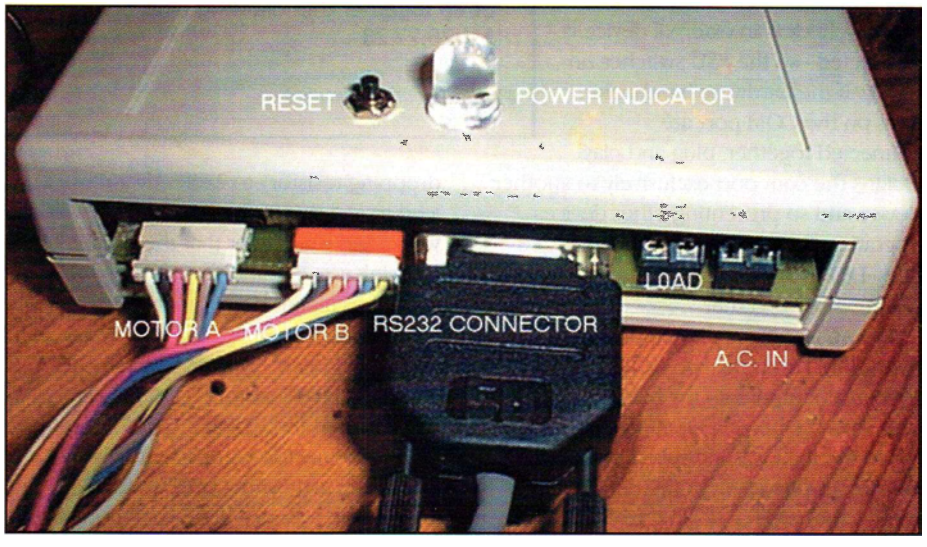


Figure 3. Panel indicator LED wiring diagram



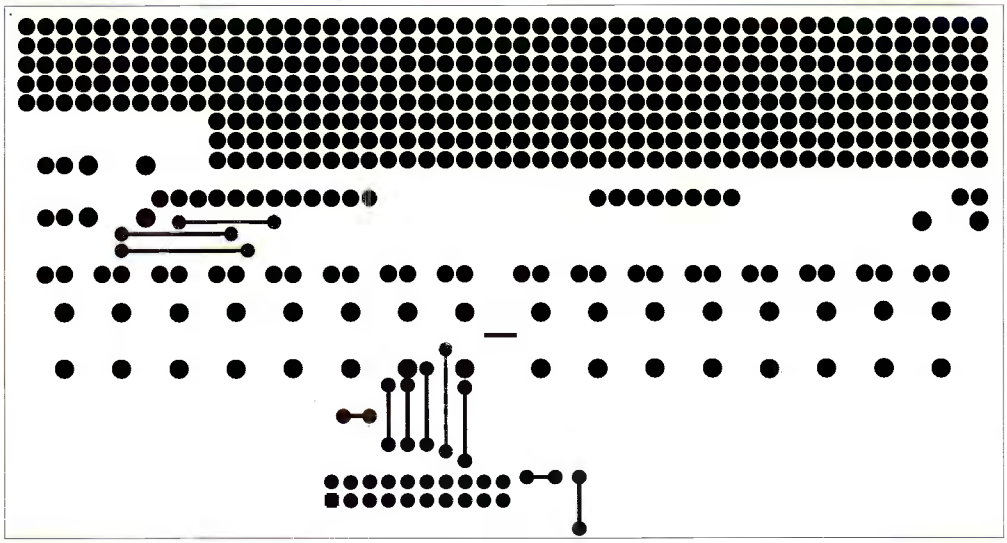


Figure 4. LED PCB top side

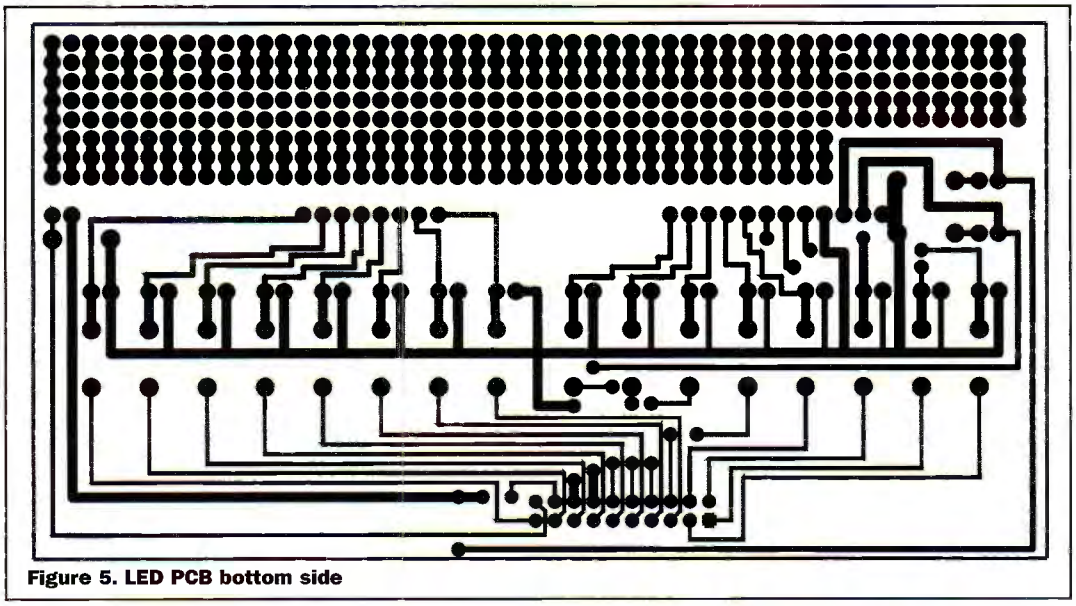
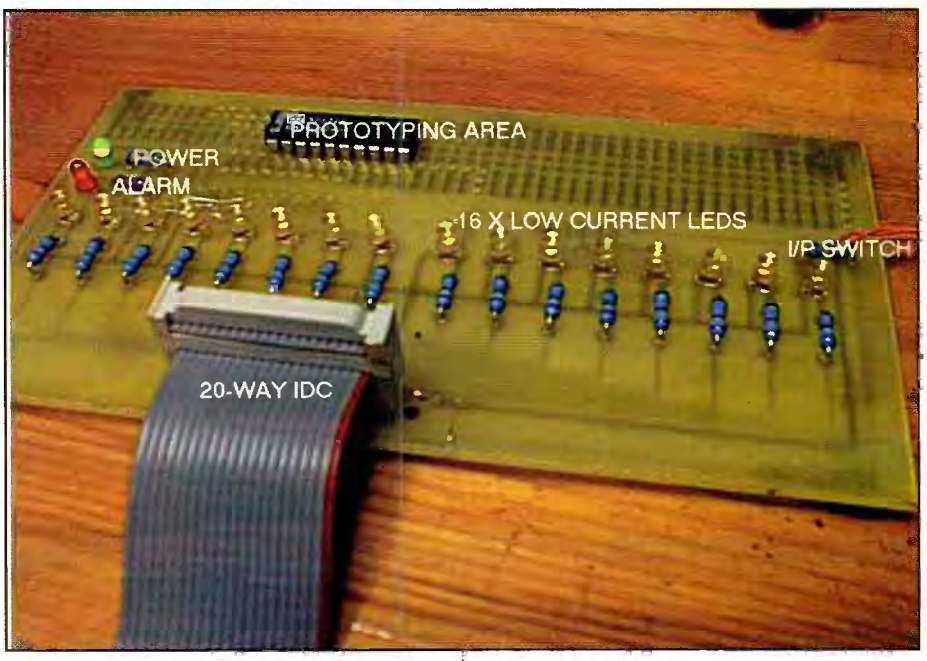


Figure 5. LED PCB bottom side



art' many years ago. Then I was equipped with an instruction set and had to tap in the opcode/ operands via 8 binary switches. Eventually I could enter whole bytes at a time, being displayed on 7-segment display! What joy.

Today software simulators are cheap (under 50 quid) and will run on any PC, so for example, software timing loops can be accurately measured before committing to firmware code. So that's fine but how does one test the EPROM software and get it from the PC to the target board? The answer is to use an EPROM emulator that connects between the PC's parallel port and the EPROM socket of the board. Object code is then simply sent from the PC and resides in flash/eeprom memory of the emulator. When the firmware has been successfully debugged it can then be permanently burnt into an EPROM. I used an EPROM emulator available from Maplin Electronics (order code UT63T) Part No.27MR512-120. This emulator has an 8Kbyte capacity and therefore had to be adapted for the 2 K-byte wide 2716 which has only 24 pins opposed to the 28 pins of a 8K-byte EPROM. See Figure 1.

A 28 dil turned pin socket was used with pins 1,2,27,28 removed. This forms a 24-pin socket that is required by the 2716 eprom. A wire connection on the underside of the socket strapped together pins 1,2,26,27,28.The EPROM emulator then sits on this socket. P.S. Don't forget to wear a static earth strap when handling this device.

A batch file sends 2kbytes of object code (TRY.OBJ) from the PC to the target board:

MICROM TRY.OBJ 2764 BIN 1800 HI 1 NV 1 0X3456

The 1800 is a hex offset so that the 2kbyte block is located in the correct memory map of the emulator.code 0x3456 is a unique i.d. assigned to an individual emulator by the manufacturer. Other emulators are available which are extras to production eprom programmers, but be prepared to pay several hundred pounds.

A pre-programmed EPROM, together with PC software driver on disc is available from: R.Grodzik (Micros) 53 Chelmsford Road Bradford West Yorkshire BD3 8QN England. Price £20.00 sterling P&P inc. Also see the authors web page at: **http://members.netscapeonline.co.uk /dgrodzik**

Test programs

Included on the next page are several short programs to test the functionality of the RS232IO board:

```
;ALARM.ASM
This program flashes the red led on the indicator card when the
microswitch is pressed. At the same time the PC alarm indicator
lights and a simple audible tone is produced on the PC.

#INCLUDE SFR51.EQU


.ORG 0
        LJMP START
.ORG 3                  ;INTERRUPT SERVICE ROUTINE GOES HERE

      CLR ES            ;OISABLE FURTHER INTERRUPTS
      CLR TR1           ;TIMER 1 OFF

      MOV A,#1
      MOV DPTR,#0803H   ;PORT A
      MOVX @DPTR,A      ;OISPLAY BYTE

      RETI              ;RETURN TO BACKGROUND PROGRAM

.ORG 0100H
START:
      ACALL DELAY
      SETB EX0
      CLR IT0
      SETB ES           ;ENABLE RECEIVE DATA INTERRUPT

      SETB TR1          ;TIMER 1 ON

      MOV A,#0FH
      MOV DPTR,#0800H
      MOVX @DPTR,A
      ACALL DELAY

      SETB EA           ;ENABLE GLOBAL INTERRUPT

LOOP:NOP
      LJMP LOOP

OELAY: MOV R1,#$FF
LOOPX: MOV R0,#$FF
INLOOP:OJNZ R0,$
       DJNZ R1,LOOPX
       RET

.org  0800H
.ENO


;MOTOR.ASM
This program simply tests both stepper motors. The speed is
adjusted by writing different values into R6.


#INCLUOE SFR51.EQU

.ORG 0h
HERE:

         LCALL DEL
         MOV P1,#088H
         LCALL DEL
         MOV P1,#0CCH
         LCALL DEL
         MOV P1,#044H
         LCALL DEL
         MOV P1,#066H
         LCALL DEL
         MOV P1,#022H
         LCALL DEL
         MOV P1,#033H
         LCALL DEL
         MOV P1,#011H
         LCALL DEL
         MOV P1,#099H

     SJMP HERE


OEL:
    MOV R6,#5   ;FAST SPEED
                ;MEDIUM 10
                ;SLOW  48

D2: OEC R6
    MOV R7,070H
D1: DEC R7
   CJNE R7,#0,D1
   CJNE R6,#0,O2
   RET


.ORG 0800H

.END



;RAMPDOWN.ASM
;Start at maximum speed (100Hz rate),reduce speed to a minimum
;and then stop

#INCLUOE SFR51.EQU

.ORG 0h

         MOV R0,#10    ; STEP VAL 50    ;50 X4 STEPS = 20
STEPS
                       ;=1 REVOLUTION
         MOV R1,#3     ;INITIAL SPEED
AGAIN:   PUSH 1
        ACALL STEPS
         POP 1
         INC R1
         CJNE R1,#24,MORE
         SJMP STOP

MORE:    MOV R0,#10     ;STEP VAL
         AJMP AGAIN

STOP:   MOV P1,#0
FOREVER:SJMP FOREVER

STEPS:
HERE:
         LCALL DEL        ;4 STEPS/CYCLE
         MOV P1,#088H
         LCALL DEL
         MOV P1,#0CCH
         LCALL DEL
         MOV P1,#044H
         LCALL DEL
         MOV P1,#066H
         LCALL DEL
         MOV P1,#022H
         LCALL DEL
         MOV P1,#033H
         LCALL DEL
         MOV P1,#011H
         LCALL DEL
         MOV P1,#099H
          DJNZ R0,HERE
          RET



DEL:    MOV A,R1
        MOV R6,A

   ;MOV R6,#3   ;FAST SPEED
                ;MEDIUM 10
                ;SLOW  48

D2: OEC R6
    MOV R7,070H
D1: DEC R7
   CJNE R7,#0,D1
   CJNE R6,#0,D2
   RET


.ORG 0800H

.ENO


RAMPUP.ASM

  ;INCREASE SPEED EVERY HALF REVOLUTIION AND THEN KEEP  MAX
SPEED

#INCLUDE SFR51.EQU

.ORG 0h

         MOV R0,#25    ;STEP VAL        ;50 X4 STEPS = 200 STEPS
                                          ;=1 REVOLUTION
         MOV R1,#24    ;INITIAL SPEED SLOW
AGAIN:   PUSH 1
        ACALL STEPS
         POP 1
         DEC R1
         CJNE R1,#3,MORE
         SJMP RUN
MORE:    MOV R0,#10     ;STEP VAL
         AJMP AGAIN

RUN:     MOV R0,#5
         MOV R1,#3
         ACALL STEPS
         SJMP RUN

STEPS:
HERE:
         LCALL DEL        ;4 STEPS/CYCLE
         MOV P1,#088H
         LCALL DEL
         MOV P1,#0CCH
         LCALL DEL
         MOV P1,#044H
         LCALL DEL
         MOV P1,#066H
         LCALL DEL
         MOV P1,#022H
         LCALL DEL
         MOV P1,#033H
         LCALL DEL
         MOV P1,#011H
         LCALL DEL
         MOV P1,#099H
          DJNZ R0,HERE
          RET



DEL:    MOV A,R1
        MOV R6,A

   ;MOV R6,#3   ;FAST SPEED
                ;MEDIUM 10
                ;SLOW  48

02: OEC R6
    MOV R7,070H
01: DEC R7
   CJNE R7,#0,01
   CJNE R6,#0,D2
   RET


.ORG 0800H

.ENO
```

Note that the relationship between stepper motor speed and delay time between steps is not linear and follows an exponential curve.
So, some room for development here!.

```
RECEIVE.ASM
This program tests the alarm led and displays the binary pattern
on portA of bytes sent from the PC.

#INCLUDE SFR51.EQU


.ORG 0
        LJMP START

.ORG 023H               ;INTERRUPT SERVICE ROUTINE GOES HERE

      CLR ES            ;DISABLE FURTHER INTERRUPTS
      CLR TR1           ;TIMER 1 OFF

      MOV A,SBUF
      MOV DPTR,#0801H   ;PORT A
      MOVX @DPTR,A      ;DISPLAY BYTE

      SETB TR1          ;TIMER 1 ON
      SETB ES           ;ENABLE INTERRUPTS
      CLR RI            ;CLEAR RECEIVE BYTE FLAG

      RETI              ;RETURN TO BACKGROUND PROGRAM

.ORG 0100H
START:
      ACALL DELAY

      MOV SCON,#050H    ;SERIAL MODE 1
      MOV TMOD,#020H    ;TIMER 1  8-BIT AUTO RELOAD
      MOV TH1,#0F0H     ;TIMER RELOAD VALUE FOR 9600 BAUD-
11.0592
      SETB ES           ;ENABLE RECEIVE DATA INTERRUPT
      SETB TR1          ;TIMER 1 ON

      MOV A,#0FH
      MOV DPTR,#0800H
      MOVX @DPTR,A
      ACALL DELAY

      SETB EA           ;ENABLE GLOBAL INTERRUPT

LOOP:NOP
      ACALL DELAY
      MOV A,#1
      MOV DPTR,#0803H
      MOVX @DPTR,A
      ACALL DELAY
      MOV A,#0
      MOV DPTR,#0803H
      MOVX @DPTR,A
      LJMP LOOP

DELAY: MOV R1,#$FF
LOOPX: MOV R0,#$FF
INLOOP:DJNZ R0,$
       DJNZ R1,LOOPX
       RET

.org  0800H
.END



TRANSMIT.ASM
This program sends a swirl ascii pattern form the RS232I0 board
to the PC.


#INCLUOE SFR51.EQU

.ORG 0h
        MOV A,#030h
CYCLE:
        ACALL TXD
        INC A
        MOV P1,A
        ACALL DEL
        LJMP CYCLE

                        ;TXO SUBROUTINE

TXO:  MOV SCON,#050H    ;SERIAL MODE 1,RECEIVER ENABLED
      MOV TMOD,#020H    ;TIMER 1  8-BIT AUTO RELOAD
      MOV TH1,#0FDH     ;TIMER RELOAD VALUE FOR 9600 BAUD-
11.0592

      CLR EA            ;DISABLE GLOBAL INTERRUPT
      MOV SBUF,A      ;LOAD SERIAL BUFFER WITH OATA
      SETB TR1        ;TIMER 1 ON
LOOP: JNB TI,LOOP     ;IF FLAG TI IS SET,BYTE HAS BEEN
TRANSMITTED
      CLR TR1         ;TIMER 1 OFF
      CLR TI          ;CLEAR BYTE RECIEVED FLAG
      RET             ;RETURN TO CALLING PROGRAM

DEL:MOV R1,#$95
LOOP2:MOV R0,#$F
INLOOP:DJNZ R0,$
       DJNZ R1,LOOP2
       RET

.ORG 0800H
.END
```