APPENDICES

A - Paperwork

B - Additional Mathematical Information

C - System Software + History + variable locations
- C1 - Main routine (TEST9.ASM)
- C2 - Analogue to Digital (ADCSUBS.ASM)
- C3 - I²C Serial Comms (I2CSUBS.ASM)
- C4 - L.C.Display (LCDSUBS.ASM)
- C5 - Peltier Drive (PWMSUBS.ASM)
- C6 - RS232 Serial Comms (RS232SUB.ASM)
- C7 - Texture Driver (TEXSUBS.ASM)
- C8 - Main Header File (SFR.H)
- C9 - Special Values Header (SSTVALUE.H)

D - Full build state details
- D1 - HUMAND PCB.
- D2 - POWERC2 PCB.
- D3 - MECHAND PCB.
- D4 - RS232 PCB.
- D5 - HEATER PCB.
- D6 - TEXOUT PCB.
- D7 - TEXIN PCB.
- D8 - LCD Circuit.
- D9 - EEPROM PCB.
- D10 - Operator's thermal finger Assembly.

E - PC Software + History + Instructions
- E1 - Data collection / Graphing (TGRAPHS)
- E2 - RS232 Comms (ASYNCH.C)
- E3 - RS232 Comms header (ASYNCH.H)
- E4 - Circular Queue (CIRCLEQ.C)
- E5 - Circular Queue (CIRCLEQ.H)
- E6 - Keycodes Header (KEYCODES.H)

F - Objects used in testing

G - Test Results
- G1 - Object material thermal reaction/recovery times & end points.
- G2 - Material recognition tests.
- G3 - Response times for Operator end of system, reacting to step changes.
- G4 - Weight of system components.
- G5 - Operator response times to temperature increases / drops.

This report was written using WordPerfect.
Report length 18,000 word, excluding appendices.
Graphs drawn using Wordperfect Presentation.
Spreadsheet software used was Quattro Pro ver 4.
PC 'C' program compiled using Borland C++ 3.1
System software compiled using TASM

PCB layout / circuit diagrams drawn using PADS PCB

For $T_{Mheater}$ to have a range of 0◦C to 54◦C

therefore volts / ◦C = 5 / 54 = approx 92.6 mv / ◦C

amp I/p V/◦C = 10mv / ◦C
R1/R2 = 3k9 / 475Ù = 8.21
gain = 1 + 8.21 = 9.21
therefore amp O/P v/◦C = 92.1052mv
max temperature value = 5v / 92.1052mv = 54.286◦C

For T_{Mtouch} to have a range of 0◦C to 59◦C

therefore volts / ◦C = 5 / 59 = approx 84 mv / ◦C

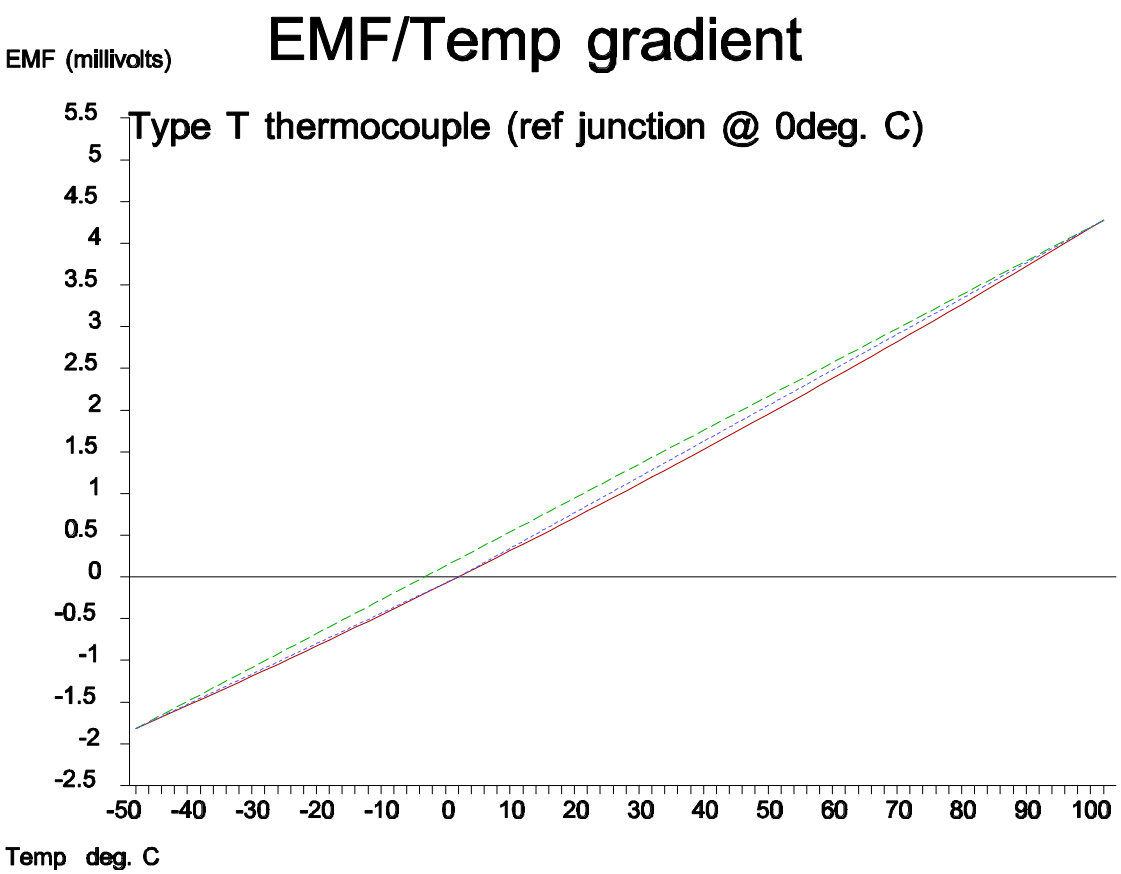
amp I/p V/◦C = 40.25Ì v / ◦C
R1/R2 = 1M / 475Ù = 2105.3
gain = 1 + 2105.3 = 2106.3
therefore amp O/P v/◦C = 84.78mv
max temperature value = 5v / 84.78mv = 58.98◦C

For $T_{Mambient}$ to have a range of 0◦C to 40◦C

therefore volts / ◦C = 5 / 40 = approx 125 mv / ◦C

amp I/p V/◦C = 10mv / ◦C
R1/R2 = 150k / 13k = 11.54
gain = 1 + 11.54 = 12.54
therefore amp O/P v/◦C = 125.4mv
max temperature value = 5v / 125.4mv = 39.88◦C

Thermocouple response graph



Dotted lines show two possible approximations to the non-linear thermocouple function.

Compiler

The code in this appendix was compiled using TASM - Table Driven Assembler Version 2.7 (A Shareware compiler). TASM is a table driven cross assembler for the MS-DOS environment. Currently, tables for the 6502, 8048, 8051, 8080/8085, Z80, 6805 and TMS32010 microprocessors are supported.

The batch file used to compile the code was;

TASM -51 -C -F00 -G0 -l %1 %2 %3

where %1 is the name of the main code file (eg TEST8.ASM)

History

VERSION		
	1.0	- Initial write - unsatisfactory performance
	2.0 to 5.0	- Tested various subroutines.
	6.0	- 1st completely working software
	7.0	- Restructured - unsatisfactory results
	8.0	- Rewrite comms to include jump,offset,tsteady,absolute commands
	8.1	- added 'proportional power max value set by comms ie command 'P'
14/12/94	8.2	- switched off interrupts while getting packet of data for comms, so no values changed half way though.
16/12/94	8.3	- added '@' command to change comms speed to 4800
19/02/95	8.5	- added power adjustment for the difference between the two faces of the peltier.
10/3/95	8.6	- fixed bug giving incorrect value after the decimal point with comms command 'A'. Added ver declaration on LCD, on switch on.
	9.0	- added all texture stuff, and 16660baud serial - didn't work cos screwed I2C
10/4/95	9.2	- didn't work - due to bug in compiler!!
11/4/95	9.3	- latest version

ROUTINE - TEST9.ASM 05/04/94

```
;//////////////////////////////////////////////
;//           PSEUDTEMPerature run-code         //
;//              version test9_3.asm            //
;//             by Steve Lawther                //
;//////////////////////////////////////////////
;
                .TITLE  "TEMPERATURE"
#INCLUDE "SFR.H"
#INCLUDE "SSTVALUE.H"
;*********************************************************************
;Routine:- Startup
;************************
;
;
                .ORG     0000H
STARTUP:        AJMP    INITIALIZE

                .ORG    0100H
INITIALIZE:     CLR     P4.2
                MOV     P4,#00H ;stop LCD enable(port reset to FFh)
                MOV     R0,#255
RESETRAM:       MOV     @R0,#0
                DJNZ    R0,RESETRAM
                MOV     SP,#0B0H        ;move stack to higher location
                LCALL   LCDINIT
                LCALL   TEMP_DISINIT
                LCALL   I2CINIT
                LCALL   CONFACTSINIT
                LCALL   TEXINIT
                LCALL   INITSERIAL
                MOV     TSTORE6,#STDYSTATETEMP
                MOV     SOUT_T6,#STDYSTATETEMP
                LCALL   PWMINIT
                MOV     TM2CON,#11001101B   ;set 8&16bit int's, clk/8(1.5mS), T2 on
                ORL     IEN0,#11100000B ;enables global,ADC,I2C
                MOV     IP0,#01100000B  ;ADC & I2C high priority
;T2 overflow low priority already as all IP1 set low p by reset
;ok, in 1.5mS there will be an interrupt due to 8-bit overflow
;this int will move chrs to LCD if flag set. that all at the mo.
;The 16-bit overflow, after 392mS will read/write to POWERC2
                MOV     DPTR,#CHRSETUP
                LCALL   STARTLCDROM
                MOV     IEN1,#10000000B ;enables T2 overflows
LOOPY:          INC     (IDLETIME + 1)
                MOV     A,(IDLETIME + 1)
                JZ      IDLE_OV
                NOP
                NOP
                SJMP    LOOPY

IDLE_OV:        INC     IDLETIME
                NOP
                SJMP    LOOPY

;new bit, 16/10/94 - clearer, fast and the same length as the old routine
CONFACTSINIT:   MOV     CON_FACT0,#CONVFACTOR0  ;91 at mo
                MOV     CON_FACT1,#CONVFACTOR1  ;40 at mo
                MOV     CON_FACT2,#CONVFACTOR2  ;91 at mo
                INC     CON_FACT3               ;makes this byte = 1
                MOV     CON_FACT4,#CONVFACTOR4  ;54 at mo
                MOV     CON_FACT5,#CONVFACTOR5  ;54 at mo
                INC     CON_FACT6               ;makes this byte = 1
                MOV     CON_FACT7,#CONVFACTOR7  ;40 at mo
                RET

;CONFACTSINIT:   MOV     R0,#CON_FACT7   ;set R0 to addr of last con_fact
;                MOV     R2,#08H         ;to store 8 bytes
;NEXT_CONV_F:    MOV     A,R2
;                ACALL   CON_DEF_TBL
;                MOV     @R0,A
;                DEC     R0
;                DJNZ    R2,NEXT_CONV_F
;                RET
;CON_DEF_TBL:    MOVC    A,@A+PC
;                RET
;                        T0h T1h T2h    T4m T5m    T7m
;                .DB      91, 40, 91, 1, 54, 59, 1, 40 ;40
;                                     htr tch    amb
;*********************************************************************
;Routine:- T2 INTERRUPT
;**********************
;
```

```
;Priority:-  LOW
;
;Called by:- T2 INTERRUPT
;
;Origin:- 0073H
RBANK2  .EQU    00010000B

REMADD  .EQU    $

                .ORG     0073H
T2INT:          PUSH    PSW
                PUSH    ACC
                MOV     PSW,#RBANK2
                JBC     T2OV,T2_16INT   ;check to see if 8 or 16-bit overflow
T2_8INT:        ANL     TM2CON,#11101111B
                JB      SHUT_REQ_bit,SHUTDOWN
SHUTBACK:       LCALL   LCD_DRIVER      ;route any messages to LCD
                JB      SHUT_REQ_bit,ENDT2
                LCALL   PWMDRIVE
                LCALL   TEXTUREDRIVE
                LCALL   STARTI2CMEC
                LCALL   STARTADC
ENDT2:          POP     ACC
                POP     PSW
                RETI
SHUTDOWN:       MOV     PWM0,#0
                JB      SHUT_ACK_bit,SHUTBACK
;******* ADDED THIS NEXT LINE TO TRY AND STOP THE PSEUDO-LOCKUP ON SHUTDOWN
                JB      LCD_DRDY_bit,SHUTBACK
                LCALL   STARTLCDROM
                SETB    SHUT_ACK_bit
                SJMP    SHUTBACK
;ok, a 16-bit overflow
T2_16INT:       ANL     TM2CON,#11101111B       ;get rid of the 8-bit interrupt
                ;CLR     T2OV            ;and 16 bit
                LCALL   DIS_POW
                LCALL   STARTI2CPOW
                INC     T2EX
                JB      SHUT_REQ_bit,SHUTSHIFT
                LCALL   TEMP_DIS
                MOV     LCD_NEXT_CHR,#T_DISP
                LCALL   STARTLCDRAM
ENDT216:        POP     ACC
                POP     PSW
                RETI
SHUTSHIFT:      JNB     SHUT_ACK_bit,ENDT216
                JB      LCD_DRDY_bit,ENDT216
                MOV     A,T2EX
                ANL     A,#00000011B
                JNZ     ENDT216
                MOV     DPTR,#SHIFTIT
                LCALL   STARTLCDROM
                SJMP    ENDT216

                .ORG    REMADD
; display power on powerc2
DIS_POW:        MOV     POWDIS,#11111111B
                MOV     A,PWM1
                CJNE    A,#255,COLDPOW
                MOV     A,PWM0
                JZ      BEATDISPOW
                CLR     C
                SUBB    A,#64
                JC      LOWPOWHEAT
                MOV     POWDIS,#11110111B
                SJMP    BEATDISPOW
LOWPOWHEAT:     MOV     POWDIS,#11101111B
                SJMP    BEATDISPOW
COLDPOW:        MOV     A,PWM0
                JZ      BEATDISPOW
                CLR     C
                SUBB    A,#64
                JC      LOWPOWCOLD
                MOV     POWDIS,#01111111B
                SJMP    BEATDISPOW
LOWPOWCOLD:     MOV     POWDIS,#10111111B
BEATDISPOW:     JB      T2EX.2,ENDISPOW
                ANL     POWDIS,#11011111B
ENDISPOW:       RET
;*********************************************************************
;Routine:- TEMP_DIS
;******************
;converts the temperatures to BCD for display
```

```
;
;Called by:- T2_16bit
;
;Origin:-
;
;Requires:-
;       R5      stores temperature number now being dealt with
;       A       various
;       R1      contains temperature table address
;       R0      contains address in display memory for number
;       B       second acc
;
; stored byte:-
; store bit:-

TEMP_DISINIT:   MOV     R5,#35
                MOV     R0,#(T_DISP+34)
NXT_BLNK_CHR:   MOV     A,R5
                LCALL   GETDISBLANK
                MOV     @R0,A
                DEC     R0
                DJNZ    R5,NXT_BLNK_CHR
                RET

GETDISBLANK:    MOVC    A,@A+PC
                RET
                .DB     32
                .DB     32,32,0,32,32,32,32,0,32,32,32,0,32,32,32,32
                .DB     LNXTLNE
                .DB     32,32,32,32,32,32,32,0,32,32,32,32,0,32,32,32
                .DB     0


TEMP_DIS:       MOV     R5,#0
                MOV     R1,#TSTORE0
NEXT_TEMP:      LCALL   LOOK_UP
                JZ      MISS_TEMP
                CJNE    A,#255,NOT_ENDDIS
                MOV     R0,#T_DISP
                MOV     @R0,#LSTART
                MOV     R0,#(T_DISP+17)
                MOV     @R0,#LNXTLNE
                MOV     R0,#(T_DISP+34)
                MOV     @R0,#LFINISH
                RET

MISS_TEMP:      INC     R1
                INC     R1
                SJMP    NEXT_TEMP

NOT_ENDDIS:     MOV     R0,A
                LCALL   CONV_N_DIS
                SJMP    NEXT_TEMP

LOOK_UP:        INC     R5
                MOV     A,R5
                MOVC    A,@A+PC
                RET
                .DB     (T_DISP+22)     ;TEMP - Hand/Peltier
                .DB     (T_DISP+27)     ;TEMP - Hand,finger temp
                .DB     (T_DISP+32)     ;TEMP - Hand,Peltier heatsink
                .DB     0               ;TEMP - Hand,SPARE
                .DB     (T_DISP+1)      ;TEMP - MECHAND, heater
                .DB     (T_DISP+5)      ;TEMP - MECHAND, skin
                .DB     (T_DISP+10)      ;TEMP - MECHAND, steady state
                .DB     1               ;SHOW STATUS
                .DB     2               ;SHOW POWER
                .DB     255             ;END


CONV_N_DIS:     JNB     ACC.7,HEXSHOW
                MOV     A,@R1
                INC     R1
;                MOV     B,#100          ;now for the hundreds
;                DIV     AB
;                ADD     A,#48
;                CJNE    A,#48,NOT_SPACE
;                MOV     A,#32
;NOT_SPACE:      MOV     @R0,A
;                INC     R0
                MOV     B,#10           ;now for the tens - now assume not > 99
;                XCH     A,B
                DIV     AB
```

```
                ADD     A,#48
                CJNE    A,#48,NOT_SPACE2
                MOV     A,#32
NOT_SPACE2:     MOV     @R0,A
                INC     R0
                MOV     A,#48           ;now for the units
                ADD     A,B
                MOV     @R0,A
                INC     R0
                MOV     A,@R0
                JZ      NO_FRACTION
;These changes 5/4/95 to speed up
                MOV     A,@R1
                ANL     A,#11000000B       ;7 bytes and 6 cycles
                RL      A
                RL      A
                INC     A
                MOV     @R0,A
NO_FRACTION:    INC     R1
;                MOV     @R0,#32           ;10 bytes and 8 or 5 cycles
;                ANL     A,#11000000B
;                JZ      NO_FRACTION
;                RL      A
;                RL      A
;                MOV     @R0,A
;NO_FRACTION:    INC     R1
                RET

HEXSHOW:        CJNE    A,#1,SHOW_POWER
                MOV     R0,#(T_DISP+13)
                MOV     A,T2EX
                ANL     A,#00000111B
                INC     A
                LCALL   HARTBEAT
                MOV     @R0,A
                INC     R0
                MOV     A,IDLETIME
                SWAP    A
                LCALL   HEXDIGIT
                MOV     A,IDLETIME
                LCALL   HEXDIGIT
                MOV     A,(IDLETIME+1)
                SWAP    A
                LCALL   HEXDIGIT
                CLR     A                       ;Clears idletime - done this way to save
                MOV     IDLETIME,A              ;1 cycle and 1 byte!
                MOV     (IDLETIME + 1),A
                RET

HARTBEAT:       JBC     ADC_OVERLAP,ADCOVER
                JBC     I2C_OVERLAP,I2COVER
                MOVC    A,@A+PC
                RET
                .DB     32,165,'+','*','#',219,'O',255

ADCOVER:        MOV     A,#'A'
                CLR     I2C_OVERLAP
                RET
I2COVER:        MOV     A,#'I'
                RET
;this routine takes 24 bytes and 8 cycles
;HEXDIGIT:       ANL     A,#00001111B
;                ADD     A,#03           ;allow for offset from movc to table
;                MOVC    A,@A+PC
;                MOV     @R0,A
;                INC     R0
;                RET

;HEXDATA         .DB     '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'

;this routine takes 14 bytes and  10/11 cycles
HEXDIGIT:       ANL     A,#00001111B
                CLR     C
                SUBB    A,#10
                JC      HNUMBER
                ADD     A,#7
HNUMBER:        ADD     A,#58
                MOV     @R0,A
                INC     R0
                RET

SHOW_POWER:     MOV     R0,#(T_DISP+18)
                MOV     R1,PWM1
```

```
                MOV     @R0,#'-'
                CJNE    R1,#255,MINUS
                MOV     @R0,#'+'
MINUS:          INC     R0
                MOV     A,PWM0
                SWAP    A
                LCALL   HEXDIGIT
                MOV     A,PWM0
                LCALL   HEXDIGIT
                RET


#INCLUDE "ADCSUBS.ASM"
#INCLUDE "LCDSUBS.ASM"
#INCLUDE "I2CSUBS.ASM"
#INCLUDE "RS232SUB.ASM"
#INCLUDE "TEXSUBS.ASM"
#INCLUDE "PWMSUBS.ASM"
CHRSETUP:       .DB     LSTART
                .DB     LTOGRS          ;clr RS
                .DB     01000000B       ;set CG address to 0,ready to load chrs
                .DB     LTOGRS          ;set RS

                .DB     136,148,136,134,137,136,137,134 ;chr 0 DEG C
                .DB     130,133,133,133,130,128,152,152 ;chr 1 POINT0
                .DB     134,129,130,132,135,128,152,152 ;chr 2 POINT2
                .DB     135,132,134,129,134,128,152,152 ;chr 3 POINT5
                .DB     135,129,129,130,130,128,152,152 ;chr 4 POINT7
                .DB     152,132,136,144,156,128,128,128 ;chr 5 SQUARED
                .DB     255,241,251,251,251,251,255,255 ;chr 6 TEST
                .DB     LTOGRS          ;toggle RS
                .DB     10000000B       ;set to display ram, posn 0
                .DB     LTOGRS          ;set rs
                .TEXT   VERNO
                .DB     32,0,1,2,3,4,5,6
                .DB     LFINISH
```

ROUTINE - ADCSUBS.ASM 29/9/94

```
;*************************************************************************
;
;Routine:- ADC INTERRUPT
;************************
;
;
;Called by:- ADC finished Interrupt
;
;notes:-        converts a data value to a Temp, and stores it, before setting
;               up for next ADC conversion, if not finished
;
;Origin:- 0053H for stub, arbitary for main code
;
;Uses:-
;       A       Accumulator
;       B       B register
;BANK1  R0      address of conversion factor
;BANK1  R1      address to store answer at
;
; stored byte:-
; store bit:-
;       ADC_END_bit     set to indicate ADC finished this set of convertns
;
;all 10bits now used
;NOTE:- If this routine can be shortened to <32 bytes all of it can be contiguous from 53h

RBANK1          .EQU    00001000B
ADCTEMPPOS      .EQU    $

                      .ORG     0053H
ADCINT:               AJMP    ADCSERVE

                      .ORG    ADCTEMPPOS

STARTADC:             MOV     BNK1R0,#CON_FACT0
                      MOV     BNK1R1,#TSTORE0
                      JBC     ADC_END_bit,ADCOK
                      SETB    ADC_OVERLAP
ADCOK:                MOV     ADCON,#11000000B
                      ORL     ADCON,#00001000B
                      RET

ADCSERVE:             PUSH    PSW
                      MOV     PSW,#RBANK1     ;set to use reg bank 1
                      XCH     A,R2            ;store A, quicker n smaller than push
                      PUSH    B
                      MOV     B,@R0           ;R0 contains address of con.fact's
                      MOV     A,ADCH          ;A now contains data value
                      MUL     AB
                      MOV     @R1,B           ;high byte (UNITS) stored
                      INC     R1
                      MOV     @R1,A           ;low byte (POINTS) stored

;****************
;
;new bit
                      MOV     B,@R0           ;R0 contains address of con fact's  2
                      INC     R0                                                  ;  -
                      MOV     A,ADCON                                             ;  3
                      ANL     A,#11000000B                                        ;  4
                      MUL     AB                                                  ;  8
                      MOV     A,B             ;move high byte into A              ;  9
                      ADD     A,@R1           ;add points to high byte            ;  10
                      MOV     @R1,A           ;and move it back                   ;  11
                      JNC     ENDADCCALC      ;if no carry to units               ;  13
                      DEC     R1                                              ;  14
                      INC     @R1             ;add carry to units             ;  15
                      INC     R1                                              ;  16
ENDADCCALC:           INC     R1                                              ;     -

;****************

                      ANL     ADCON,#11000011B        ;clear interrupt flag
                      INC     ADCON           ;increase channel number
                      MOV     A,ADCON
                      JB      ACC.2,ENDADC   ;end ADC convertn if ADC3 was read
                      ORL     ADCON,#00001000B        ;start next convertn
ADC_TIDYUP:           POP     B
                      XCH     A,R2
                      POP     PSW
                      RETI

ENDADC:               SETB    ADC_END_bit     ;flag ADC finished this set of readings
                      SJMP    ADC_TIDYUP
```

ROUTINE - I2CSUBS.ASM 29/9/94

```
;////////////////////////////////////////////
;//           PSEUDTEMPerature run-code        //
;//                 version a.04               //
;//             by Steve Lawther               //
;////////////////////////////////////////////
;
;

;To setup I2C Bus...leave S1ADR set to 00H, set SCL & SDA pins high
I2CINIT:              SETB    P1.6
                      SETB    P1.7
                      MOV     HADD,#05
;S1CON set to 100kHz (CSR2=0,CSR1=CSR0=1),ENSI set,AA reset,all others reset
                      MOV     S1CON,#NSTA_NSTO_NAA
                      SETB    STA             ;ask for a start condition
;test to see if MECHAND is alive and kickin'
                      MOV     DPTR,#ERRSTA
                      MOV     A,#08H
                      JNB     SI,$            ;no start condition until SI set
                      CJNE    A,S1STA,I2CERROR        ;check for error
                      MOV     S1DAT,#WRITE_MECHAND    ;addr of MECHAND + Write (0)
                      MOV     S1CON,#NSTA_NSTO_AA     ;Send addr+W
                      MOV     DPTR,#ERRMEC
                      MOV     A,#18H
                      JNB     SI,$            ;wait for status (80uS'ish)
                      CJNE    A,S1STA,I2CERROR
;mechand has ack'ed - send setup AOUT off,4 single I/P's auto-Incr on,a/d #1
                      MOV     S1DAT,#00000101B
                      MOV     S1CON,#NSTA_NSTO_AA     ;send setup data
                      MOV     A,#28H
                      JNB     SI,$            ;wait for status
                      CJNE    A,S1STA,I2CERROR

;ok,so MECHAND is there, now for POWERC2...first a repeat start
                      MOV     S1CON,#STA_NSTO_AA
                      MOV     DPTR,#ERRBUS    ;set error code back to bus problem
                      MOV     A,#10H
                      JNB     SI,$
                      CJNE    A,S1STA,I2CERROR
                      MOV     S1DAT,#WRITE_POWER      ;addr of POWERC2 +Write (0)
                      MOV     S1CON,#NSTA_NSTO_AA
                      MOV     DPTR,#ERRPOW
                      MOV     A,#18H
                      JNB     SI,$
                      CJNE    A,S1STA,I2CERROR
                      MOV     S1DAT,#00000111B        ;all LED's on,and I/P's high
                      MOV     S1CON,#NSTA_NSTO_AA
                      MOV     A,#28H
                      JNB     SI,$
                      CJNE    A,S1STA,I2CERROR
                      MOV     S1CON,#NSTA_STO_AA      ;now to send STOP for the mo.
                      JB      STO,$           ;wait until stop has been transmitted
                      RET
I2CERROR:             SETB    SHUT_REQ_bit
                      RET


STARTI2CMEC:          MOV     SLA_ADD,#READ_MECHAND
                      JBC     I2C_FIN_bit,I2C_OK
                      SETB    I2C_OVERLAP
I2C_OK:               MOV     S1CON,#STA_NSTO_AA
                      RET
STARTI2CPOW           MOV     SLA_ADD,#READ_POWER
                      SETB    POW_bit
                      CLR     POWKEY.3
                      MOV     S1CON,#STA_NSTO_AA
                      RET

;*********************************************************************
;Routine:- I2C INTERRUPT
;***********************
;
;Called by:- I2C sent/received byte
;
;       notes:-
;
;
;Origin:- 002BH for stub, arbitary for main code
;
;       notes:- stub can be up to 40 bytes long
;
```

```
;Uses:-
;
; stored byte:-
;       HADD    high address byte of jump to I2C routines
;       SLA     address of
; store bit:-

;DEFS
NSTA_STO_AA     .EQU    01010111B
NSTA_NSTO_AA    .EQU    01000111B
NSTA_NSTO_NAA   .EQU    01000011B
STA_NSTO_AA     .EQU    01100111B
READ_POWER      .EQU    01000001B
READ_MECHAND    .EQU    10010001B
WRITE_POWER     .EQU    01000000B
WRITE_MECHAND   .EQU    10010000B


                .ORG    002BH
I2CINT:         PUSH    PSW             ;save psw
                PUSH    ACC
                MOV     A,S1STA
                CJNE    A,#60H,FOOLOOP  ;a false compare n jump, just to check
FOOLOOP:        POP     ACC
                JNC     ERRMMETC                ;under 60h otherwise multimaster
error
                PUSH    S1STA
                PUSH    HADD
                RET                     ;jumps to address HADD,S1STA

ERRMMETC:       AJMP    ERR_MM
;main subroutines for 00H to 58H; already rid of 60H and above
                .ORG    0500H   ;code 00H; bus error
                MOV     S1CON,#NSTA_STO_AA
                MOV     DPTR,#ERRBUS
                SJMP    ERROR

                .ORG    0508H   ;code 08H; start sent, waiting for address
                MOV     S1DAT,SLA_ADD
                MOV     S1CON,#NSTA_NSTO_AA
                SJMP    END_I2C

                .ORG    0510H   ;code 10H; repeat start,waiting for address
                MOV     S1DAT,SLA_ADD   ;note addr changed to write at RSTA
                MOV     S1CON,#NSTA_NSTO_AA
                SJMP    END_I2C

                .ORG    0518H   ;code 18H SLA+W xmitted,ACK received,waiting for data
byte to send
                MOV     S1DAT,POWDIS    ;occurs when sending data to POWERC2
                MOV     S1CON,#NSTA_NSTO_AA
                SJMP    END_I2C

                .ORG    0520H   ;code 20H; SLA+W transmitted,NOTACK received
                MOV     S1CON,#NSTA_STO_AA
                MOV     DPTR,#ERRDEV
                SJMP    ERROR

                .ORG    0528H   ;code 28H; DATA transmitted, ACK received
                MOV     S1CON,#NSTA_STO_AA      ;this must be after sending display
byte to POWERC2
END_I2C:        POP     PSW
                RETI

                .ORG    0530H   ;code 30H; DATA transmitted, NOTACK received
                MOV     S1CON,#NSTA_STO_AA
                MOV     DPTR,#ERRDATA
                SJMP    ERROR

                .ORG    0538H   ;code 38H; Arbitration lost
ERR_MM:         MOV     S1CON,#NSTA_STO_AA
                MOV     DPTR,#ERRMM
                SJMP    ERROR

                .ORG    0540H   ;code 40H; SLA+R xmitted,ACK received,waiting for
read data to come
                JNB     POW_bit,MECH40
                MOV     S1CON,#NSTA_NSTO_NAA
                SJMP    END_I2C

                .ORG    0548H   ;code 48H; SLA+R transmitted,NOTACK received
                MOV     S1CON,#NSTA_STO_AA
                MOV     DPTR,#ERRDEV
```

```
                    SJMP    ERROR

                    .ORG    0550H   ;code 50H; DATA received, ACK returned
                    SJMP    STOREDATA50

                    .ORG    0558H
;for code 58H; DATA received, NOTACK returned,last data to read
; either 4th byte from MECHAND or only byte from POWERC
                    JB      POW_bit,POW58
                    SJMP    MECH58

ERROR:              SETB    SHUT_REQ_bit
                    SJMP    END_I2C

POW58:              MOV     POWKEY,S1DAT
                    SETB    POWKEY.3
                    MOV     S1CON,#STA_NSTO_AA
                    CLR     SLA_ADD.0
                    CLR     POW_bit
                    POP     PSW
                    RETI

MECH40:             MOV     S1CON,#NSTA_NSTO_AA
                    MOV     BYTECOUNT,#4
                    SJMP    END_I2C


STOREDATA50:        DJNZ    BYTECOUNT,STORE1AND2
;it's the 3rd byte of 4 from the MECHAND...texture (in the end!)
                    MOV     TEXTURE,S1DAT
                    MOV     S1CON,#NSTA_NSTO_NAA
                    SJMP    END_I2C

;it's one of the first 2 bytes from MECHAND, so convert n store it
STORE1AND2:         PUSH    ACC
                    MOV     A,S1DAT
                    MOV     S1CON,#NSTA_NSTO_AA
                    PUSH    B
                    DJNZ    BYTECOUNT,SBYTE1
                    INC     BYTECOUNT
                    JB      STEP_TEST_bit,BYPASS
                    MOV     B,CON_FACT5     ;2nd byte from MECHAND
                    MUL     AB
                    MOV     TSTORE5,B       ;store UNITS
                    MOV     (TSTORE5 + 1),A ;store POINTS
BYPASS:             POP     B
                    POP     ACC
                    POP     PSW
                    RETI
SBYTE1:             MOV     B,CON_FACT4
                    MUL     AB
                    MOV     TSTORE4,B
                    MOV     (TSTORE4 + 1),A
                    POP     B
                    POP     ACC
                    POP     PSW
                    RETI

MECH58:             PUSH    ACC
                    MOV     A,S1DAT
                    SETB    I2C_FIN_bit
                    MOV     S1CON,#NSTA_STO_AA
                    PUSH    B
                    MOV     B,CON_FACT7     ;4TH byte from MECHAND
                    MUL     AB
                    MOV     TSTORE7,B       ;store UNITS
                    MOV     (TSTORE7 + 1),A ;store POINTS
                    POP     B
                    POP     ACC
                    POP     PSW
                    RETI

ERRSTA:             .DB     LSTARTB         ;set DD address to 20
                    .DB     '*',32,'I',5
                    .TEXT   "C Bus Dead *"
                    .DB     LNXTLNE         ;set DD address to 64 (20 on 2nd line)
                    .TEXT   "Gone back to Bed"
                    .DB     LSHIFT

ERRMEC:             .DB     LSTARTB
                    .TEXT   "* MECHAND not  *"
                    .DB     LNXTLNE
                    .TEXT   "*    answering *"
```

```
                    .DB     LSHIFT

ERRPOW:             .DB     LSTARTB
                    .TEXT   "* POWERSEE not *"
                    .DB     LNXTLNE
                    .TEXT   "*    answering *"
                    .DB     LSHIFT

ERRBUS:             .DB     LSTARTB
                    .DB     '*',32,'I',5
                    .TEXT   "C Bus error "
                    .DB     LNXTLNE
                    .TEXT   "you touch owt? *"
                    .DB     LSHIFT

ERRMM:              .DB     LSTARTB
                    .TEXT   "Multiple master "
                    .DB     LNXTLNE
                    .TEXT   "you do it then!!"
                    .DB     LSHIFT

ERRDATA:            .DB     LSTARTB
                    .DB     'I',5
                    .TEXT   "C Data Error  "
                    .DB     LNXTLNE
                    .TEXT   "F*** know's why!"
                    .DB     LSHIFT

ERRDEV:             .DB     LSTARTB
                    .TEXT   "!Missing Device!"
                    .DB     LNXTLNE
                    .TEXT   "it Vapourized???"
                    .DB     LSHIFT
```

ROUTINE - LCDSUBS.ASM 29/9/94

```
;////////////////////////////////////////////
;//           LCD RUN-CODE SUBROUTINES          //
;//                version A.10                 //
;//             by Steve Lawther               //
;////////////////////////////////////////////
;
.TITLE  "LCD SUBROUTINES"

#DEFINE         SEND_TO_LCD     SETB    P4.2\   CLR     P4.2

LCDINIT:                MOV     R2,#40          ;wait approx. 15mS
                        ACALL   LONGWAIT
                        MOV     P4,#00110000B
                        SEND_TO_LCD
                        MOV     R2,#11          ;wait approx. 4.2ms
                        ACALL   LONGWAIT
                        SEND_TO_LCD             ;try again. p4 set to 00110000b
                        MOV     R3,#67
                        DJNZ    R3,$            ;wait 100uS more
                        SEND_TO_LCD             ;try again. p4 still set to 00110000b
                        MOV     R3,#34          ;wait 50uS more
                        DJNZ    R3,$
                        MOV     P4,#00100000B
                        SEND_TO_LCD             ;and tell it to change to 4-bit mode
                        MOV     R3,#34          ;wait 50uS more
                        DJNZ    R3,$
;and set Function set...now in 4bit mode, so 2 send's are required
                        MOV     P4,#00100000B
                        SEND_TO_LCD             ;high nibble
                        MOV     P4,#10000000B
                        SEND_TO_LCD             ;low nibble
                        MOV     R3,#34
                        DJNZ    R3,$            ;wait 50uS more
;and Clear Display...00000001b
                        MOV     P4,#00000000B
                        SEND_TO_LCD             ;high nibble
                        MOV     P4,#00010000B
                        SEND_TO_LCD             ;low nibble
                        MOV     R2,#6           ;wait 2mS
                        ACALL    LONGWAIT
;and DISPLAY ON...00001111b
                        MOV     P4,#00000000B
                        SEND_TO_LCD             ;high nibble
                        MOV     P4,#11000000B
                        SEND_TO_LCD             ;low nibble
                        MOV     R3,#67
                        DJNZ    R3,$            ;wait 100uS more
                        MOV     P4,#11111000B   ;D7-4 high for i/p,RS low,R/W high
                        SETB    P4.2
                        JB      P4.7,LCD_NOT_PRES
                        CLR     P4.2
                        SEND_TO_LCD             ;dump 2nd nibble
                        SETB    LCD_PRES_bit    ;LCD present
LCD_NOT_PRES:           RET

                        RET

;simply waits approx. R2*384uS
LONGWAIT:               DJNZ    R3,$
                        DJNZ    R2,LONGWAIT
                        RET

STARTLCDROM:            MOV     LCD_NEXT_CHR,#0
STARTLCDRAM:            SETB    LCD_RS_bit
                        SETB    LCD_DRDY_bit
                        RET

;*********************************************************************
;
;Routine:- LCD DRIVER
;*********************
;
;Called by:- any routine sending stuff to LCD
;Origin:- arbitary
;
;Requires:-
;       DPTR    set to start of character data for ROM messages
;       LCD_NEXT_CHR    offset address of chr to be displayed
;       LCD_PRES_bit    set to indicate LCD is present or possibly present
;       LCD_DRDY_bit    set to indicate new write to LCD required
;
;Uses:-
;       R2      use as temp. store
;       A       accumulator
;       C       carry bit
;       R1 (BANK?) offset address of chr to be displayed
;       TEMP_RS_bit     reset indicates this byte only is an instruction
;       LCD_RS_bit      reset indicates next codes are instructions
;
```

```
;       P4 consists of '7654 R/w EN sp RS'
;
LSTARTB .EQU     00010011B
LSTART  .EQU     00010001B
LTOGRS  .EQU     00010101B
LNXTLNE .EQU     00010111B
LFINISH .EQU     00011001B
LSHIFT  .EQU     00011011B

LCD_DRIVER:             MOV     C,LCD_DRDY_bit  ;check there's a message and
                        ANL     C,LCD_PRES_bit  ;a display to show it on
                        JNC     SOD_LCD         ;if not sod this!
;check LCD is ready for data first
CHECK_LCD_RDY:          MOV     P4,#11111000B   ;D7-4 high for i/p,RS low,R/W high
                        SETB    P4.2
                        JNB     P4.7,LCD_READY  ;LCD ready for next chr
                        MOV     R2,P4
                        CLR     P4.2
                        SEND_TO_LCD             ;dump 2nd nibble if there is one
                        CJNE    R2,#11111100B,LCD_BUSY
                        CLR     LCD_PRES_bit    ;no LCD present
SOD_LCD:                CLR     LCD_DRDY_bit
LCD_BUSY:               RET                     ;(20 cycles)

LCD_READY:              CLR     P4.2         ;ok LCD present'n'waiting, so
                        SEND_TO_LCD          ;dump 2nd nibble, and send a chr
GETNXTCHR:              MOV     R1,LCD_NEXT_CHR
                        MOV     A,@R1
                        JB      LCD_NXT_CHR.7,NOT_ROM   ;jump if it's a message in RAM
                        MOV     A,R1            ;if a message from ROM
                        MOVC    A,@A+DPTR       ;get chr
NOT_ROM:                INC     LCD_NEXT_CHR
                        MOV     R2,A
                        MOV     C,LCD_RS_bit
                        MOV     TEMP_RS_bit,C
                        ANL     A,#11110001B    ;note that number and'ed is 11110001b
;and not 11110000b (last bit is wiped by the MOV RS below)- this is so the
;display/cursor shift command can get past code-trap below if it has a 0 last.
;also CG data shouldn't start 0001 -as 1st 3 bits are waste, they can be high
                        CJNE    A,#00010001B,NOTCODE    ;(29cycles)
; Stuff for codes etc just here !!!!!!
                        CJNE    R2,#LTOGRS,NOT_TOGGLE
                        CPL     LCD_RS_bit
                        SJMP    GETNXTCHR               ;(34cycles)
NOT_TOGGLE:             CLR     TEMP_RS_bit     ;next send is gonna be an instruction
                        JC      STARTISH
                        CJNE    R2,#LNXTLNE,NOT_NXTLNE
NXTLNE:                 MOV     A,#11000000B
                        SJMP    TYPE_AORB
NOT_NXTLNE:             CJNE    R2,#LSHIFT,FINISH
                        MOV     DPTR,#SHIFTIT
                        MOV     LCD_NEXT_CHR,#0
                        SJMP    GETNXTCHR
FINISH:                 JB      LCD_TYPEB_bit,SOD_LCD
                        MOV     A,#00000010B    ;Display/cursor home
                        CLR     LCD_DRDY_bit
                        SJMP    TYPEA
STARTISH:               CLR     LCD_TYPEB_bit
                        CJNE    R2,#LSTARTB,START
STARTB:                 SETB    LCD_TYPEB_bit
START:                  MOV     A,#10000000B    ;start of first line
TYPE_AORB:              JNB     LCD_TYPEB_bit,TYPEA
                        ADD     A,#00010100B    ;move to 20'th chr
TYPEA:                  MOV     R2,A
                        ANL     A,#11110000B
NOTCODE:                MOV     P4,A
                        MOV     C,LCD_RS_bit
                        ANL     C,TEMP_RS_bit
                        MOV     P4.0,C
                        SEND_TO_LCD             ;send first nibble
                        MOV     A,R2
                        SWAP    A
                        ANL     A,#11110000B
                        MOV     P4,A
                        MOV     P4.0,C
                        SEND_TO_LCD             ;send second nibble
                        SETB    TEMP_RS_bit
                        RET                     ;(51cycles)

SHIFTIT:                .DB     LTOGRS
                        .DB     18H,18H,18H,18H,18H
                        .DB     18H,18H,18H,18H,18H
                        .DB     18H,18H,18H,18H,18H
                        .DB     18H,18H,18H,18H,18H,LFINISH
```

ROUTINE - PWMSUBS.ASM 09/04/94

```
;LIMITS

;nearest possible is 16kHz using PWMP=1 (PWMP=0 gives 32kHz). PWM0 &1 set to
;00 by reset
PWMINIT:        MOV     PWMP,#01H
                MOV     PWM0,#0
                MOV     PWM1,#0
                MOV     MAXPOWER_BG,#MAXPWRBANG
                MOV     MAXPOWER_PR,#MAXPWRPROP
                RET



OUTOFLIMITSH:   MOV     DPTR,#ERRLIMH
OOL:            MOV     PWM0,#0
                SETB    SHUT_REQ_bit
                RET
OUTOFLIMITSO:   MOV     DPTR,#ERROVER
                SJMP    OOL
OUTOFLIMITSM:   MOV     DPTR,#ERRLIMM
                SJMP    OOL
;time to check limits....first tstore0 (=T hand/peltier) min 0, max 45
PWMDRIVE:       MOV     A,TSTORE0
                ;JZ      OUTOFLIMITSH     ;a bit daft realy, but so wot!
                CLR     C
                SUBB    A,#46
                JNC     OUTOFLIMITSH
;Thp ok, now tstore1 (=T finger) for now just check less than 40
                MOV     A,TSTORE1
                CLR     C
                SUBB    A,#41
                JNC     OUTOFLIMITSH
;Thf ok, now tstore2 (T peltier heatsink) check less than 64
                MOV     A,TSTORE2
                ANL     A,#11000000B
                JNZ     OUTOFLIMITSO
;Tp ok, ignore tstore3 - not used
;       tstore4 (T mechand/heater) check less than 48
                MOV     A,TSTORE4
                CLR     C
                SUBB    A,#48
                JNC     OUTOFLIMITSM
;Tmh ok, now for tstore5( T mechand touch) check BELOW 64
;
                MOV     A,TSTORE5
                ANL     A,#11000000B
                JNZ     OUTOFLIMITSH
;13/12/94 commented out below to replace with above, when check was changed from 120 to 64
;               CLR     C
;                SUBB    A,#121
;                JNC     OUTOFLIMITSM

;limits check over....now to calc pwm numbers, if not in test
PWM_DRIVER:     JB      TEST_bit,ENDPWM
;THE FORMULA:-      (brackets show order of calculation)
;               (((T1 + T5) - T6) + offset ) - T0
;  ie ((( Tskin + Tmtouch ) - Tmsteady ) + Toffset) - Tpeltier
;note that this maths needs a full check to cover all possiblities

                MOV     A,(TSTORE1 + 1)
                ADD     A,(TSTORE5 + 1)
                MOV     (TEMPT + 1),A
                MOV     A,TSTORE1
                ADDC    A,TSTORE5
                MOV     TEMPT,A

;if c isn't zero anyway,help  ( <63 + <63, therefore not greater than 255)
                MOV     A,(TEMPT + 1)
                SUBB    A,(TSTORE6 + 1)
                MOV     (TEMPT + 1),A
                MOV     A,TEMPT
                SUBB    A,TSTORE6
                MOV     TEMPT,A
                JNC     GTHANZERO
                MOV     A,#1
                SJMP    OHDEAR          ;less than 0 deg C
GTHANZERO:      JNB     OFFSET_bit,NOOFFSET

;Time to add the offset - as offset is a 2's compliment signed no., just add
                MOV     A,(TEMPT + 1)
```

```
                ADD     A,(OFFSET + 1)
                MOV     (TEMPT + 1),A
                MOV     A,TEMPT
                ADDC    A,OFFSET
                MOV     TEMPT,A
;now subtract Tpeltier to find difference between actual and required
NOOFFSET:       MOV     A,(TEMPT + 1)
                SUBB    A,(TSTORE0 + 1)
                MOV     (TEMPT + 1),A
                MOV     A,TEMPT
                SUBB    A,TSTORE0
OHDEAR:         MOV     TEMPT,A
;if C is set it means the peltier has to cool
                MOV     PWM0,#0
                JC      COOLDOWN
;if within 2 deg c, use proportional control, otherwise bang bang
                MOV     PWM1,#0         ;set to heat
                MOV     A,TEMPT
                SUBB    A,#2
                JC      HEATPROP
HEATBANG:       MOV     PWM0,MAXPOWER_BG
ENDPWM:         RET

HEATPROP:       MOV     A,TEMPT
                RRC     A               ;gets LSB of temp units in C
                MOV     A,(TEMPT+1)
                RRC     A               ;number in A between 0 and 255
;commented put again 13/12/94 to add new code below.
;;commented out 27/3/94 as powermax is now FFhex,
;;back in 22/4/94 as unit unstable at FFhex proportional
;                CLR     C
;                RRC     A               ;number in A between 0 and 127
;                MOV     PWM0,A
;new code below
                MOV     B,MAXPOWER_PR
                MUL     AB
;B now contains the power value
;New bit here 19/2/95 for heat-pump differenece adjustment
                ACALL   VALUEADJ
                CLR     C
                JB      ACC.7,NEGHEATADJ          ;if neg adjust no
                XCH     A,B             ;just so that the adj is subb from the heat value
                SUBB    A,B        ;SUB heat pump adjust value
                JNC     ENDHEATPROP
                MOV     PWM1,#255   ;cool, not heat
                CPL     A

ENDHEATPROP:    MOV     PWM0,A
                RET

NEGHEATADJ:     XCH     A,B
                SUBB    A,B
                JC      ENDHEATPROP
                MOV     A,#255            ;overflowed 255
                SJMP    ENDHEATPROP

;13/12/94 THIS BIT OF CODE had a cpl a missing. It's possibly still not right (for one,
it's
; missing an inc a)
COOLDOWN:       MOV     PWM1,#255       ;set to cool
                MOV     A,TEMPT
                ADD     A,#2
                JNC     COOLBANG
                MOV     A,TEMPT
                CPL     A
                RRC     A
                MOV     A,(TEMPT+1)
                CPL     A
                RRC     A
; see comments above 27/3/94    & 13/12/94
;                CLR     C
;                RRC     A
;                MOV     PWM0,A

                MOV     B,MAXPOWER_PR
                MUL     AB
;B now contains the power value
;new 18/2/95
                ACALL   VALUEADJ
                JB      ACC.7,NEGCOOLADJ          ;if neg adjust no

                ADD     A,B        ;add heat pump adjust value
                JNC     ENDCOOLPROP
```

```
                MOV     A,#255

ENDCOOLPROP:    MOV     PWM0,A
                RET

NEGCOOLADJ:     ADD     A,B
                JC      ENDCOOLPROP
                CPL     A               ;HEATING NOT COOLING
                MOV     PWM1,#0
                SJMP    ENDCOOLPROP


COOLBANG:       MOV     PWM0,MAXPOWER_BG
                RET

VALUEADJ:       MOV     A,(TSTORE0 + 1)
                SUBB    A,(TSTORE2 + 1) ;To get any carry required

                MOV     A,TSTORE0
                SUBB    A,TSTORE2
                JNC     VALNOTNEG
                ADD     A,#32           ;SHIFT NEGS TO POS
                JC      NEGWITHIN32
                MOV     A,#0            ;The difference peltier to heatsink is >-32degC so
limit
NEGWITHIN32:    ACALL   VALADJLOOKUP
                RET

VALNOTNEG:      SUBB    A,#32
                JC      POSWITHIN32
                MOV     A,#0
POSWITHIN32:    ADD     A,#64
                ACALL   VALADJLOOKUP
                RET

VALADJLOOKUP    INC     A
                MOVC    A,@A+PC
                RET

;neg deg C indicates heatsink hotter that peltier, therefore neg adjust value is to cool
;Values in this table are between -127 and +128
;                       -32,-31,-30,-29,-28,-27,-26,-25,-24,-23,-22,-21,-20,-19,-18,-17
VALADJTABLE:    .DB      49, 47, 45, 43, 41, 39, 37, 35, 33, 31, 29, 27, 25, 24, 23, 21
;                       -16,-15,-14,-13,-12,-11,-10, -9, -8, -7, -6, -5, -4, -3, -2, -1
                .DB      20, 19, 18, 16, 15, 14, 13, 11, 10,  9,  8,  6,  5,  4,  3,  1
;                         0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15
                .DB       0,  0,255,255,254,254,254,253,253,252,252,251,251,251,250,250
;                        16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32

                .DB     249,249,249,248,248,247,247,246,246,246,245,245,244,244,244,243,243

ERROVER:
                .DB     LSTARTB
                .TEXT   "Peltier OVERHEAT"
                .DB     LNXTLNE
                .TEXT   "..gone for 11's "
                .DB     LFINISH

ERRLIMM:        .DB     LSTARTB
                .TEXT   "M Limit exceeded"
                .DB     LNXTLNE
                .TEXT   "now were cooking"
                .DB     LFINISH

ERRLIMH:        .DB     LSTARTB
                .TEXT   "H Limit exceeded"
                .DB     LNXTLNE
                .TEXT   "now were cooking"
                .DB     LFINISH
```

ROUTINE - RS232SUB.ASM 08/04/94

```
;////////////////////////////////////////////////
;//           RS232 subroutines                //
;//                version C - 29/9/94         //
;//             by Steve Lawther               //
;////////////////////////////////////////////////
;*********************************************************************
;Routine:- Initiatize RS232
;*************************
;
;Setup RS232, so that there is an interrupt generated if a byte is sent
; to the microC from a PC
;setup= MODE1,stop must be 1,enable reception, TB8=0, RB8=0, TI=0, RI=0
;15/12/94 - stop set to anything (bit 5 low), due to timing differences between
;kit and PC getting too great by the stop bit. (ugh!)
;**** to be changed back if able to tweak PC baud rate ****
INITSERIAL:     MOV     S0CON,#01010000B
;timer1 setup - GATE off, TIMER selected, timer MODE 2; 8-bit,reloaded from TH1
                MOV     TMOD,#00100000B
;reload value; 256 - 17 for close to 2400 Baud
;equn of TH1 = 256 - (41667/baudrate) with SMOD = 0
                MOV     TH1,#0EEH ;changed from efh 2/7/93***ie now 18******
                SETB    TR1             ;start timer1
                ORL     IEN0,#10010000B ;enables UART, assume low priority
                SETB    PS0
                MOV     TXADDR,SERIALOUT
                ; this next line is not needed now
                ; MOV     (SERIALOUT +17),#1 ; to stop incorrect packet sync-in
                SETB    TI
                RET


;*********************************************************************
;Routine:- RS232 INTERRUPT
;*************************
;
;Priority:-  LOW
;
;Called by:- SIO0 INTERRUPT, either RI if the PC is requesting data for the
;               1st time, or TI if a byte has been sent, and ready for next
;
;Origin:- 0023H
;
;Requires:-     NOTHING
; stored byte:-
; store bit:-
;       RS232ON_bit     set to send data to PC
TEMPADDRS       .EQU    $
                .ORG    0023H
RS232INT:       PUSH    PSW
                PUSH    ACC
                LJMP    RSROUTINETEX

                .ORG    TEMPADDRS
;If its the 2nd of a 2 byte receive, then it jumps here
SECONDBYTE:     JB      BYTE2JUMP_bit,BYTE2JUMP
                JB      BYTE2OFST_bit,BYTE2OFFSET
                JB      BYTE2ABS_bit,BYTE2ABS
                JB      BYTE2TSS_bit,BYTE2TSS
                JB      BYTE2PPW_bit,BYTE2PPOWER
                JB      BYTE2BPW_bit,BYTE2BPOWER
                SJMP    END2BYTE

BYTE2PPOWER:    MOV     MAXPOWER_PR,A
                SJMP    END2BYTE

BYTE2BPOWER:    MOV     MAXPOWER_BG,A
                SJMP    END2BYTE

BYTE2TSS:       RRC     A
```

```
                JNC     NOTTSSQTR
                ORL     (TSTORE6 + 1),#01000000B
NOTTSSQTR:      RRC     A
                JNC     NOTTSSHLF
                ORL     (TSTORE6 + 1),#10000000B
NOTTSSHLF:      ANL     A,#00111111B
                MOV     TSTORE6,A
                MOV     SOUT_T6,A
                MOV     (SOUT_T6 + 1),(TSTORE6 + 1)
                SJMP    END2BYTE

BYTE2ABS:       SETB    STEP_TEST_bit   ;to stop I2C writing a new val of T5
                RRC     A
                MOV     (TSTORE5 + 1),#0  ;missing, added 7/6/95; causes uC ver
8.6
                JNC     NOTABSQTR
                ORL     (TSTORE5 + 1),#01000000B
NOTABSQTR:      RRC     A
                JNC     NOTABSHLF
                ORL     (TSTORE5 + 1),#10000000B
NOTABSHLF:      ANL     A,#00111111B
                MOV     TSTORE5,A
CLEAROFFSET:    CLR     OFFSET_bit
                SJMP    END2BYTE

BYTE2JUMP:      SETB    STEP_TEST_bit    ;to stop I2C writing a new val of T5
BYTE2OFFSET:    CLR     C               ;if number is positive to keep bits 6&7
low
                JNB     ACC.7,POSNUMBER
;Number is meant to be negative so 2's complement time
                CLR     ACC.7
                CPL     A
                INC     A
                SETB    C               ;as number is negative to keep bits 6&7
high
POSNUMBER:      JZ      CLEAROFFSET     ;if number is +/-0
                MOV     (OFFSET + 1),#0
                RRC     A
                JNC     NOTOFFQTR
                ORL     (OFFSET + 1),#01000000B
NOTOFFQTR:      MOV     C,ACC.7         ;keep sign bit at acc.7
                RRC     A
                JNC     NOTOFFHLF
                ORL     (OFFSET + 1),#10000000B
NOTOFFHLF:      MOV     OFFSET,A
                SETB    OFFSET_bit
END2BYTE:       MOV     FIRSTBYTE,#0
                SJMP    ENDRX

;*************************************************************************
;THIS IS THE START POINT OF THIS ROUTINE
;
;6 codes from the PC are handled:-  (BITS; S -sign, p -power, u -units, d
-after d.p)
;JUMP     J jump +/- X from current value (holds value)   2 byte - 2nd =
Suuuuudd B
;OFFSET   O put +/- offset on current value (no hold)     2 byte - 2nd =
Suuuuudd B
;Tsteady  T Sets value of Tsteadystate(TSTORE6)           2 byte - 2nd =
uuuuuudd B
;ABSOLUTE A Sets absolute value of TSTORE5 (holds value)  2 byte - 2nd =
uuuuuudd B
;POWER    P Set maximum power level (Proportional)        2 byte - 2nd =
pppppppp B
;BANGPWR  B Set maximum power level (Bang-Bang)           2 byte - 2nd =
pppppppp B
;SHOCK    S Sets TSTORE5 to 5'C; shock temp (holds value)  1 byte
;CLEAR    C Clears effects of Jump,Offset,Absolute & Shock 1 byte
;baud up  @ Ups baud rate to 4800                          1 byte
;*************************************************************************
RSROUTINETEMP:  JNB     RI,TXMORE
                CLR     RI
```

```
                MOV     A,S0BUF
                JB      SECOND_bit,SECONDBYTE   ;if it's second byte of a 2
byte cmd

                CJNE    A,#'C',NOTCLEAR
                CLR     STEP_TEST_bit
                CLR     OFFSET_bit
                SJMP    ENDRX

NOTCLEAR:       CJNE    A,#'S',NOTSHOCK
                SETB    STEP_TEST_bit
                MOV     TSTORE5,#5              ;shock treatment
                CLR     OFFSET_bit
                SJMP    ENDRX

NOTSHOCK:       CJNE    A,#'J',NOTJUMP
                ;SETB    BYTE2JUMP_bit
                MOV     FIRSTBYTE,#10000001B     ;indicate 2nd byte and JUMP
                SJMP    ENDRX

NOTJUMP:        CJNE    A,#'O',NOTOFFSET
                ;SETB    BYTE2OFST_bit
                MOV     FIRSTBYTE,#10000010B     ;indicate 2nd byte and OFFSET
                SJMP    ENDRX

NOTOFFSET:      CJNE    A,#'P',NOTPROPPOWER
                ;SETB    BYTE2POW_bit
                MOV     FIRSTBYTE,#10000100B     ;indicate 2nd byte and
Proportional POWER
                SJMP    ENDRX

NOTPROPPOWER:   CJNE    A,#'B',NOTBANGPOWER
                ;SETB    BYTE2POW_bit
                MOV     FIRSTBYTE,#10100000B     ;indicate 2nd byte and
Bang-Bang POWER
                SJMP    ENDRX

NOTBANGPOWER:   CJNE    A,#'A',NOTABSOLUTE
                ;SETB    BYTE2ABS_bit
                MOV     FIRSTBYTE,#10001000B     ;indicate 2nd byte and
ABSOLUTE
                SJMP    ENDRX

NOTABSOLUTE:    CJNE    A,#'T',NOTTSTEADY        ;not steady state temp
                ;SETB    BYTE2TSS_bit
                MOV     FIRSTBYTE,#10010000B     ;indicate 2nd byte and STEADY
                SJMP    ENDRX

;new 13/12/94 - this bit of code is to change the baud rate to 4800 - it will
trash the
;serial port for a few millisecs, but sod it!
NOTTSTEADY:     CJNE    A,#'@',NOTBAUD            ;not baud change

                MOV     TH1,#0F7H ;***ie 9******
                SJMP    ENDRX
;new 29/3/95 - this bit of code is to setup texture data out at 16667baud

NOTBAUD:        CJNE    A,#'X',ENDRX               ;not change to texture out
                MOV     TH1,#0FBH  ;****IE 5*****
                ORL     PCON,#10000000B           ;Set SMOD to 1
                SETB    TEXRS232_bit
                MOV     TEXRSBYTE,#255
ENDRX:          JNB     TI,ENDRS232       ;check both TI & RI (just in case)
TXMORE:         CLR     TI
                MOV     PSW,#RBANK3
                MOV     R0,TXADDR
                MOV     A,@R0
                INC     TXADDR
                CJNE    R0,#(SERIALOUT + 22),NOTSYNC
; I got a feeling that interrupts should be temp switched off for this bit
; of code
;13/12/94 - the 2 Timer2 reg's are ok, as T2int is low priority
```

```
;except there is a possiblity of tmh2 having rolled over to 0 without t2ex
having been updated
                MOV    SERIALOUT,T2EX
                MOV    (SERIALOUT +1),TMH2
;13/12/94 - the next 6 data words need to be read without any INTERRUPTion!
(for acccuracy)
;the global interrupt enable bit is cleared
                CLR    EA
;the next 3 data words could be changed by ADCinit if it was enabled
                MOV    (SERIALOUT +2),TSTORE0
                MOV    (SERIALOUT +3),(TSTORE0 +1)
                MOV    (SERIALOUT +4),TSTORE1
                MOV    (SERIALOUT +5),(TSTORE1 +1)
                MOV    (SERIALOUT +6),TSTORE2
                MOV    (SERIALOUT +7),(TSTORE2 +1)
;the next 3 data words could be changed be I2Cinit if it was enabled
                MOV    (SERIALOUT +10),TSTORE4
                MOV    (SERIALOUT +11),(TSTORE4 +1)
                MOV    (SERIALOUT +12),TSTORE5
                MOV    (SERIALOUT +13),(TSTORE5 +1)
                MOV    (SERIALOUT +16),TSTORE7
;Enable interrupts again. Why one instruction early?...cos it'll be 1 other
instruction
; before interrupts will be serviced
                SETB   EA
                MOV    (SERIALOUT +17),(TSTORE7 +1)
;the power data word can only be changed by t2int and therefore will match the
temps
;recorded above!
                MOV    (SERIALOUT +18),PWM1
                MOV    (SERIALOUT +19),PWM0
                MOV    (SERIALOUT +20),STATUS
;send 255 to sync in data packet at PC end
                MOV     A,#255
                MOV     TXADDR,#SERIALOUT
NOTSYNC:        MOV     S0BUF,A
ENDRS232:       POP     ACC
                POP     PSW
                RETI
;*************************************
;If its the 2nd of a 2 byte receive IN TEXTURE MODE, then it jumps here
SECONDTEXBYTE:  JB      BYTE2TEX_bit,BYTE2TEXIN
                JB      BYTE2MAF_bit,BYTE2MANAF
                JB      BYTE2PCAF_bit,BYTE2PCAF
                SJMP    ENDTEX2BYTE
BYTE2PCAF:      MOV     PC_TEXFACTOR,A
                SJMP    ENDTEX2BYTE

BYTE2MANAF:     MOV     M_TEXFACTOR,A
                SJMP    ENDTEX2BYTE

BYTE2TEXIN:     MOV     PC_TEX,A

ENDTEX2BYTE:    MOV     FIRSTBYTE,#0
                SJMP    ENDTEXRX

;*************************************************************************
;THIS IS THE START POINT OF THIS ROUTINE
;
;3 codes from the PC are handled:-
;T - Texture data from the PC                    2bytes

;M - Manipulator amplification factor            2bytes

;% - PC amplification factor                     2bytes

;*************************************************************************
RSROUTINETEX:   JB      TEXRS232_bit,TEXROUTINE
                LJMP    RSROUTINETEMP
TEXROUTINE:     JNB     RI,TXTEXMORE
                CLR     RI
                MOV     A,S0BUF
```

```
                JB      SECOND_bit,SECONDTEXBYTE   ;if it's second byte of a 2
byte cmd

                CJNE    A,#'T',NOTTEXFROMPC
                MOV     FIRSTBYTE,#10000001B     ;indicate 2nd byte and texture
from PC
                SJMP    ENDTEXRX

NOTTEXFROMPC:   CJNE    A,#'M',NOTMANIAMPFACT
                MOV     FIRSTBYTE,#10000010B     ;indicate 2nd byte and MANIP
AMP FACTOR
                SJMP    ENDTEXRX

NOTMANIAMPFACT: CJNE    A,#'%',ENDTEXRX
                MOV     FIRSTBYTE,#10000100B     ;indicate 2nd byte and PC AMP
FACTOR
                SJMP    ENDTEXRX

;new 13/12/94 - this bit of code is to change the baud rate to 4800 - it will
trash the
;serial port for a few millisecs, but sod it!
;NOTTSTEADY:     CJNE    A,#'@',NOTBAUD            ;not baud change

;                MOV     TH1,#0F7H ;***ie 9******
;                SJMP    ENDRX
ENDTEXRX:       JNB     TI,ENDRS232       ;check both TI & RI (just in case)
TXTEXMORE:      CLR     TI
                JB      BYTE1GONE_bit, BYTE2GONE  ;If 2nd BYTE GONE don't write
anything TIL t8
                MOV     S0BUF,TEXOUTBYTE
                SETB    BYTE1GONE_bit
                POP     ACC
                POP     PSW
                RETI
BYTE2GONE       SETB    BYTE2GONE_bit
                POP     ACC
                POP     PSW
                RETI
```

ROUTINE - TEXSUBS.ASM 10/04/95

```
;////////////////////////////////////////////////////////////
;/                                                          /
;/       texural subroutine (28/3/95)                       /
;/                                                          /
;////////////////////////////////////////////////////////////

TEXINIT:        MOV     M_TEXFACTOR,#MANTEXF
                MOV     PC_TEXFACTOR,#PCTEXF
                RET


;Requires MANIP_TEX - texture data from manipulator
;         M_TEXFACTOR - manipulator texture factor
;         PC_TEX - texture data from PC
;         PC_TEXFACTOR - PC texture factor


TEXTUREDRIVE:   MOV     A,TEXTURE
                JNB     TEXRS232_bit, NOSERIALTEX
                JNB     BYTE2GONE_bit, NOSERIALTEX      ;last byte was not finished in time
                MOV     S0BUF,A         ;send texture byte
                ANL     TEXRSBYTE, #128 ;clear byte2gone and byte1gone
                MOV     TEXTURE,TMH2    ;Reuse texture byte for messing with t2 low
                ANL     TEXTURE,#00000011B ;to get timing signal for PC
NOSERIALTEX:    MOV     B,M_TEXFACTOR
                MUL     AB
                MOV     TEMPT, B
                MOV     A,PC_TEX
                MOV     B,PC_TEXFACTOR
                MUL     AB
                MOV     A,TEMPT
                ADD     A,B
                ANL     A,#11111100B
                ORL     A,TEXTURE
                RR      A
                RR      A
                MOV     TEXOUTBYTE,A    ;store texture out byte for sending - note top 2
                                        ;bits are timing
                ORL     A,#11000000B    ;To keep the I2C pins functioning!!!!!(restored
5/4/95)
                MOV     P1,A
                RET
```

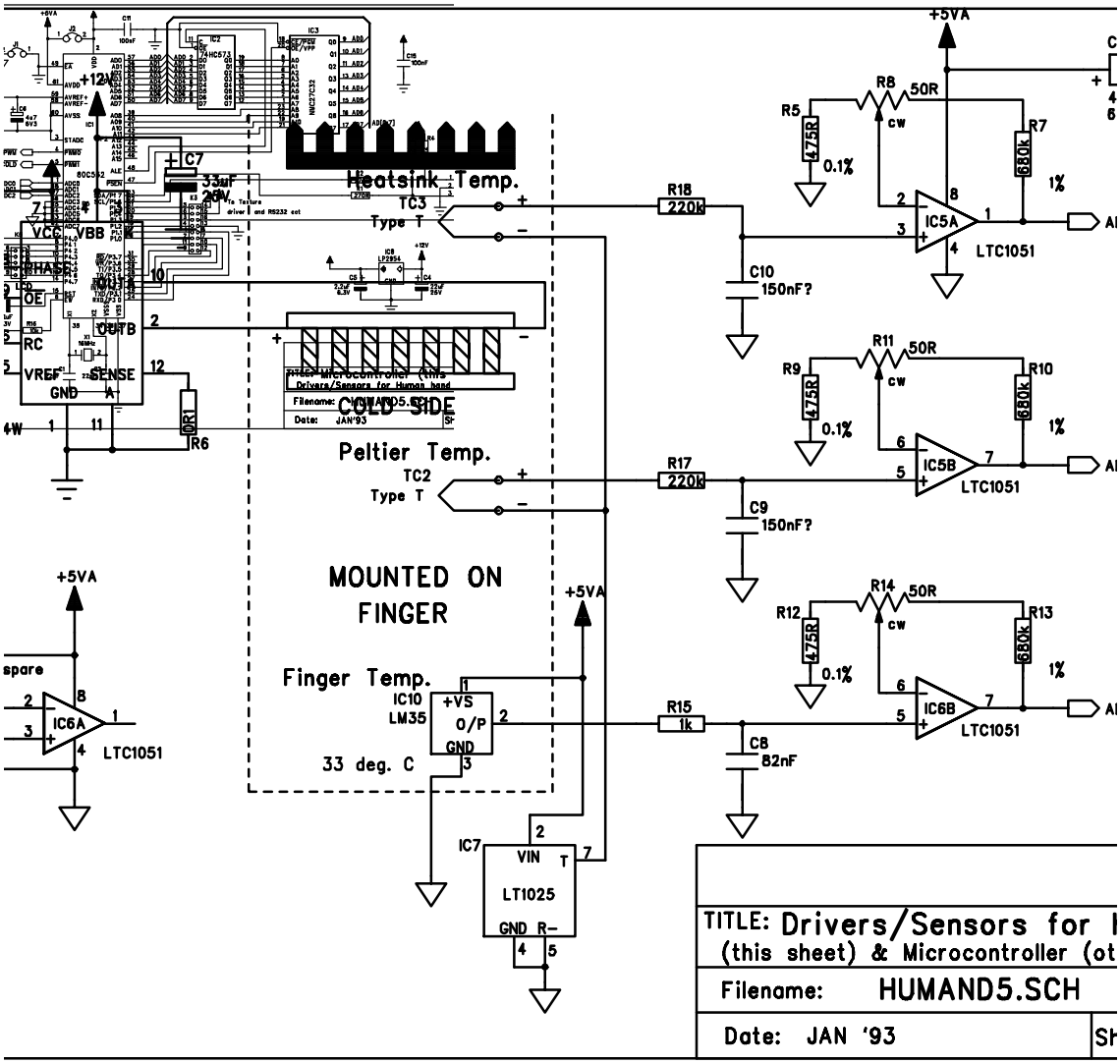
HEADER FILE - SFR.H 05/04/95

```
RBANK0          .EQU    000H
RBANK1          .EQU    008H
RBANK2          .EQU    010H
RBANK3          .EQU    018H
BNK1R0  .EQU    008H
BNK1R1  .EQU    009H
P0      .equ    080H    ;Port 0
SP      .equ    081H    ;Stack pointer
DPL     .equ    082H
DPH     .equ    083H
PCON    .equ    087H
TCON    .equ    088H
TMOD    .equ    089H
TL0     .equ    08AH
TL1     .equ    08BH
TH0     .equ    08CH
TH1     .equ    08DH
P1      .equ    090H    ;Port 1
S0CON   .equ    098H
S0BUF   .equ    099H
P2      .equ    0A0H    ;Port 2
IEN0    .equ    0A8H
P3      .equ    0B0H    ;Port 3
IP0     .equ    0B8H
P4      .EQU    0C0H
ADCON   .EQU    0C5H
ADCH    .EQU    0C6H
TM2IR   .EQU    0C8H
PSW     .equ    0D0H
S1CON   .EQU    0D8H
S1STA   .EQU    0D9H
S1DAT   .EQU    0DAH
ACC     .equ    0E0H    ;Accumulator
IEN1    .EQU    0E8H
TM2CON  .EQU    0EAH
TML2    .EQU    0ECH
TMH2    .EQU    0EDH
B       .equ    0F0H    ;Secondary Accumulator
IP1     .EQU    0F8H
PWM0    .EQU    0FCH
PWM1    .EQU    0FDH
PWMP    .EQU    0FEH

T2EX            .EQU    020H
STATUS          .EQU    021H
POWKEY          .EQU    023H
LCD_NEXT_CHR    .EQU    024H
SLA_ADD         .EQU    025H
FIRSTBYTE       .EQU    026H
TEXRSBYTE       .EQU    027H

IDLETIME        .EQU    030H
;IDLETIME2 IS 31H
POWDIS          .EQU    032H
TEXOUTBYTE      .EQU    033H
M_TEXFACTOR     .EQU    034H
PC_TEX          .EQU    035H
PC_TEXFACTOR    .EQU    036H

TSTORE0         .EQU    040H
TSTORE1         .EQU    042H
TSTORE2         .EQU    044H
TSTORE4         .EQU    048H
TSTORE5         .EQU    04AH
TSTORE6         .EQU    04CH
TSTORE7         .EQU    04EH
CON_FACT0       .EQU    050H
CON_FACT1       .EQU    051H
CON_FACT2       .EQU    052H
CON_FACT3       .EQU    053H
CON_FACT4       .EQU    054H
CON_FACT5       .EQU    055H
CON_FACT6       .EQU    056H
CON_FACT7       .EQU    057H
;JUNKA           .EQU    058H
MAXPOWER_PR     .EQU    058H
BYTECOUNT       .EQU    059H
TEXTURE         .EQU    05AH
TEMPT           .EQU    05BH
;tempt+1 equ 05Ch
```

```
HADD            .EQU    05DH
TXADDR          .EQU    05EH
MAXPOWER_BG     .EQU    05FH
SERIALOUT       .EQU    060H
SOUT_T6         .EQU    06EH
OFFSET          .EQU    07CH
;OFFSET+1 EQU 07DH
;QUEUE           .EQU    07DH
SUM             .EQU    07EH
;sum + 1 equ 07Fh
T_DISP          .EQU    080H
;QUEUE_MIN       .EQU    0C0H
;QUEUE_MAX       .EQU    0C9H

    ;Now some bit addresses
TR1             .EQU    08EH
RI              .EQU    098H
TI              .EQU    099H
EA              .EQU    0AFH    ;global interrupt enable
ACC.0           .equ    0E0H    ;Acc bit 0
ACC.1           .equ    0E1H    ;Acc bit 1
ACC.2           .equ    0E2H    ;Acc bit 2
ACC.3           .equ    0E3H    ;Acc bit 3
ACC.4           .equ    0E4H    ;Acc bit 4
ACC.5           .equ    0E5H    ;Acc bit 5
ACC.6           .equ    0E6H    ;Acc bit 6
ACC.7           .equ    0E7H    ;Acc bit 7
P1.6            .EQU    096H
P1.7            .EQU    097H
PS0             .EQU    0BCH
P4.0            .EQU    0C0H
P4.2            .EQU    0C2H
P4.4            .EQU    0C4H
P4.6            .EQU    0C6H
P4.7            .EQU    0C7H
SI              .EQU    0DBH
STO             .EQU    0DCH
STA             .EQU    0DDH
T2OV            .EQU    0CFH

T2EX.2          .EQU    002H
ADC_END_bit     .EQU    008H
I2C_FIN_bit     .EQU    009H
OFFSET_bit      .EQU    00AH
POW_bit         .EQU    00BH
STEP_TEST_bit   .EQU    00CH
TEST_bit        .EQU    00DH
SHUT_REQ_bit    .EQU    00EH
SHUT_ACK_bit    .EQU    00FH

LCD_PRES_bit    .EQU    010H
LCD_TYPEB_bit   .EQU    011H
LCD_RS_bit      .EQU    012H
TEMP_RS_bit     .EQU    013H
LCD_DRDY_bit    .EQU    014H
ADC_OVERLAP     .EQU    016H
I2C_OVERLAP     .EQU    017H

POWKEY.0        .EQU    018H
POWKEY.1        .EQU    019H
POWKEY.2        .EQU    01AH
POWKEY.3        .EQU    01BH

LCD_NXT_CHR.7   .EQU    027H
SLA_ADD.0       .EQU    028H

BYTE2JUMP_bit   .EQU    030H
BYTE2TEX_bit    .EQU    030H
BYTE2OFST_bit   .EQU    031H
BYTE2MAF_bit    .EQU    031H
BYTE2PPW_bit    .EQU    032H
BYTE2PCAF_bit   .EQU    032H
BYTE2ABS_bit    .EQU    033H
BYTE2TSS_bit    .EQU    034H
BYTE2BPW_bit    .EQU    035H
SECOND_bit      .EQU    037H

BYTE1GONE_bit   .EQU    038H
BYTE2GONE_bit   .EQU    039H
TEXRS232_bit    .EQU    03FH

.END
```

HEADER FILE - SSTVALUE.H 10/04/95

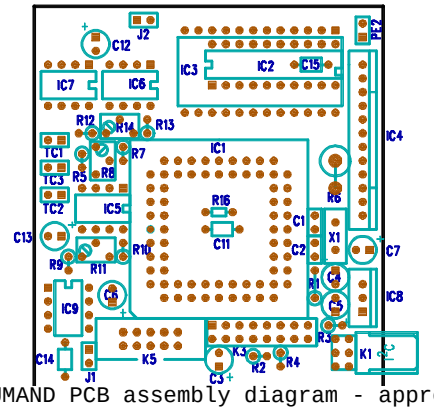
```
;The Steady state temp for MECHAND
STDYSTATETEMP           .EQU    31

CONVFACTOR0             .EQU    91      ;t
CONVFACTOR1             .EQU    40      ;t
CONVFACTOR2             .EQU    91      ;t
CONVFACTOR4             .EQU    54      ;tmheater
CONVFACTOR5             .EQU    59      ;tmtouch
CONVFACTOR7             .EQU    40      ;tmambient

MAXPWRPROP              .EQU    128     ;maximum proportional
                                        ;power limit
MAXPWRBANG              .EQU    255     ;maximum power (bang-bang)

MANTEXF                 .EQU    255     ;Manipulator texture factor
PCTEXF                  .EQU    0       ;PC texture factor

#DEFINE         VERNO   "9.3Texture"
```


+5VA
+12V
C7
33uF
20V
Heatsink Temp.
TC3
Type T
R18
220k
C10
150nF?
R5
475R
0.1%
R8
50R
cw
R7
680k
1%
IC5A
LTC1051
R11
50R
R9
475R
R10
680k
IC5B
LTC1051
R17
220k
C9
150nF?
COLD SIDE
Peltier Temp.
TC2
Type T
MOUNTED ON
FINGER
Finger Temp.
IC10
LM35
+VS
O/P
GND
33 deg. C
R15
1k
C8
82nF
R12
475R
R14
50R
R13
680k
IC6B
LTC1051
spare
IC6A
LTC1051
IC7
LT1025
VIN
T
GND R-
VCC
VBB
PHASE
OE
RC
VREF
SENSE
GND
OUTB
R6
10R
TITLE: Drivers/Sensors for
(this sheet) & Microcontroller (ot
Filename: HUMAND5.SCH
Date: JAN '93

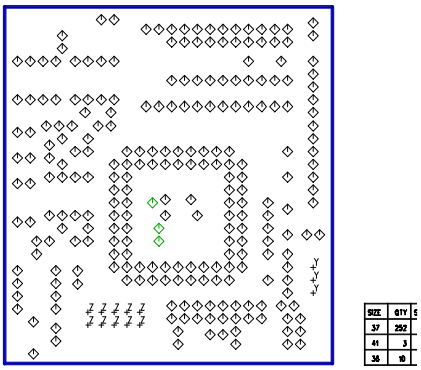
Bill Of Materials for HUMANDCD.SCH on Tue Jan 25 22:55:27 1994

Qty	Ref	Part Name	Part type	Description
1	IC2	74HC573		OCTAL D-TYPE TRANSPARENT LATCH
1	IC1	80C552		
2	C1-2	CAP0.1,22pF		RADIAL CAP BODY:3.6x2.3mm CENTRES:.2"
2	C9-10	CAPCC1206, 150nF?		SMD Capacitor, 1206 package (3.2x1.6)
1	C8	CAPCC1206,82nF		SMD Capacitor, 1206 package (3.2x1.6)
2	C3 C5	CAPR+,2.2uF		DECOUP CAP RADIAL BODY:.2 PITCH 0.08 or 0.1
1	C4	CAPR+,22uF		DECOUP CAP RADIAL BODY:.2 PITCH 0.08 or 0.1
1	C7	CAPR+,33uF		DECOUP CAP RADIAL BODY:.2 PITCH 0.08 or 0.1
3	C6 C12-13	CAPT\1\2,4u7	???	CAP TANTALUM RADIAL, 0.1" PIN SPACE, 0.2" DIAMETER
1	K1	CON\I2C	RS CODE 483-354	6 WAY (2x3) CONNECTOR
1	K3	CON\PCB\2X8		2x8 Way PCB Header
1	K5	HEADER\10V	ESD 077516D	5 X 2 PIN POLARIZED HEADER, 0.1" CENTERS
2	J1-2	JUMPER		JUMPER BLOCK, 2 PINS 0.1" SPACING
1	IC10	LM35	MAP UF52G	Precision Tempurature Sensor
1	IC8	LP2954	RS 265-178	LP2954IT Micropower Fixed 5V Reg
1	IC7	LT1025		
2	IC5-6	LTC1051		DUAL PRECISION CHOPPER STABILIZED OP-AMP WITH INT CAP
1	IC3	NMC27C32		8K X 8 BIT EPROM
1	PE2	PELT		Peltier Heat Pump 8V @1.8A
1	R6	R-W21,0R1		2.5W Power resistor 12.7x5.6mm
1	R16	R0.6W,10k		RES BODY:2.5mm CENTRES:0.4"
1	R15	R0.6W,1k		RES BODY:2.5mm CENTRES:0.4"
2	R17-18	R0.6W,220k		RES BODY:2.5mm CENTRES:0.4"
2	R1-2	R0.6W,270R		RES BODY:2.5mm CENTRES:0.4"
2	R3-4	R0.6W,2k2		RES BODY:2.5mm CENTRES:0.4"
3	R5 R9 R12	R0.6W,475R,0.1%		RES BODY:2.5mm CENTRES:0.4"
3	R7 R10 R13	R0.6W,680k,1%		RES BODY:2.5mm CENTRES:0.4"
1	IC9	REF-02CP	RS CODE No. 652-695 Maplin UL09K	5V PRECISION REFERENCE
3	C11 C14-15	SA10,100nF		MILITARY CAP AXIAL BODY:.150 X.100 CENTERS:.250
2	TC2-3	Tc-T		Type T thermocouple
1	IC4	UDN2954W		
3	R8 R11 R14	VRES\MT1/4,50R		1/4" MULTI TURN PRESET
1	X1	XTAL3,16MHz		CRYSTAL

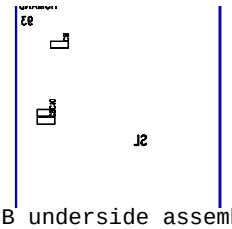
```
BOARD STATISTICS REPORT -- humand7.job -- Wed Jan 13 20:37:28 1993


Job Design Time:    19:42

Part Types:         25
Parts TopSide:      46    BottomSide:     3     Total: 49
Drilled pads:       262   Undrilled pads: 6     Total: 268
Thru vias :         12
Buried vias :       0

Signal Nets:        78
Connections Routed: 169  Partially 0    Unrouted: 0    Total: 169

Routed Connection Length (inches) X:   47.26    Y: 54.06    Total: 101.32
Unrouted Connection Length (inches) X: 0.00     Y: 0.00     Total: 0.00

Number of copper clearance errors:         0

Number of Routing Layers:                  2
Size of Board (square inches):             7.08
Equivalent IC count (1-IC/14 pins):        19
Board Density(boardsize/14pin-components): 0.37
```



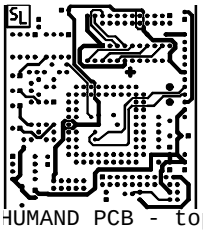
HUMAND PCB assembly diagram - approx. 2:1 Scale



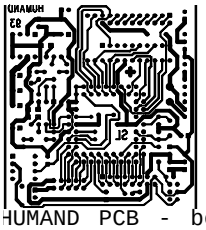
HUMAND PCB - drilling diagram - scale approx. 2:1



Humand PCB underside assembly diagram - approx. 1:1 Scale



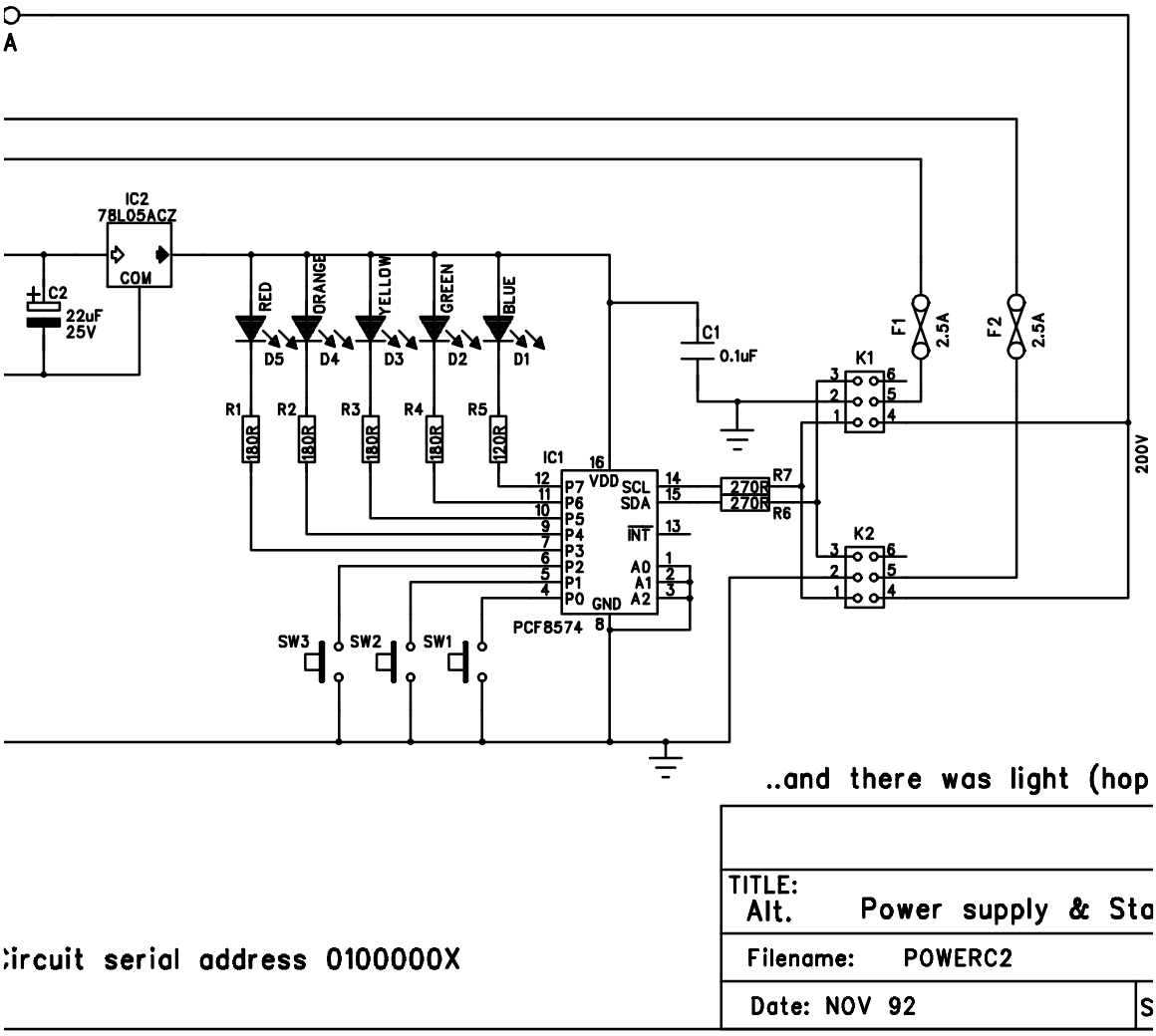
HUMAND PCB - top layer, approx. 1:1 scale (board 6.75h x 6.25w)



HUMAND PCB - bottom layer, view from top layer - approx. 1:1 scale (board 6.75h x 6.25w)

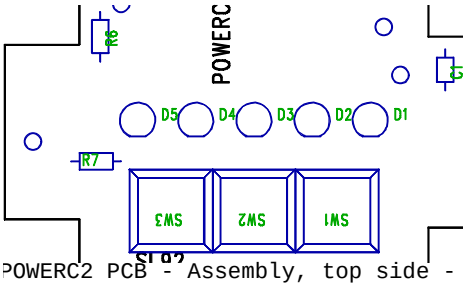
Board Modifications

Passive filters added to all analogue inputs.
Shield earthing point added to back of I²C conector gnd pin

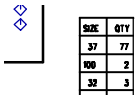

A
IC2
78L05ACZ
COM
C2
22uF
25V
RED
ORANGE
YELLOW
GREEN
BLUE
D5
D4
D3
D2
D1
C1
0.1uF
F1
2.5A
F2
2.5A
K1
K2
R1
R2
R3
R4
R5
180R
180R
180R
180R
120R
IC1
VDD
SCL
SDA
INT
A0
A1
A2
GND
P7
P6
P5
P4
P3
P2
P1
P0
PCF8574
270R
270R
R7
R6
SW3
SW2
SW1
200V
..and there was light (hop
TITLE:
Alt. Power supply & Sta
Filename: POWERC2
Date: NOV 92
S
ircuit serial address 0100000X

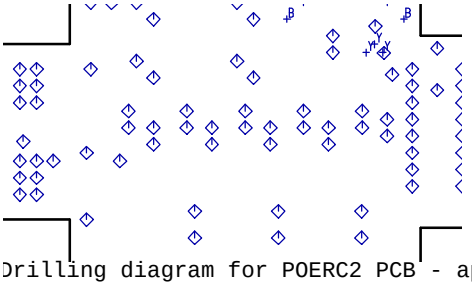
Bill Of Materials for POWERC2.SCH on Tue Jan 25 23:03:53 1994

Qty	Ref	Part Name	Part type	Description
1	D6	1N4001,???		1N4001 Recifier Diode
2	K3-4	4MMPLUG		WIRED CONNECTION TO POWER,4MM PLUG
1	IC2	5VREG		5V 100mA REGULATOR
1	C2	CAPR+,22uF		DECOUP CAP RADIAL BODY:.2 PITCH 0.08 or 0.1
2	K1-2	CON\I2C	RS 483-354	6 WAY (2x3) CONNECTOR
2	F1-2	FUSE\20\PCB, 2.5A		20mm FUSE + PCB MOUNTING CLIPS
1	F3	F\20\RS,50mA		20mm FUSE HOLDER CONNECT TO 4MM PLUG (brown)
1	D1	LED,BLUE	MAP UL89W RS 577-617	LIGHT EMITTING DIODE
1	D2	LED,GREEN		LIGHT EMITTING DIODE
1	D4	LED,ORANGE		LIGHT EMITTING DIODE
1	D5	LED,RED		LIGHT EMITTING DIODE
1	D3	LED,YELLOW		LIGHT EMITTING DIODE
1	IC1	PCF8574		Remote 8-bit I/O expander
1	R5	R0.6W,120R		RES BODY:2.5mm CENTRES:0.4"
4	R1-4	R0.6W,180R		RES BODY:2.5mm CENTRES:0.4"
2	R6-7	R0.6W,270R		RES BODY:2.5mm CENTRES:0.4"
1	C1	SA10,0.1uF		MILITARY CAP AXIAL BODY:.150 X.100 CENTERS:.250
3	SW1-3	SW\PCB\PUSH	????	PCB push-to-make switch (colour ???)

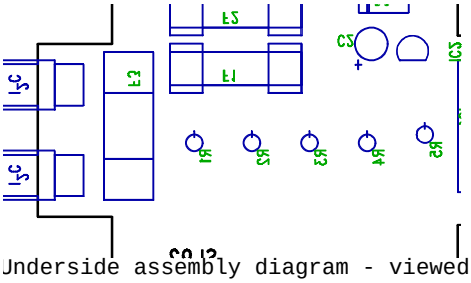


POWERC2 PCB - Assembly, top side - approx. 2:1

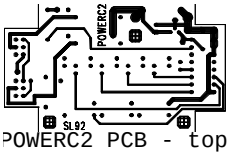




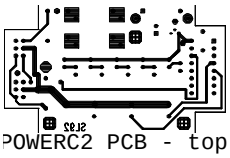
Drilling diagram for POERC2 PCB - approx. 2:1 scale.



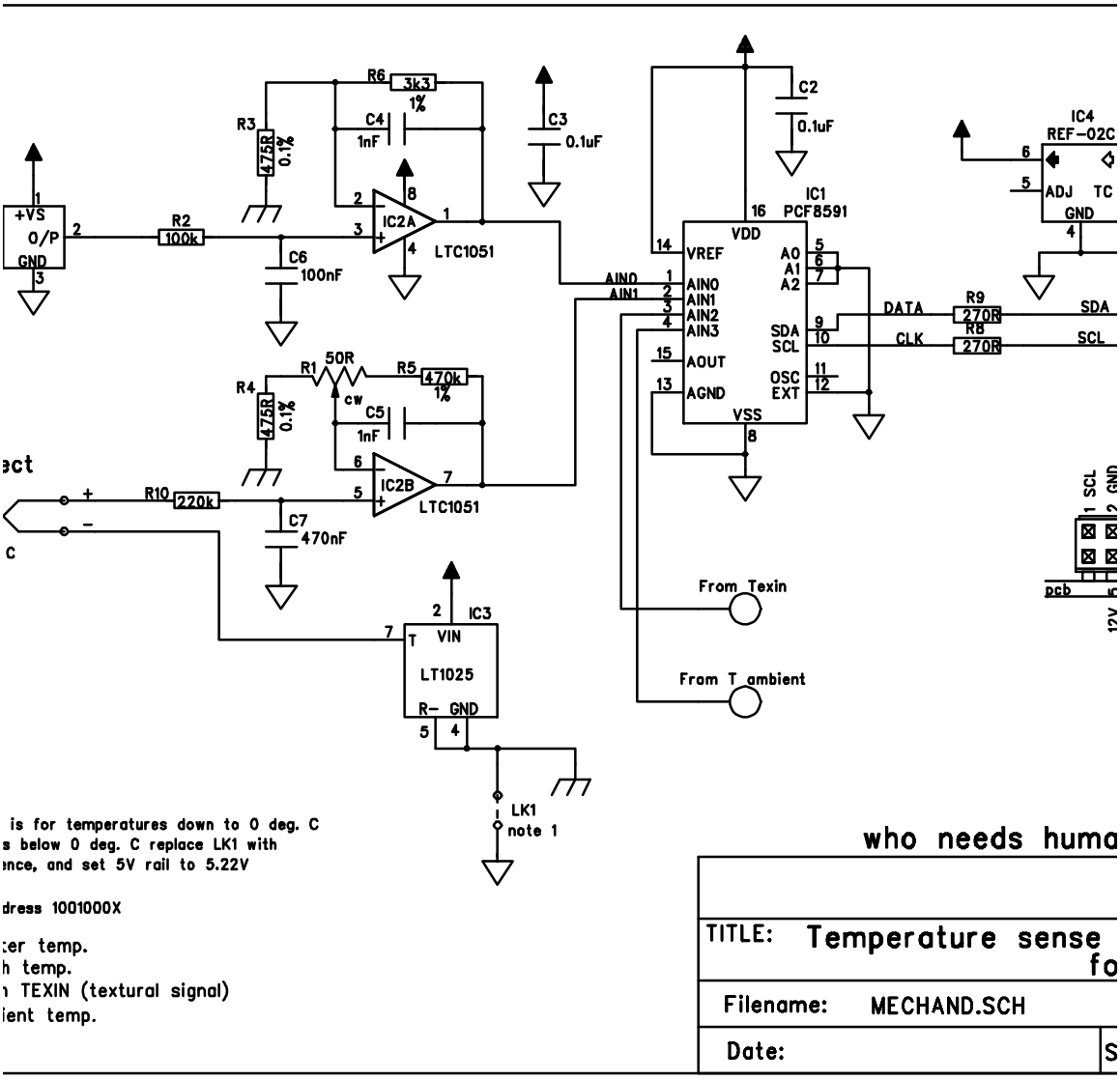
Underside assembly diagram - viewed from top - approx 2:1 scale



POWERC2 PCB - top layer, approx. 1:1 scale (board 4.25 h x 7.25w)

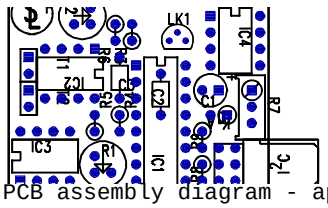


POWERC2 PCB - top layer, approx. 1:1 scale (board 4.25 h x 7.25w)

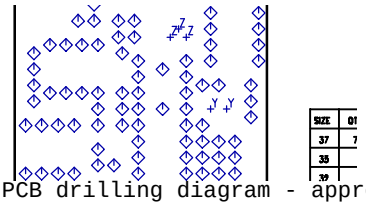

R6 3k3 1%
C4 1nF
R3 475R 0.1%
C3 0.1uF
C2 0.1uF
IC4 REF-02C
ADJ TC
GND
+VS
O/P
GND
R2 100k
IC2A
LTC1051
C6 100nF
IC1 PCF8591
VDD
VREF
AIN0
AIN1
AIN2
AIN3
AOUT
AGND
VSS
A0
A1
A2
SDA
SCL
OSC EXT
DATA
CLK
R9 270R
R8 270R
SDA
SCL
R1 50R
cw
R5 470k 1%
R4 475R 0.1%
C5 1nF
IC2B
LTC1051
R10 220k
C7 470nF
IC3
LT1025
VIN
R- GND
From Texin
From T ambient
LK1
note 1
is for temperatures down to 0 deg. C
s below 0 deg. C replace LK1 with
ance, and set 5V rail to 5.22V
dress 1001000X
ter temp.
h temp.
TEXIN (textural signal)
ient temp.
who needs huma
TITLE: Temperature sense
Filename: MECHAND.SCH
Date:

Bill Of Materials for MECHAND.SCH on Wed Jan 26 20:13:25 1994

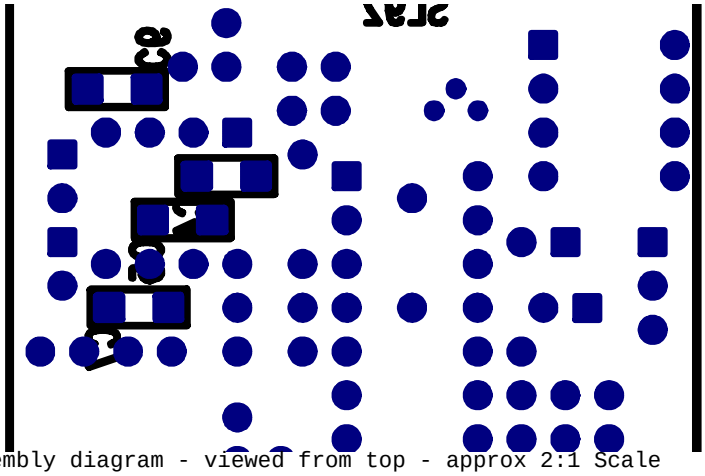
Qty	Ref	Part Name	Part type	Description
1	D1	1N4001,???		1N4001 Recifier Diode
1	C6	CAPCC1206, 100nF,5%		SMD Capacitor, 1206 package (3.2x1.6)
2	C4-5	CAPCC1206,1nF, 5%		SMD Capacitor, 1206 package (3.2x1.6)
1	C7	CAPCC1206, 470nF,5%		SMD Capacitor, 1206 package (3.2x1.6)
1	C1	CAPR+,33uF		DECOUP CAP RADIAL BODY:.2 PITCH 0.08 or 0.1
1	K1	CON\I2C	RS CODE 483-354	6 WAY (2x3) CONNECTOR
1	LK1	LINK		
1	IC5	LM35	MAP UF52G	Precision Tempurature Sensor
1	IC3	LT1025	MAC LT1025ACN8	
1	IC2	LTC1051	RS 263-835 MAC LTC1051CN8	DUAL PRECISION CHOPPER STABILIZED OP-AMP WITH INT CAP
1	IC1	PCF8591	ESD ?????? MAC PCF8591P	QUAD 8BIT A/D + D/A CONVERTER (I2C BUS)
1	R2	R0.6W,100k		RES BODY:2.5mm CENTRES:0.4"
1	R10	R0.6W,220k,1%		RES BODY:2.5mm CENTRES:0.4"
2	R8-9	R0.6W,270R,1%		RES BODY:2.5mm CENTRES:0.4"
1	R6	R0.6W,3k3,1%		RES BODY:2.5mm CENTRES:0.4"
1	R5	R0.6W,470k,1%		RES BODY:2.5mm CENTRES:0.4"
2	R3-4	R0.6W,475R,0.1%		RES BODY:2.5mm CENTRES:0.4"
1	IC4	REF-02CP	RS 652-695 MAP UL09K	5V PRECISION REFERENCE
1	C2	SA10,0.1uF	RS 126-590	MILITARY CAP AXIAL BODY:.150 X.100 CENTERS:.250
1	C3	SA10,0.1uF		MILITARY CAP AXIAL BODY:.150 X.100 CENTERS:.250
1	T2	TC-T		Type T thermocouple (FOIL)
1	R1	VRES\ST1/4,50R		1/4" ROUND, SINGLE TURN PRESET
1	K2	WIRECONN		wire connection (color-???)



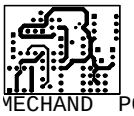
MECHAND PCB assembly diagram - approx 2:1 Scale



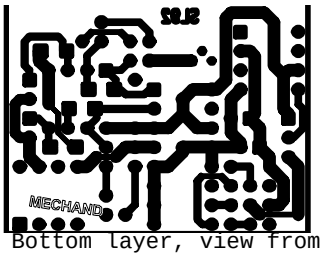
MECHAND PCB drilling diagram - approx 2:1 Scale



MECHAND PCB bottom assembly diagram - viewed from top - approx 2:1 Scale



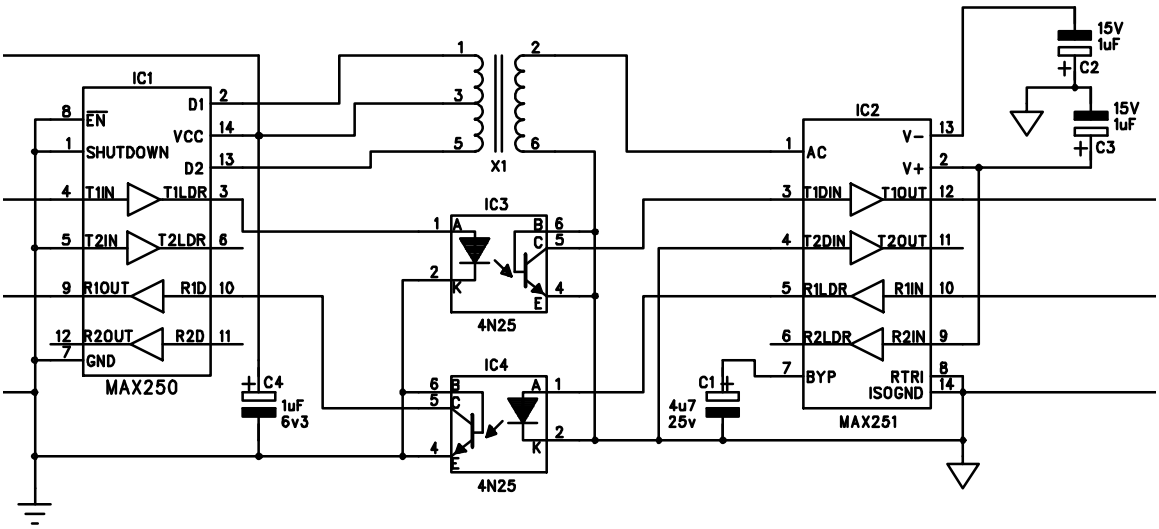
MECHAND PCB - Top layer, approx. 1:1 scale (board 2.9h x 3.9w)



Bottom layer, view from above.

Board Modifications

Filters added to each input
Presets shorted out

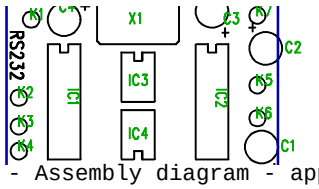


The Art of Communication

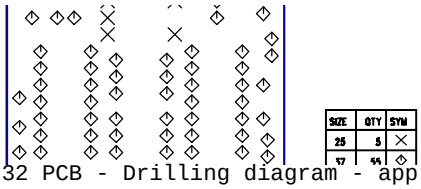
TITLE:	RS232 Connection t
Filename:	RS232CON
Date:	7 DEC 92

Bill Of Materials for RS232CON.SCH on Sat Jan 29 18:52:12 1994

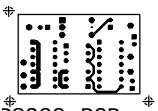
Qty	Ref	Part Name	Part type	Description
2	IC3-4	4N25	?	Opto-Transistor Isolator
3	C2-4	CAPT\1\2,1uF	???	CAP TANTALUM RADIAL, 0.1" PIN SPACE, 0.2" DIAMETER
1	C1	CAPT\1\2,4u7	???	CAP TANTALUM RADIAL, 0.1" PIN SPACE, 0.2" DIAMETER
1	IC1	MAX250	RS 659-618	+5V Powered Isolated RS232
1	IC2	MAX251	RS 659-624	+5V Powered Isolated RS232
7	K1-7	WIRECONN		wire connection (color-???)
1	X1	XFR-ISO	RS 196-460	Isolating interface transformer



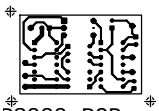
RS232 PCB - Assembly diagram - approx. 2:1 scale



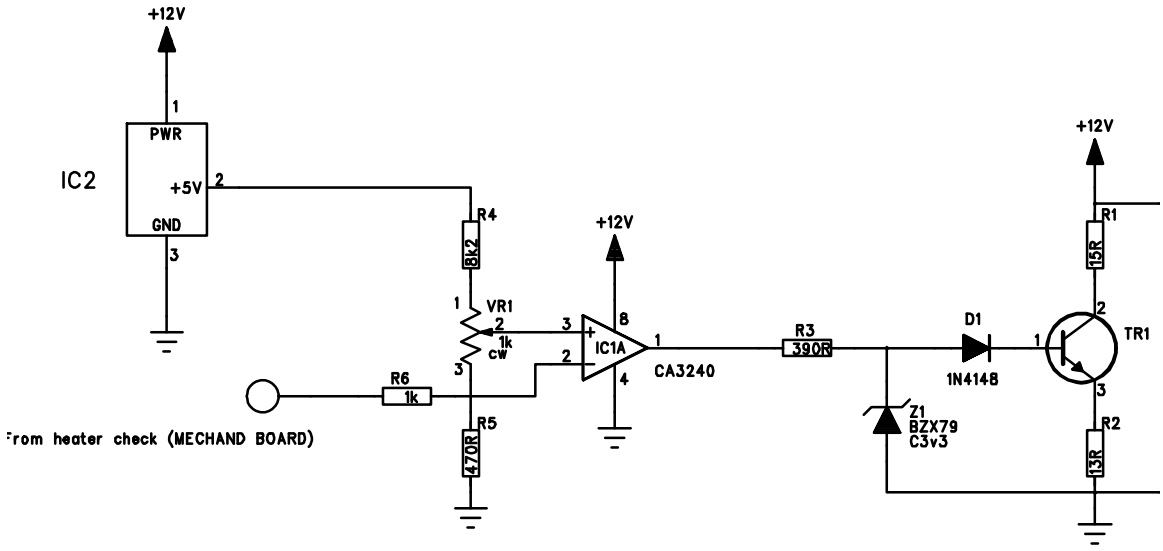
RS232 PCB - Drilling diagram - approx. 2:1 scale



RS232 PCB - Top layer, approx. 1:1 scale (board 2.6h x 4.05w)



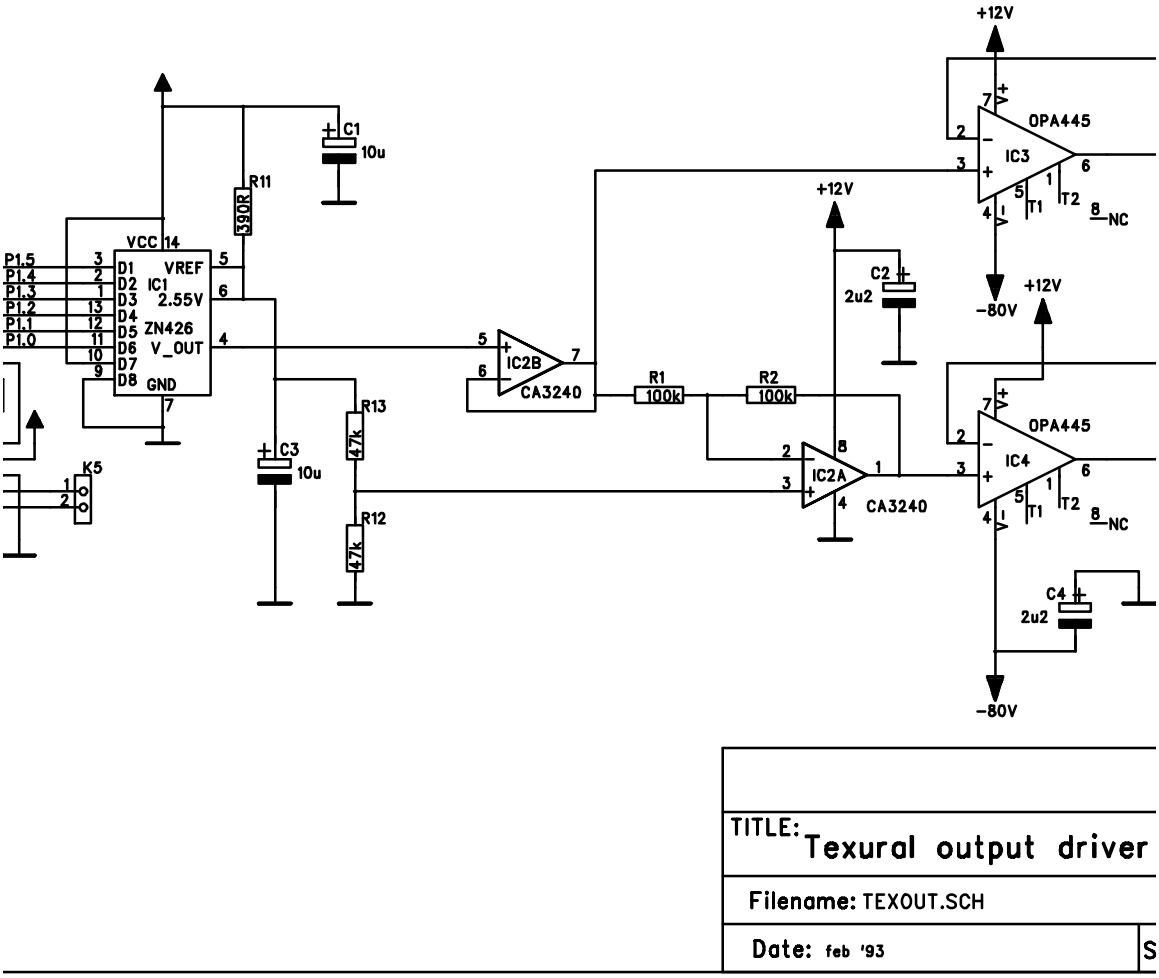
RS232 PCB - Bottom layer, view from the top.


+12V
1
PWR
IC2
+5V
2
GND
3
R4
8k2
1
VR1
2
1k
cw
3
R6
1k
From heater check (MECHAND BOARD)
R5
470R
+12V
3
+
2
IC1A
8
1
4
CA3240
R3
390R
D1
1N4148
Z1
BZX79
C3v3
+12V
R1
15R
2
TR1
1
3
R2
43R
...not too hot!!
TITLE: Heater control
Filename: heat2_2.sch
Date: aug '93

Bill Of Materials for HEAT2_2.SCH on Sun Apr 23 08:06:35 1995

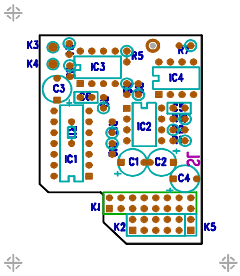
Qty	Ref	Part Name	Part type	Description
1	IC2	+5VREG		+5V LINEAR REGULATOR
1	IC1	CA3240		DUAL OP-AMP
1	D1	DIODE		Axial ss Diode OA47
1	Z1	DIODEZ,C3v3		Axial ss Zener
1	R2	R0.6W,13R		RES BODY:2.5mm CENTRES:0.4"
1	R1	R0.6W,15R		RES BODY:2.5mm CENTRES:0.4"
1	R6	R0.6W,1k		RES BODY:2.5mm CENTRES:0.4"
1	R3	R0.6W,390R		RES BODY:2.5mm CENTRES:0.4"
1	R5	R0.6W,470R		RES BODY:2.5mm CENTRES:0.4"
1	R4	R0.6W,8k2		RES BODY:2.5mm CENTRES:0.4"
1	TR1	TRNPN-TO220-BCE		TO-220 POWER TRANSISTOR
1	VR1	VRES\TOP3/8,1k	ESD 0997XX RS 160-XXX	3/8" 25 TURN POT, TOP ADJUST
3	K1-3	WIRECONN		wire connection (color-???)

This circuit was built on Veroboard, and was not transferred to PCB.


+12V
C1
10u
R11
390R
VCC 14
P1.5
P1.4
P1.3
P1.2
P1.1
P1.0
D1 VREF
D2 IC1
D3 2.55V
D4
D5 ZN426
D6 V_OUT
D7
D8 GND
IC2B
CA3240
R1
100k
R2
100k
C2
2u2
OPA445
IC3
T1
T2
NC
-80V
+12V
OPA445
IC4
T1
T2
NC
C4
2u2
-80V
+12V
C3
10u
R13
47k
R12
47k
K5
IC2A
CA3240
TITLE: Texural output driver
Filename: TEXOUT.SCH
Date: feb '93

Bill Of Materials for TEXOUT.SCH on Wed Jan 26 20:21:44 1994

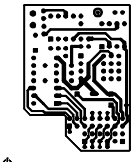
Qty	Ref	Part Name	Part type	Description
1	IC2	CA3240		DUAL OP-AMP
2	C1 C3	CAPT\2\2,10u		Tantalum Capacitor .2 pitch .25 dia
2	C2 C4	CAPT\2\2,2u2		Tantalum Capacitor .2 pitch .25 dia
1	K5	CON\2P	Homebrew (cut-down dil skt)	NOT FITTED
1	K2	CON\PCB\2X5		2x5 Way PCB Header socket
1	K1	CON\PCB\P2X8	FAR	2x8 PCB Header pins
2	IC3-4	OPA445	??	High Voltage FET-Input Op. Amp.
2	R1-2	R0.6W,100k		RES BODY:2.5mm CENTRES:0.4"
2	R6 R8	R0.6W,1k5		RES BODY:2.5mm CENTRES:0.4"
2	R3-4	R0.6W,2K2		RES BODY:2.5mm CENTRES:0.4"
1	R11	R0.6W,390R		RES BODY:2.5mm CENTRES:0.4"
4	R5 R7 R12-13	R0.6W,47k		RES BODY:2.5mm CENTRES:0.4"
2	K3-4	WIRECONN		wire connection (color-???)
1	IC1	ZN426	MAP UF39N	8-bit D/A Converter



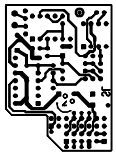
texout2.job - Wed May 12 14:47:32 1993

TEXOUT PCB - Assembly diagram - approx. 2:1 scale

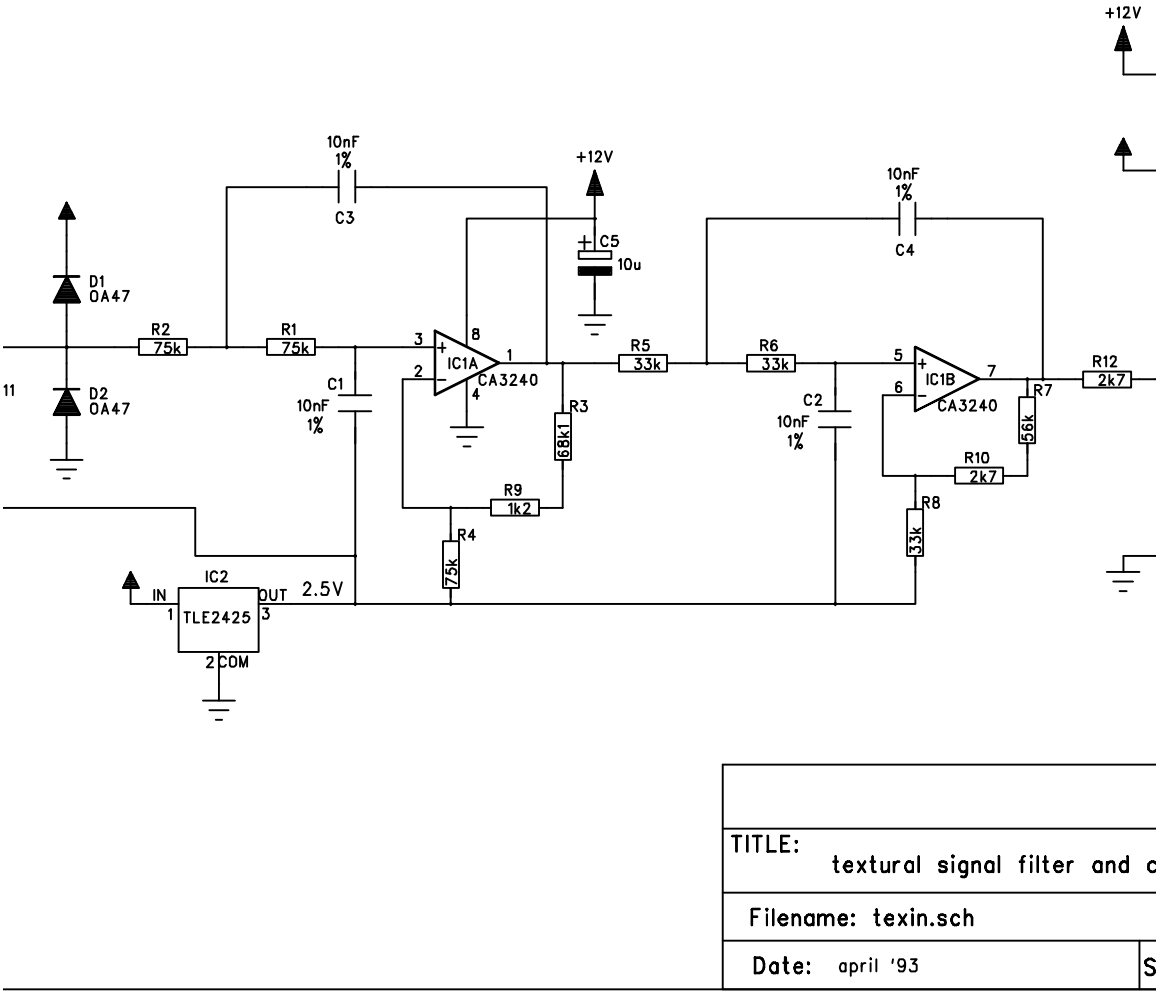
This diagram is missing!!!!!!!!!!!!!!!!
TEXOUT PCB - Drilling diagram - approx. 2:1 scale



TEXOUT PCB - Top layer - approx. 1:1 scale (board 3.4 x 7.5)

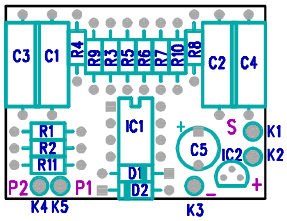


TEXOUT PCB - Bottom layer - viewed from top

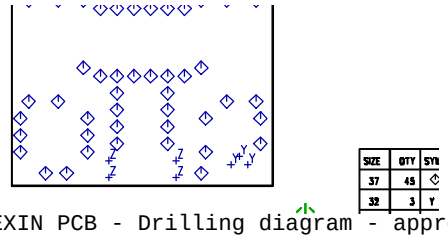

+12V
10nF
1%
C3
+12V
C5
10u
10nF
1%
C4
D1
OA47
D2
OA47
R2
75k
R1
75k
C1
10nF
1%
IC1A
CA3240
R5
33k
R6
33k
C2
10nF
1%
IC1B
CA3240
R12
2k7
R7
56k
R10
2k7
R3
68k1
R9
1k2
R4
75k
R8
33k
IC2
TLE2425
IN
OUT
COM
2.5V
11
TITLE:
textural signal filter and c
Filename: texin.sch
Date: april '93

Bill Of Materials for TEXIN.SCH on Wed Jan 26 20:41:23 1994

Qty	Ref	Part Name	Part type	Description
1	IC1	CA3240		DUAL OP-AMP
1	C5	CAPT\2\2,10u		Tantalum Capacitor .2 pitch .25 dia
4	C1-4	CAP\POLY\1%, 10nF,1%		polystyrene 1% tolerance
2	D1-3	DIODE,OA47		Axial ss Diode w Alternate.
1	R11	R0.6W,1M		RES BODY:2.5mm CENTRES:0.4"
1	R9	R0.6W,1k2		RES BODY:2.5mm CENTRES:0.4"
2	R10 R12	R0.6W,2k7		RES BODY:2.5mm CENTRES:0.4"
3	R5-6 R8	R0.6W,33k		RES BODY:2.5mm CENTRES:0.4"
1	R7	R0.6W,56k		RES BODY:2.5mm CENTRES:0.4"
1	R3	R0.6W,68k1		RES BODY:2.5mm CENTRES:0.4"
3	R1-2 R4	R0.6W,75k		RES BODY:2.5mm CENTRES:0.4"
1	IC2	TLE2425	MAP CR12N	Precision Virtual Ground
5	K1-5	WIRECONN		wire connection (color-???)

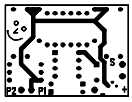


TEXIN PCB - Assembly diagram - approx. 2:1 scale.

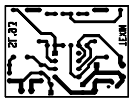


SIZE	QTY	SYM
.37	45	◇
.32	3	Y

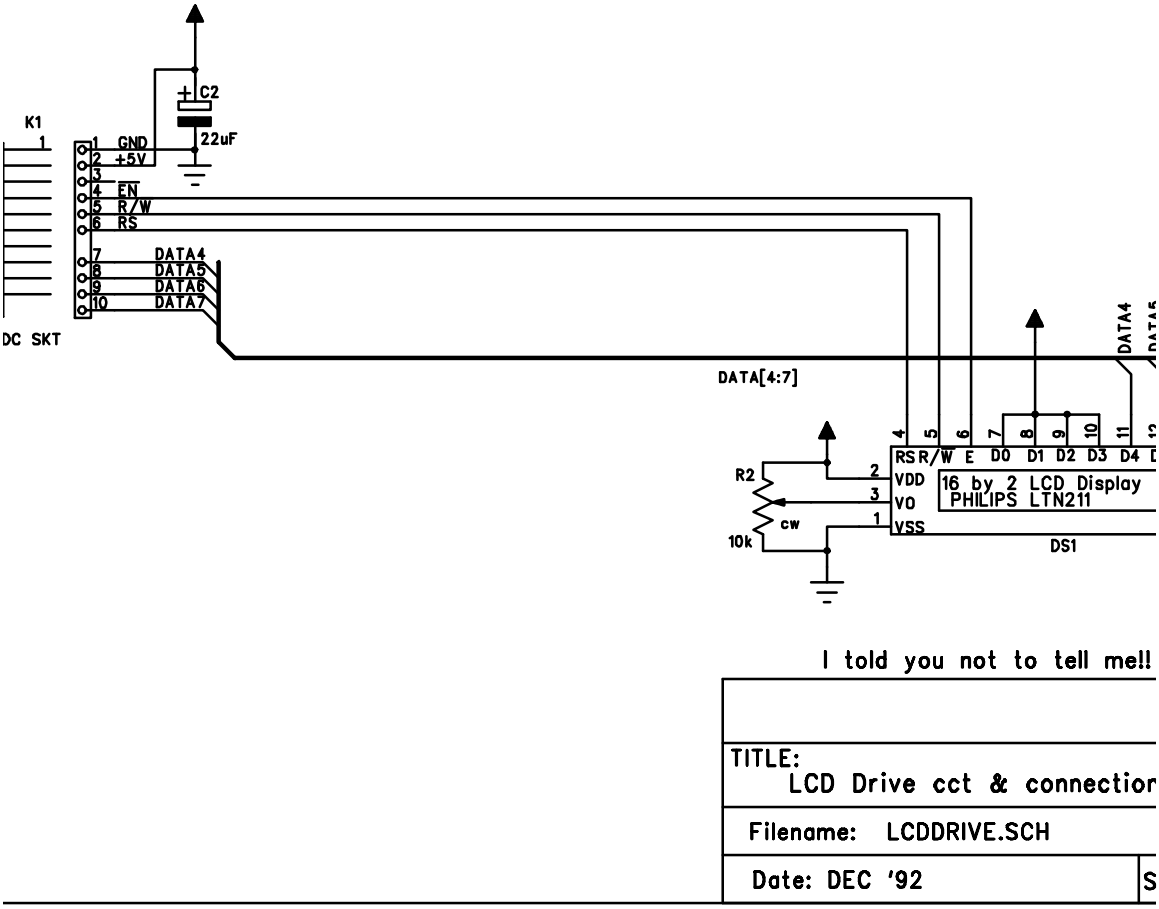
TEXIN PCB - Drilling diagram - approx. 2:1 scale.



TEXIN PCB - Top layer, approx. 1:1 scale (board 2.8h x 3.8w)



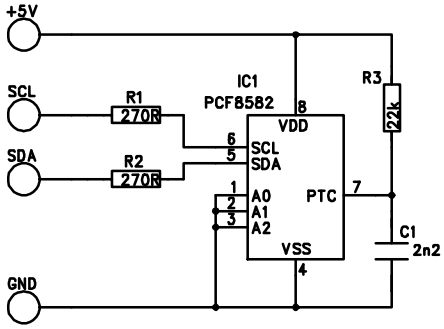
TEXIN PCB - Bottom layer, view from the top.


K1
1
1 GND
2 +5V
3
4 EN
5 R/W
6 RS
7 DATA4
8 DATA5
9 DATA6
10 DATA7
DC SKT
+C2
22uF
DATA[4:7]
DATA4
R2
10k
cw
2 VDD
3 VO
1 VSS
RS R/W E D0 D1 D2 D3 D4
16 by 2 LCD Display
PHILIPS LTN211
DS1
I told you not to tell me!!
TITLE:
LCD Drive cct & connectior
Filename: LCDDRIVE.SCH
Date: DEC '92

Bill Of Materials for LCDDRIVE.SCH on Wed Jan 26 20:46:44 1994

Qty	Ref	Part Name	Part type	Description
1	C2	CAPR+,22uF		DECOUP CAP RADIAL BODY:.2 PITCH 0.08 or 0.1
1	K1	CON\SORIB\10	RS 474-271 (SKT)	Surface mount ribbon cable + connector, 10way
1	DS1	LCD16X2	ESD 098072C	
1	R2	VRESSMD-1/4,10k	ESD 071915D	Side adjust, surface mount multiturn pot, 1/4"

The components for this circuit were mounted off the LCD module.



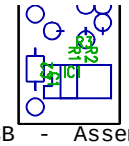
TITLE: EEPROM add-on for storing conv	
Filename: EEPROM	
Date: JAN '93	S

Non-Volatile Memory

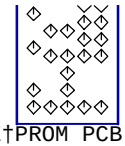
Note that this board was built into the system, connected to the POWERC2 board, but was not taken advantage of in the software. It is presented here for completeness.

Bill Of Materials for EEPROM.SCH on Tue Jan 25 23:20:26 1994

Qty	Ref	Part Name	Part type	Description
1	C1	CAP0.1,2n2		RADIAL CAP BODY:3.6x2.3mm CENTRES:.2"
1	IC1	PCF8582		256x8bit CMOS EEPROM, I2C bus
1	R3	R0.6W,22k		RES BODY:2.5mm CENTRES:0.4"
2	R1-2	R0.6W,270R		RES BODY:2.5mm CENTRES:0.4"
4	K1-4	WIRECONN		wire connection (color-???)

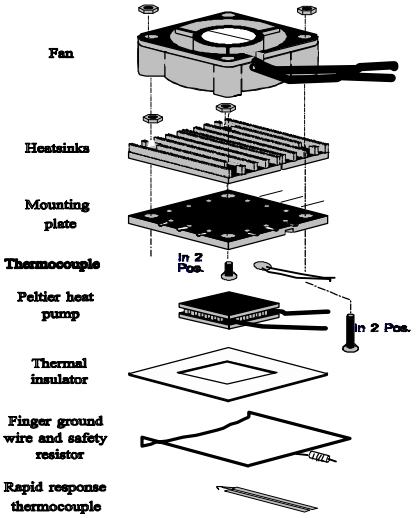


E†PROM PCB - Assembly diagram - approx. 2:1



E†PROM PCB - Drilling diagram scale.

E†PROM PCB - Bottom layer, viewed from top - approx. 1:1 scale (board 1.95 x 1.5)



	Assembly Drawing	Drawing D10.1	Sht 1 of 3

Item	Qty	Description	Source / Part No	Comments
1	1	Fan, 25x25x10, 12v	ESD Pt No 409457G Motor-One D02F12MWS	
3	2	Heatsink	Maplin Pt No KU49D	As per Drg D10.2
5	1	Mounting Plate	Aluminium Blank, 25x25x3thk	As per Drg D10.3
7	1	Thermocouple, Type T	RS Pt No 158-907	See Note 1
9	1	Peltier Heat Pump	Marlow Industries MI1023T	See Note 4
11	1	Thermal Insulator	1mm Thick Plastic	As per Drg D10.4
13	1	Finger grounding wire	Tinned copper wire, 22SWG	As per Drg D10.5
14	A/R	Clear sleeving, 1.4mm bore		
15	1	Resistor 1MÙ		
17	1	Rapid Response Thermocouple, Type T	RS Pt No 256-146	See Note 2
19	2	Screw, Countersunk, M2.5x6		
21	2	Screw, Countersunk, M2.5x20		
23	6	Full Nut, M2.5		
25	A/R	Thermal heatsink compound		See Note 3
27	A/R	Thermal epoxy	ESD Pt No 401898H Electrolube ER2074	See Notes 1,2,4

Assembly Drawing Item list	**Drg D10.1**	**Sht2 of 3**

Notes:-

1. Thermocouple, item 7, to be encapsulated in item 27, for electrical insulation, then mounted on mounting plate (5), again using item 7.

2. Peltier heat pump (9), to be mounted on mounting plate (5), using item 7.

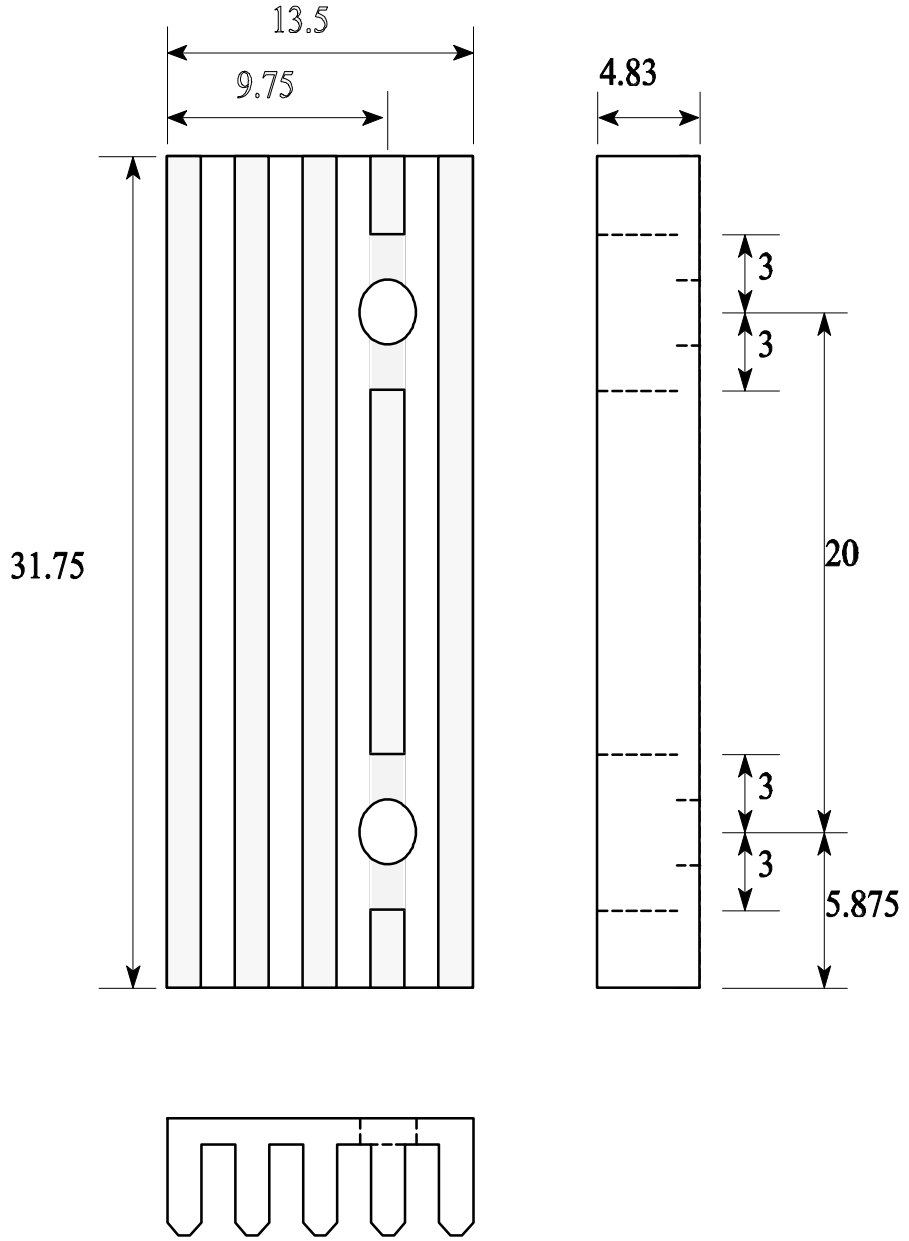
3. Base of heatsinks (3), to be smeared with thermal grease, item 25, before screwing to mounting plate.

4. Rapid thermocouple (17), to be mounted on mounting plate (5) using item 27, and insulated by smearing item 27 over it.

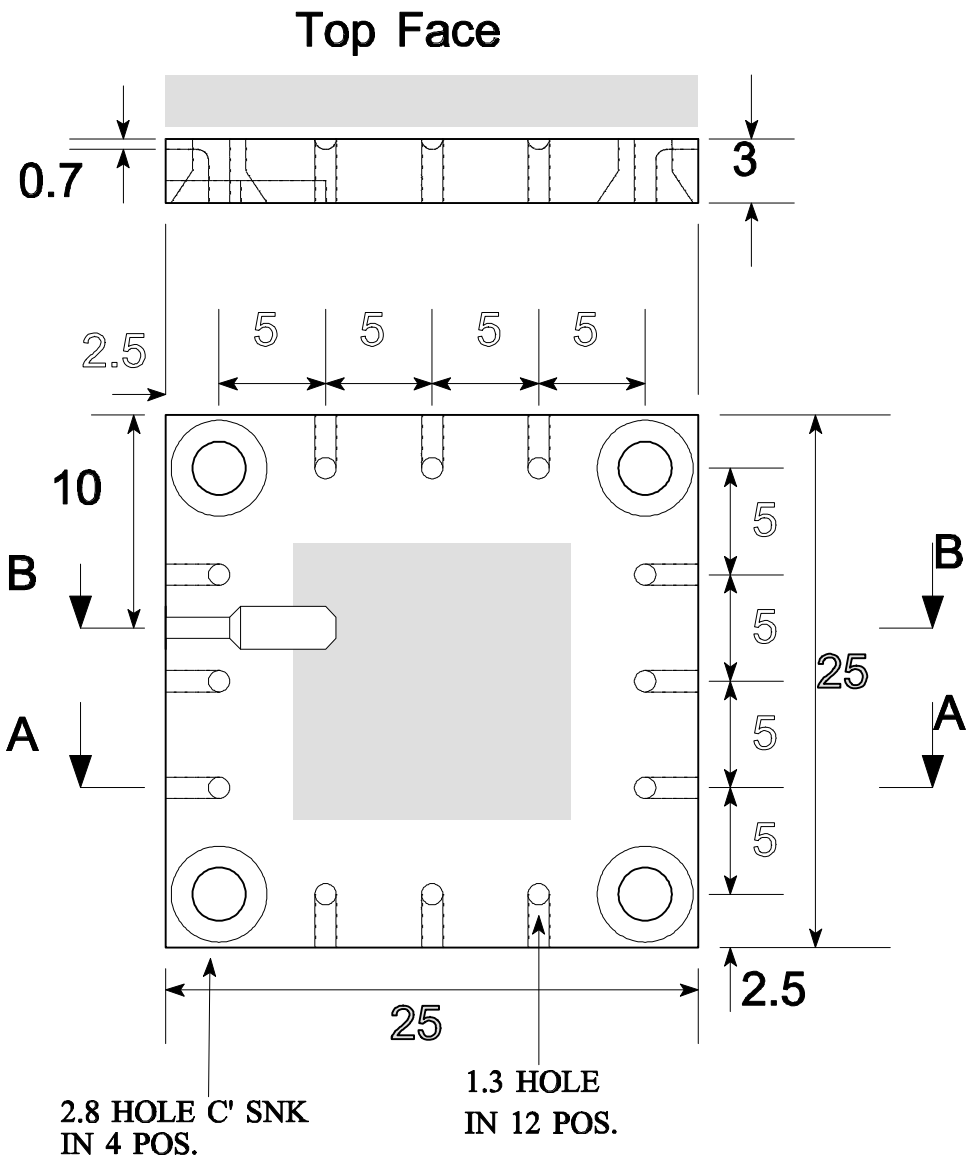
All dimensions in mm	**Assembly Drawing Notes**	**Drawing D10.1**	**Sht 3 of 3**

Notes:-

1. Heatsink fin metal removed in area hatched.
2. Outer dimensions for reference only



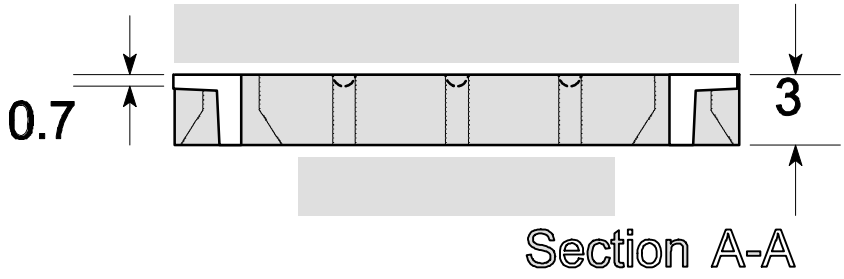
All dimensions in mm	**Detail Drawing - Heatsink Modification**	Drawing **D10.2**	**Sht 1 of 1**

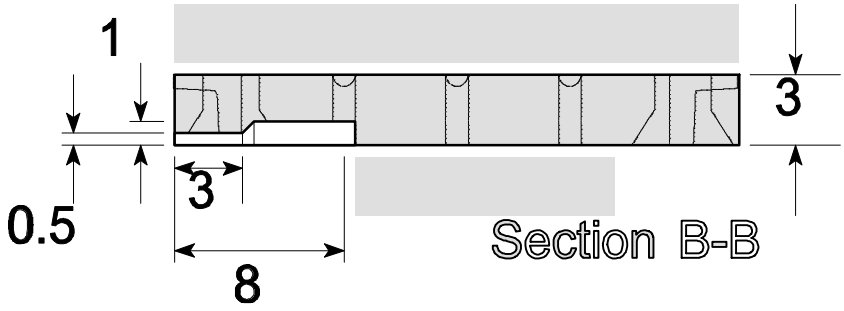


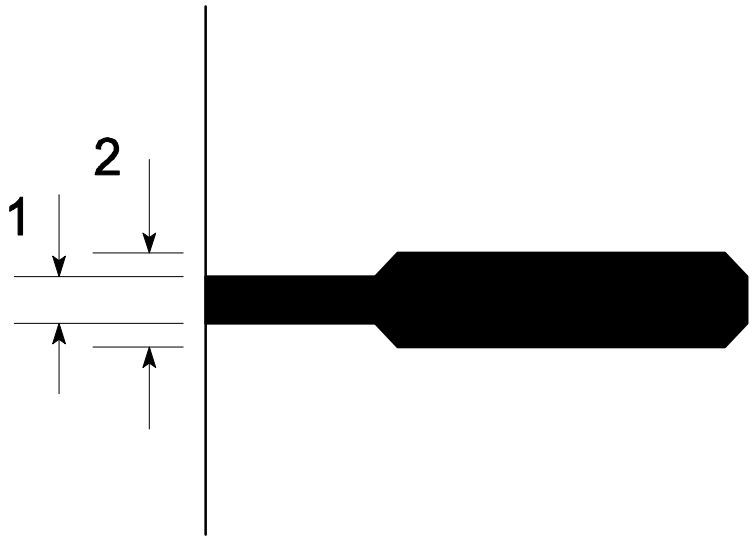
All dimensions in mm	Detail Drawing - Mounting plate	Drawing D10.3	Sht 1 of 2

Notes:-

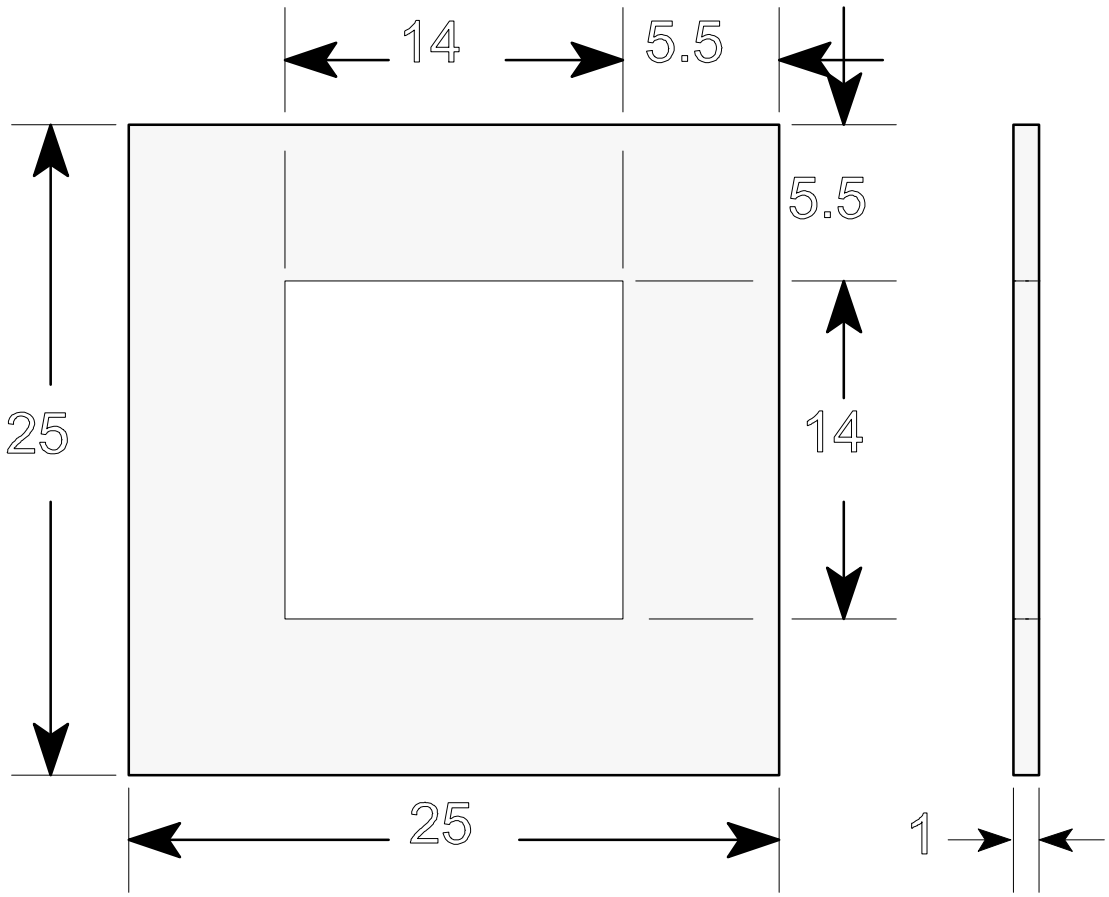
1. Areas hatched indicate surface to be smooth
2. View below shows grooves to be filed / machined from each 1.3mm hole to edge of plate.







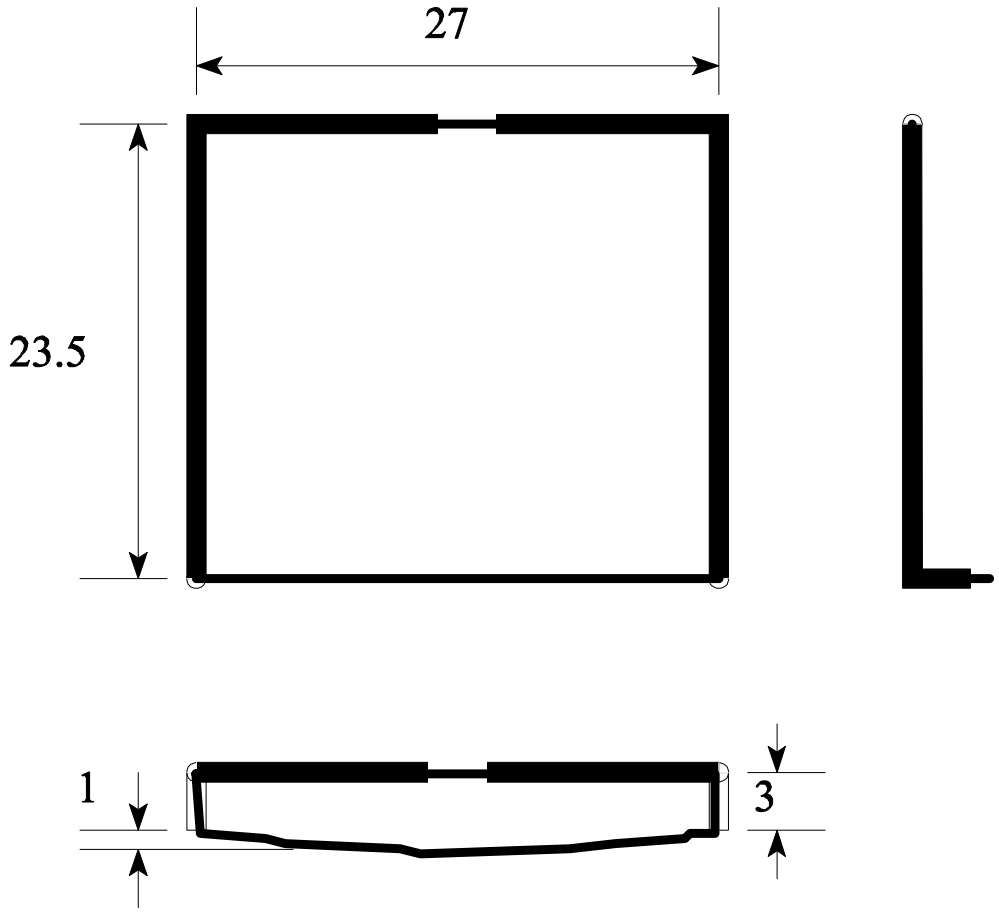
All dimensions in mm	**Detail Drawing - Mounting plate Notes**	**Drawing D10.3**	**Sht 2 of 2**



All dimensions in mm	**Detail Drawing - Thermal Insulator**	**Drawing D10.4**	**Sht 1 of 1**

Notes:-

1. Insulate areas shown using Item 14



All dimensions in mm	**Detail Drawing - Finger Ground Wire**	**Drawing D10.5**	**Sht 1 of 1**

Compiler

The C code given in this appendix was complied using Borland C++ 3.1

Instructions for program use

TGRAPH is a data storing/graphing, and parameter changing program for the project. N.B CIRCLEQ.C/H & ASYNCH.C/H are required for the interrupt driven comms. It works fine at 2400baud, and should work up to 19200baud, I think!

Commands for TGRAPH;

F1 - very basic help.

F2 - store data from Project (at 2400 baud, upto 110 secs will be stored, at 4800 baud 55 secs.)

Note; setup comms first (F4). remember kit is at 2400 baud at switch-on.

keys for 1st Question;

ENTER - Just straight data acquiresition.

M - Material recognition.

C - Computer generated material.

If answer is 'M' or 'C', 2nd and 3rd question;

Note that 'M' material recognision asks Q 2 just for saving with the data, in case of post processing (see Appendix G2 for example)

2nd Question - material type

A - Aluminium

B - Black foam

C - Card

D - Diecast

E - Eraser

F - White Foam

G - Glass

H - Hot

I - Ice cold

P - Plastic

T - Tufnol™

W - Wood

3rd question - Room temperature.

use '+' and '-' to alter.

keys while collecting data; ENTER to execute.

Note all valid commands 'bookmark' present data position.

J - JUMP +/- X from current value (holds value). Range -31.75 to + 31.75

O - OFFSET - put +/- offset on current value (no hold). Range -31.75 to +31.75

T - Tsteady - Sets value of Tsteadystate(TSTORE6). Range 0 to 63.75

A - ABSOLUTE Sets absolute value of TSTORE5 (holds val). Range 0 to 63.75

P - POWER - Set maximum power level (Proportional). Range 0 to 255

B - BANGPWR - Set maximum power level (Bang-Bang). Range 0 to 255

S - SHOCK Sets TSTORE5 to 5'C; shock temp (holds value)

C - CLEARs effects of Jump,Offset,Absolute & Shock.

@ - baud up - Ups baud rate to 4800.

ESC - end data storing.

TAB - start computer generated 'virtual material' (when answer 'C' to 1st Q)

SPACE - leave 'bookmark' at present data position.

F1 - Simulates 'ENTER' after a random time (0 to 25 seconds)

F3 - Setup Graphs - setup which lines to display/plot, colour, line type, show limits, etc. Note;

'BookMks' shows when a valid command was sent to the kit, or when SPACE pressed

'StatMks' shows when any of the status bits have changes;

"Offs" - There is an Offset set in the kit.

"Step" - Step change in kit.

"Test" - the test bit is set.

"SReq" - 'Shut Req' is shutdown request (something's gone out-of-limits)

"SAck" 'Shut Ack' is shutdown acknowledge (system has shutdown, LCD message sent)

'Error' is the calculated error term, from which the power is calculated.

'Power' is the power-level...Note, that there is possibly still a slight bug somewhere in the graphing of power, so don't believe everything you see.

F4 - Setup Comms, - comms port and baud rate, (Default COM1: & 2400 Baud)

F5 - Load data from disk (no wildcards, no dir listing), but it does handle disk drive/path etc ok

F6 - Notes - change the title/ notes on the data

F7 - Plot - Converts the graph currently on-screen, into an HPGL plotter file. (HP7550 8 pen format .. ie library plotter). If require to other plotters / printers use the PRINTGL program.

F8 - Graph - not used anymore, as all routines automatically update the onscreen graph, but it's still there, just in case.

F9 - Cursors - puts cursors on a graph, and cursor stats at top of screen

keys:

Left/Right - move along line

Shift Left/right - move along line, accelerated.

Insert - switch between cursors 0 and 1

keypad +/- switch between graph lines

F10 - Save data file

Home - reset zoomed graph, Y-axis only.

Pg Dn - reset zoomed box.

Pg Up - get zoom box onscreen

keys;

Pg Up - zoom in

Pg Dn - zoom out

cursors - move zoom box around screen

shift-cursors - alter proportions of zoom box

return - zoom to area in zoom-box

ROUTINE - TGRAPHS.C 10/04/95

```
//Written by and ' (1993-1995) Steve Lawther
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <stdlib.h>
#include <fcntl.h>
#include <io.h>
#include <string.h>
#include <graphics.h>
#include <math.h>
#include <dos.h>
#include <ctype.h>
#include <dir.h>
#include <errno.h>
#include <time.h>
/* ok, asynch.h thanks (not!) to Steve Resnick, Asylum Software*/
#include "asynch.c"
#include "circleq.c"

#include "keycodes.h"
#define LENGTH 1100
#define OFF     0
#define ON      1
#define TERRS   8
#define POWS    9
#define BOOKMARKS 10
#define STATMARKS 11
#define SHUTREQBIT 0x40
#define SHUTACKBIT 0x80
#define SCREENY 384
#define SCREENX 512
#define PLOTY   6000
#define PLOTX   9000


#define LOAD          1
#define SAVE          2
#define PLOT          3

#define NOWAIT        0
#define WAIT          1
#define PAUSE         2
#define ONELINE       8
#define TWOLINE       1
#define CALC_DISP 1
#define REMOVE    0
#define CBUFFER 1024 /*2048*/

char filename[MAXPATH]=" \0";
char drive[MAXDRIVE],dir[MAXDIR],file[MAXFILE+MAXEXT],ext[MAXEXT];
int disk,poszero=0,testint;
FILE *filehd;
/* typedef unsigned char byte;*/
enum {P_NOPEN, P_BLACK, P_RED,P_GREEN,P_BLUE,
                            P_LIGHTBLUE,P_PINK,P_YELLOW,P_BROWN};
char tempstr[160];
char tempstr2[40];
long value[2];
unsigned long timet[2];
//byte status[LENGTH];
struct {unsigned int timer;
             signed int t[8];
             signed int power;
             byte status;
             unsigned hitimer:2;
             unsigned bookmark:1;
             unsigned automark:1;} data[LENGTH];
struct {char title[0x40],notes[0x50];int comno, baud, per_sec;} dataxtra;
struct time smh = {0,0,0,0};
struct date dmy = {0,0,0};
struct
  { byte value[7]; byte on_off;int screen_col, plot_col;byte line,limits;int mx,mn;}
   setup[13] = {{"peltier", ON, LIGHTRED, P_RED,       ON, OFF,46,0},
                     {"finger ", ON, GREEN,    P_GREEN,     ON, OFF,0, 0},
                     {"heatsnk", OFF,RED,      P_PINK,      ON, ON ,76,0},
                     {"spare  ", OFF,WHITE,    P_YELLOW,    OFF,OFF,0, 0},
                     {"Mheater", OFF,CYAN,     P_PINK,      OFF,OFF,49,21},
                     {"Mtouch ", ON, BLUE,     P_LIGHTBLUE, ON, OFF,121,0},
                     {"Msteady", ON, YELLOW,   P_YELLOW,    OFF,OFF,0, 0},
```

```
                      {"Mambint", OFF,WHITE,    P_YELLOW,    OFF,OFF,0, 0},
                      {"Error  ", OFF,LIGHTGREEN,P_BROWN,    ON, OFF,0, 0},
                      {"Power  ", OFF,BROWN,    P_BLUE,      ON, OFF,0, 0},
                      {"BookMks", OFF,BLACK,    P_BLACK,     ON, OFF,0, 0},
                      {"StatMks", OFF,WHITE,    P_BLACK,     OFF,OFF,0, 0},
                      {"Sparebt", OFF,WHITE,    P_BLACK,     OFF,OFF,0, 0}};
struct { int comno; int baud;} comms = {1,2400};
byte temp,dat[0x40]="",title[0x40],notes[0x50],outno[10];
unsigned long number, lastno;
int i=0,saveflag=0,column,datalength=0,disdatamin=0,disdatamax=0,calcx,calcy;
struct {int X; int Y; int line;} cursor[2] ={{ 0,0,0},{0,0,0}};
int csrflag=0,winmangled=1,csronflag=0;

unsigned long len;
long mintx=0,maxtx=0,maxpow=0,minpow=0,mine=0,maxe=0,larger;


byte initialize(void);
byte load_from_disk(void);
byte save_data(void);
byte menu(void);
 int cursordata(int);
byte help(byte);
void zoom(void);
byte setup_graph(void);
 void displayline(int);
byte setup_comms(void);
byte alter_notes(void);
 void disnotes(char);
byte plotter(void);
 void plotln_n_txt(int,int,char*,char,char);
byte live_data_in(void);
 int initialcomm(void);
 unsigned int getcomm(void);
 void setcomm(void);
byte graph(void);
 char *decimal(int,int);
 char *dec_and_units(int,int);
 void line_n_text(int,int,char*,char,char);
 void findmaxmin(void);
 long terror(int);
 long scale(long,long,long,int);

  void drawcursor(int x,int y);
char getfilename(int);
 char message(char*,char,char,int);
 int vtoa(int);
 void updatescreen(int);
 int decode_instr(void);
 byte qp_data(void);


void main(){ //int argc, char *argv[]) {
/* if(argc > 1) {cprintf("options not supported yet!!!");exit(0);}*/
disk=getdisk();
initialize();
menu();
restorecrtmode();
setdisk(disk);
cprintf("COM1/2 ports needs resetting to original value\n\r");
cprintf("I can't be bothered to program it!\n\r");
cprintf("\n\r   Bye!!\n\r");
exit(0);
}


byte initialize(void) {

int graphics_adaptor,graphics_mode;
graphics_adaptor=VGA;graphics_mode=VGAHI;
initgraph(&graphics_adaptor,&graphics_mode,"D:\\bc31\\bgi");
if (graphresult() !=grOk) {cprintf("initgraph failed");abort();}

settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);

setcolor(GREEN);
rectangle(180,130,480,310);
setfillstyle(SOLID_FILL,GREEN);
floodfill(200,200,GREEN);
setfillstyle(SLASH_FILL,CYAN);
floodfill(10,10,GREEN);
setcolor(WHITE);
```

```
outtextxy(200,150,"Temperature Graphing");
outtextxy(260,200,"(12/1/95)  By");
outtextxy(210,250,"Steve Lawther (c)1993");
dataxtra.per_sec=10;
return(0);
}



byte menu(void) {
int key;

setcolor(WHITE);
while (1==1) {

 setbkcolor(BLACK);
 setcolor(YELLOW);
 if (csronflag) cursordata(CALC_DISP);
 settextstyle(SMALL_FONT,HORIZ_DIR,5);
 if (winmangled)
 {
  winmangled=0;
  setviewport(0,0,637,32,1);
  clearviewport();
  setcolor(RED);
  rectangle(0,0,637,32);
  setcolor(LIGHTGRAY);
  setfillstyle(SOLID_FILL,RED);
  floodfill(4,4,3);
  outtextxy(8,0,"F1-HELP F3-SETGRAPH F5-FILE  F7-PLOT  F9-CURSOR");
  outtextxy(8,16,"F2-LIVE F4-SETCOMMS F6-NOTES F8-GRAPH F10-SAVE");
  setcolor(YELLOW);
  if (!datalength) outtextxy(400,0,"NO");
           else outtextxy(390,0,itoa((datalength+1),outno,10));
  outtextxy(420,0,"data points in memory");
  if (comms.comno==0) strcpy(tempstr,"COM1: ");
           else strcpy(tempstr,"COM2: ");
  itoa(comms.baud,tempstr+6,10);
  outtextxy(400,16,strcat(tempstr," Baud"));
 }
 setcolor(YELLOW);

 while (!bioskey(1));
 key=bioskey(0);
 if (csronflag) cursordata(REMOVE);
 switch (key) {
  case F1: help(1);break;
  case F2:           if (!live_data_in()) if (datalength > 20)
{graph();qp_data();alter_notes();graph();}
                     break;
  case F3:           setup_graph(); if (datalength > 20) graph(); break;
  case F4:           setup_comms(); break;
  case F5:           if (!load_from_disk()) graph();setbkcolor(BLACK); break;
  case F6:           alter_notes();break;
  case F7:           plotter(); break;
  case F8:           graph();break;
  case F9: if (datalength > 20) csronflag=1-csronflag;break;
  case F10:          save_data(); break;
  case ESC:          if (saveflag)
                        {if(tolower(message("Data not saved...save it? Y/n",
                                    ONELINE,WAIT,BLUE)) !='n') save_data();}
                     return(0);
  case INSERT:       if (datalength > 20) csrflag=(1-csrflag);
                     break;
  case KPPLUS:  if (csronflag) if (++cursor[csrflag].line>POWS) cursor[csrflag].line=0;
                     break;
  case KPMNUS:  if (csronflag) if (--cursor[csrflag].line<0) cursor[csrflag].line=POWS;
                     break;
  case CURLT:        if (csronflag)
                       {int step;
                        step = (bioskey(2) & 0x03)? 5: 1;
                        if ((cursor[csrflag].X-=step) < 0) cursor[csrflag].X=0;
                        if (cursor[csrflag].X > disdatamax) cursor[csrflag].X =
disdatamax;
                       }
                     break;
  case CURRT:   if (csronflag)
                      if (bioskey(2) & 0x03) {if ((cursor[csrflag].X+=5) > datalength)
cursor[csrflag].X=datalength;}
                       else if ((cursor[csrflag].X+=1) > datalength)
cursor[csrflag].X=datalength;
                     if (cursor[csrflag].X < disdatamin) cursor[csrflag].X = disdatamin;
```

```
                        break;
  case PGDN:
  case KPPGDN:  disdatamin=0;disdatamax=datalength;
  case KPHOME:
  case HOME:            findmaxmin();
                        graph();
                        break;
  case PGUP:
  case KPPGUP:  zoom();
                        i=graph();
                        break;
  }
 }
}

void zoom(void) {
int scrmaxy=384,scrminy=0;
int key,minx=0,maxx=SCREENX,miny=0,maxy=SCREENY;
int centrex,centrey,boxx,boxy;
int jumpx,jumpy,shift=0,ddmin;
long y1,y2,eemin,ppmin,ttmin;

if ((disdatamax-disdatamin) < 20 || (scrmaxy-scrminy)<20) return;
jumpx=maxx/20;jumpy=maxy/20;
centrex=(minx+maxx)/2;
centrey=(miny+maxy)/2;
boxx=(maxx/2)-jumpx;
boxy=(maxy/2)-jumpy;
setviewport(80,64,639,479,1);
setwritemode(1);
do
 {
 rectangle(centrex-boxx,centrey-boxy,centrex+boxx,centrey+boxy);
 key=bioskey(0);
 if (bioskey(2) & 0x03) shift=1;
            else shift=0;
 rectangle(centrex-boxx,centrey-boxy,centrex+boxx,centrey+boxy);
 switch((key & 0xFF00)+shift) {
            case CURLT:         centrex-=jumpx;
                                break;
            case CURRT:         centrex+=jumpx;
                                break;
/* remember y axis upsidedown */
            case CURDN:         centrey+=jumpy;
                                break;
            case CURUP:         centrey-=jumpy;
                                break;
/* shift + cursor keys alter shape of box  */
            case(CURRT+1):  if (boxx >= (2*jumpx)) boxx-=jumpx;
                                break;
            case(CURLT+1):  if ((boxx+jumpx)<=(maxx-minx)/2) boxx+=jumpx;
                                break;
            case(CURDN+1):  if ((boxy+jumpy)<=(maxy-miny)/2) boxy+=jumpy;
                                break;
            case(CURUP+1):      if (boxy >= (2*jumpy)) boxy-=jumpy;
                                break;
            case PGUP:          if (boxx >= (2*jumpx) && boxy >= (2*jumpy))
                                        {boxx-=jumpx;boxy-=jumpy;}
                                break;
            case PGDN:          if ((boxx+jumpx)<=(maxx-minx)/2&&(boxy+jumpy)<=(maxy-miny)/2)
                                        {boxx+=jumpx;boxy+=jumpy;}
                                break;
            }
 if ((centrex-boxx) < minx) centrex=minx+boxx;
 if ((centrex+boxx) > maxx) centrex=maxx-boxx;
 if ((centrey+boxy) > maxy) centrey=maxy-boxy;
 if ((centrey-boxy) < miny) centrey=miny+boxy;
 }while ((key != ESC) && (key != ENTER));
setwritemode(0);
if (key==ESC) return;
ddmin=disdatamin+((long)(disdatamax-disdatamin)*(long)(centrex-boxx))/SCREENX;
disdatamax=disdatamin+((long)(disdatamax-disdatamin)*(long)(centrex+boxx))/SCREENX;
disdatamin=ddmin;
if (disdatamin < 0) disdatamin=0;
y1=SCREENY-(centrey+boxy);
y2=SCREENY-(centrey-boxy);
if ((mintx != 0) || (maxtx != 0))
            {
            ttmin = mintx+(((maxtx-mintx)*y1)/SCREENY);
            maxtx = mintx+(((maxtx-mintx)*y2)/SCREENY);
            mintx = ttmin;
            }
```

```
if ((minpow != 0) || (maxpow != 0))
           {
           ppmin = minpow+(((maxpow-minpow)*y1)/SCREENY);
           maxpow = minpow+(((maxpow-minpow)*y2)/SCREENY);
           minpow = ppmin;
           }
if ((mine != 0) || (maxe != 0))
           {
           eemin = mine+(((maxe-mine)*y1)/SCREENY);
           maxe = mine+(((maxe-mine)*y2)/SCREENY);
           mine = eemin;
           }

return;
}


int cursordata(int function) {

setviewport(0,0,639,479,1);
if (function==CALC_DISP) {
 settextstyle(SMALL_FONT,HORIZ_DIR,4);
 setfillstyle(EMPTY_FILL,BLACK);
 bar(100,48,500,65);
 for (i=0;i<2;i++) {
  if (cursor[i].line==POWS)
                     {value[i]=data[vtoa(cursor[i].X)].power;
                     if(setup[POWS].on_off==ON) cursor[i].Y =
SCREENY-scale(value[i],maxpow,minpow,SCREENY);}
    else {if (cursor[i].line==TERRS)
                     {value[i]=terror(cursor[i].X);
                     cursor[i].Y = SCREENY-scale(value[i],maxe,mine,SCREENY);}
           else      {value[i]=data[vtoa(cursor[i].X)].t[cursor[i].line];
                     cursor[i].Y = SCREENY-scale(value[i],maxtx,mintx,SCREENY);}}

timet[i]=((((long)data[vtoa(cursor[i].X)].timer)+data[vtoa(cursor[i].X)].hitimer*0x10000)*1
536L)/1000L;
  strcat(strcat(strcpy(tempstr,"C"),itoa(i,tempstr2,10))," ");

strcat(strcat(tempstr,setup[cursor[i].line].value),dec_and_units(value[i],cursor[i].line));
  outtextxy(100+(i*120),48,tempstr);
  strcat(strcat(strcpy(tempstr,"Time:"),ltoa(timet[i],tempstr2,10))," mS");
  outtextxy(116+(i*120),56,tempstr);
  }
 if (cursor[1].line==cursor[0].line)
    {strcat(strcpy(tempstr,"\xEB" " "),dec_and_units(value[1]-value[0],cursor[1].line));
     outtextxy(340,48,tempstr);}
 strcat(strcat(strcpy(tempstr,"\xEB" "t "),ltoa(timet[1]-timet[0],tempstr2,10))," mS");
 outtextxy(340,56,tempstr);
 }
drawcursor(cursor[0].X,cursor[0].Y);  //note totally inefficient at this point
drawcursor(cursor[1].X,cursor[1].Y);
while( bioskey(1)) bioskey(0);
return(0);
}

void drawcursor(int x,int y)
{
int xpos;
long offsetx=80,offsety=64;
if ((x < disdatamin) || (x > disdatamax)) return;
setwritemode(1);
setlinestyle(DASHED_LINE,0,NORM_WIDTH);
xpos=offsetx+scale(x,disdatamax,disdatamin,SCREENX);
moveto(xpos,offsety);
linerel(0,SCREENY);
moveto(offsetx,y+offsety);
linerel(SCREENX,0);
setlinestyle(SOLID_LINE,0,NORM_WIDTH);
setwritemode(0);
}


byte graph(void) {

byte j=1,anyton=0,yscale=80;
int mn,mx,slen,counts=0;
long li,t,terr,pow,xpos;
if ((disdatamin>disdatamax-3) || (datalength < 5)) return(1);
setviewport(0,32,639,479,1);
clearviewport();
```

```
setviewport(yscale,64,639,479,1);
setlinestyle(DOTTED_LINE,0,1);
settextjustify(RIGHT_TEXT,TOP_TEXT);
settextstyle(SMALL_FONT,VERT_DIR,4);
if (disdatamax-disdatamin > 600 )  j=5;
for (li=((disdatamin/dataxtra.per_sec)+j)*dataxtra.per_sec
                      ;li<=((disdatamax/dataxtra.per_sec)*dataxtra.per_sec)
                      ;li+= j*dataxtra.per_sec)
 {
 moveto(scale(li,disdatamax,disdatamin,SCREENX),0);
 setcolor(DARKGRAY);
 linerel(0,SCREENY);
 moverel(2,2);
 setcolor(LIGHTGRAY);
 outtext(itoa(li/dataxtra.per_sec,tempstr,10));
 }
setviewport(yscale,64,639,64+SCREENY,1);
settextjustify(LEFT_TEXT,TOP_TEXT);
setlinestyle(SOLID_LINE,0,1);
settextstyle(SMALL_FONT,HORIZ_DIR,4);
for (j=0;j<8;j++) {
 if (setup[j].on_off == ON)
  {
  moveto(0,192);
  anyton++;
  setcolor(setup[j].screen_col);
  for (li=disdatamin;li<=disdatamax;li++)
   {
   xpos=scale(li,disdatamax,disdatamin,SCREENX);
   t=data[vtoa(li)].t[j];
   if (setup[j].line==ON)
            lineto(xpos,SCREENY-scale(t,maxtx,mintx,SCREENY));
     else putpixel(xpos,SCREENY-scale(t,maxtx,mintx,SCREENY),setup[j].screen_col);
   }
  moveto(xpos,SCREENY-scale(t,maxtx,mintx,SCREENY));
  outtext("T");
  outtext(setup[j].value);
  if (setup[j].limits == ON)
    {
    setlinestyle(DOTTED_LINE,0,1);
    mx=setup[j].mx*256;mn=setup[j].mn*256;
    if (mx && mx>mintx && mx<maxtx)
                      {
                      moveto(SCREENX,SCREENY-scale(mx,maxtx,mintx,SCREENY));
                      linerel(-SCREENX,0);
                      }
    if (mn && mn>mintx && mn<maxtx)
                      {
                      moveto(SCREENX,SCREENY-scale(mn,maxtx,mintx,SCREENY));
                      linerel(-SCREENX,0);
                      }
    setlinestyle(SOLID_LINE,0,1);

    }
  }
 }
//show power line if required
if (setup[POWS].on_off == ON)
 {
 moveto(0,192);
 setcolor(setup[POWS].screen_col);
 for (li=disdatamin;li<=disdatamax;li++) {
  xpos=scale(li,disdatamax,disdatamin,SCREENX);
  pow=data[vtoa(li)].power;
  if ( pow < 0) pow=(-(pow & 0xFF));
  if (setup[POWS].line==ON) lineto(xpos,SCREENY-scale(pow,maxpow,minpow,SCREENY));
    else putpixel(xpos,SCREENY-scale(pow,maxpow,minpow,SCREENY),setup[POWS].screen_col);
 }
 }
//show Temp error line if required
moveto(0,192);
if (setup[TERRS].on_off == ON)
 {
 setcolor(setup[TERRS].screen_col);
 for (li=disdatamin;li<=disdatamax;li++) {
  xpos=scale(li,disdatamax,disdatamin,SCREENX);
  terr=terror(li);
  if (setup[TERRS].line==ON) lineto(xpos,SCREENY-scale(terr,maxe,mine,SCREENY));
    else putpixel(xpos,SCREENY-scale(terr,maxe,mine,SCREENY),setup[TERRS].screen_col);
 }
 }
//show bookmarks
```

```
if (setup[BOOKMARKS].on_off == ON)
 {
 setcolor(WHITE);
 for (li=disdatamin;li<=disdatamax;li++)
  {
  if ((data[vtoa(li)].bookmark) || (data[vtoa(li)].automark))
           {
           xpos=scale(li,disdatamax,disdatamin,SCREENX);
                     {
                     if (data[vtoa(li)].bookmark) setcolor(WHITE);
                                  else setcolor(LIGHTBLUE);
                     moveto(xpos,370);
                     linerel(0,15);
                     }
           }
  }
 }

moveto(0,192);
if (setup[STATMARKS].on_off == ON)
 {
 char codes[8][5] = {"ADC","I2C","Offs","Pow","Step","Test","SReq","SAck"};
 setcolor(setup[STATMARKS].screen_col);
 for (j=2;j<8;j++)
   if ((data[vtoa(disdatamin)].status >> j) % 2)
outtextxy(scale(li,disdatamax,disdatamin,SCREENX),370- 8 * counts++,codes[j]);
 for (li=disdatamin+1;li<=disdatamax;li++) {
  counts=0;
  xpos=scale(li,disdatamax,disdatamin,SCREENX);
  for (j=2;j<8;j++)
  {
   switch (((data[vtoa(li)].status >> j) %2) - ((data[vtoa(li-1)].status >> j) %2))
           {
           case -1:   moveto(xpos,378-8*counts);
                              linerel(0,-2);
linerel(-2,-4);linerel(2,4);linerel(2,-4);linerel(-2,4);linerel(0,-5);
                              outtextxy(xpos+4,370-8*counts++,codes[j]);
                              break;
           case 0:    break;
           case 1:         moveto(xpos,378-8*counts);
                              linerel(0,-6);
linerel(-2,4);linerel(2,-4);linerel(2,4);linerel(-2,-4);linerel(0,-1);
                              outtextxy(xpos+4,370-8*counts++,codes[j]);
                              break;
           }
  }
 }
}

setviewport(0,32,639,479,1);
setcolor(WHITE);
moveto(yscale,416);
linerel(512,0);
line_n_text(yscale,424,decimal(disdatamin,dataxtra.per_sec),0,1);
line_n_text((yscale+SCREENX),424,decimal(disdatamax,dataxtra.per_sec),0,1);
outtextxy(yscale+SCREENX/2-60,430,"Seconds (Approx)");
j=0;
if (anyton)
 {
 setcolor(LIGHTGRAY);
 moveto(yscale,32);
 linerel(0,382);
 line_n_text(yscale,32,decimal(maxtx,256),-1,0);
 line_n_text(yscale,416,decimal(mintx,256),-1,0);
 line_n_text(yscale,32,"TEMP",0,-1);
 if (labs(mintx)==maxtx) {line_n_text(yscale,224,"0",-1,0);j++;};
 yscale-=28;
 }
if (setup[POWS].on_off ==ON)
 {
 setcolor(setup[POWS].screen_col);
 moveto(yscale,32);
 linerel(0,382);
 line_n_text(yscale,32,itoa((maxpow),tempstr,10),-1,0);
 line_n_text(yscale,416,itoa((minpow),tempstr,10),-1,0);
 line_n_text(yscale,32,"Power",0,-1);
 if (labs(minpow)==maxpow) {line_n_text(yscale,224,"0",-1,0);j++;}
 yscale-=28;
 }
if (setup[TERRS].on_off ==ON)
 {
 setcolor(setup[TERRS].screen_col);
```

```
 moveto(yscale,32);
 linerel(0,382);
 line_n_text(yscale,32,decimal(maxe,256),-1,0);
 line_n_text(yscale,416,decimal(mine,256),-1,0);
 line_n_text(yscale,32,"Error",0,-1);
 if (labs(mine)==maxe) {line_n_text(yscale,224,"0",-1,0);j++;}
 }
setcolor(LIGHTGRAY);
if (j) {
           moveto(80,224);
           linerel(512,0);
           }
//put title, date , baud, etc on screen
settextstyle(SMALL_FONT,HORIZ_DIR,4);
settextjustify(CENTER_TEXT,TOP_TEXT);
slen=79-strlen(dataxtra.title);
strncat(strcat(strcpy(tempstr,dataxtra.title),":"),dataxtra.notes,slen);
outtextxy(260,4,tempstr);
settextjustify(LEFT_TEXT,TOP_TEXT);
strcat(strcpy(tempstr,"COM"),itoa(dataxtra.comno+1,tempstr2,10));
strcat(tempstr," ");
strcat(tempstr,itoa(dataxtra.baud,tempstr2,10));
outtextxy(520,0,strcat(tempstr," Baud"));
strcat(strcat(strcpy(tempstr,"Time "),itoa(smh.ti_hour,tempstr2,10)),":");
outtextxy(520,8,strcat(tempstr,itoa(smh.ti_min,tempstr2,10)));
strcat(strcat(itoa(dmy.da_day,tempstr,10),"/"),itoa(dmy.da_mon,tempstr2,10));
outtextxy(580,8,strcat(strcat(tempstr,"/"),itoa(dmy.da_year,tempstr2,10)));
outtextxy(520,16,strcat(itoa(dataxtra.per_sec,tempstr,10)," Readings/sec"));
return(0);
}

char *decimal(int top,int bot) {
int i;
i=top%bot;
if ( (i < 0) && ((top/bot) == 0) ) strcpy(tempstr2,"-\0");
   else strcpy(tempstr2,"\0");
strcat(itoa(top/bot,tempstr2+strlen(tempstr2),10),".");
i=abs(i)*100/bot;
if ((i < 10) && (i > -10)) strcat(tempstr2,"0\0");
itoa(i,tempstr2+strlen(tempstr2),10);
return(tempstr2);
}

void findmaxmin(void) {
int i;
byte j;
long larger;
maxtx=0;mintx=0;maxpow=0;minpow=0;maxe=0;mine=0;
if (datalength==0) exit(1);
for (i=disdatamin;i<=disdatamax;i++) {
  for (j=0;j<8;j++)  {
    if (setup[j].on_off == ON)
           {
           if (data[vtoa(i)].t[j] > maxtx) maxtx=data[vtoa(i)].t[j];
           if (data[vtoa(i)].t[j] < mintx) mintx=data[vtoa(i)].t[j];
           }
    }
  if (setup[POWS].on_off == ON && data[vtoa(i)].power > maxpow) maxpow=data[vtoa(i)].power;
  if (setup[POWS].on_off == ON && data[vtoa(i)].power < minpow) minpow=data[vtoa(i)].power;
  if (setup[TERRS].on_off == ON && terror(i) > maxe) maxe=terror(i);
  if (setup[TERRS].on_off == ON && terror(i) < mine) mine=terror(i);
  }
if ((labs(mintx))>(labs(maxtx))) larger=(labs(mintx)); else larger=maxtx;
larger=(((larger/2560)+1)*2560);
if (mintx<0)    {
                      if (maxtx<0) {maxtx=0;mintx=-larger;}
                             else {maxtx=larger;mintx=-larger;}
                      }
           else {maxtx=larger;mintx=0;}
if (setup[POWS].on_off == ON)
 {
 if ((labs(minpow))>(labs(maxpow))) larger=(labs(minpow)); else larger=maxpow;
 if (larger>128) larger=255; else larger=128;
 if (minpow<0)  {
                      if (maxpow<0) {maxpow=0;minpow=-larger;}
                             else {maxpow=larger;minpow=-larger;}
                      }
           else {maxpow=larger;minpow=0;}
 }
if (setup[TERRS].on_off == ON)
 {
 if ((labs(mine))>(labs(maxe))) larger=(labs(mine)); else larger=maxe;
```

```
 larger=(((larger/2560)+1)*2560);
 if (mine<0)    {
                      if (maxe<0) {maxe=0;mine=-larger;}
                              else {maxe=larger;mine=-larger;}
                      }
           else {maxe=larger;mine=0;}
 }
}

long terror(int i) {
return (data[vtoa(i)].t[0]-((data[vtoa(i)].t[1]+data[vtoa(i)].t[5])-data[vtoa(i)].t[6]));
}

long scale(long y,long maxy,long miny,int scale) {
return (((y-miny)*scale)/(maxy-miny));
}

byte help(byte helpno) {
setcolor(YELLOW);
setviewport(80,136,560,360,1);
setfillstyle(SOLID_FILL,BLUE);
bar(1,1,480,224);
setfillstyle(SOLID_FILL,BLACK);
bar(4,28,476,220);
outtextxy(240,2,"TGRAPH Ver. 6  Keys / Help");
outtextxy(4,38,"F1 - Help (this screen surprisingly!!!)");
outtextxy(4,50,"F2 - Read data from Kit");
outtextxy(4,62,"F3 - Alter graph lines shown and the colours of data lines etc");
outtextxy(4,74,"F4 - Set up comms port number and Baud rate.");
outtextxy(4,86,"F5 - Load data from disk.");
outtextxy(4,98,"F6 - Alter notes for the graph.");
outtextxy(4,110,"F7 - Plot current graph to disk/plotter (use plotter use LPT1: for
filename)");
outtextxy(4,122,"F8 - Redraw graph");
outtextxy(4,134,"F9 - Toggle on/off cursors.");
outtextxy(4,146,"F10- Save data.");
message("oh dear...I knew there was sumut missing...sorry..press any
key",ONELINE,WAIT,GREEN);
return(helpno);
}

byte setup_graph(void) {
int key,x=1,y, posn[8]={8,88,128,228,296,352,412,452};
int len[8]={26,24,80,50,30,24,22,22};
setviewport(80,136,560,360,1);
setfillstyle(SOLID_FILL,BLUE);
bar(1,1,480,224);
setfillstyle(SOLID_FILL,BLACK);
bar(4,28,476,220);
outtextxy(0,2, "   Line    Line   Screen     Plotter  Line  Limits  Limit Limit ");
outtextxy(0,14,"   Name   on/off  Colour     Colour   Type  on/off  max  min  ");
for (i=0;i<13;i++) displayline(i);
y=0;
do{
 moveto(posn[x]-2,y*14+32);
 setwritemode(1);
 linerel(0,13);linerel(len[x],0);
 key=bioskey(0);
 linerel(-len[x],0);linerel(0,-13);
 setwritemode(0);
 switch (key) {
    case CURLT:       if (!(--x)) x=1;
                      break;
    case CURRT:       if (++x >= 8) x=7;
                      if (x>4 && y>9) x=4;
                      break;
    case CURUP:       if (--y <=0) y=0;
                      break;
    case CURDN:       if (++y >=12) y=12;
                      if (x>4 && y>9) y=9;
                      break;
    case KPMNUS:if (x==2) {setup[y].screen_col--;
                             if (setup[y].screen_col<0) setup[y].screen_col=15;
                             break;}
                      if (x==3) {setup[y].plot_col--;
                             if (setup[y].plot_col < 0) setup[y].plot_col=0;
                             break;}
                      if (x==6) {setup[y].mx=(setup[y].mx-1)%128;break;}
                      if (x==7) {setup[y].mn=(setup[y].mn-1)%128;break;}
    case SPACE:
    case KPPLUS:if (x==1) setup[y].on_off=1-setup[y].on_off;
                      if (x==2) {setup[y].screen_col++;
```

```
                            if (setup[y].screen_col>15) setup[y].screen_col=0;}
                        if (x==3) {setup[y].plot_col++;
                            if (setup[y].plot_col >8) setup[y].plot_col=8;}
                        if (x==4) setup[y].line=1-setup[y].line;
                        if (x==5) setup[y].limits=1-setup[y].limits;
                        if (x==6) setup[y].mx=(setup[y].mx+1)%128;
                        if (x==7) setup[y].mn=(setup[y].mn+1)%128;
    }
 displayline(y);
 }while ((key !=ENTER) && (key !=ESC));
if (datalength >20) findmaxmin();
return(1);
}

void displayline(int line)
{
int posn[8]={8,88,128,228,296,352,412,452};
char onoff[2][4]={"OFF","ON"};
char dotfull[2][5]={"DOT","FULL"};
char pcolor[9][8]
   ={"NO PEN","BLACK","RED","GREEN","BLUE","LitBLUE","PINK","YELLOW","BROWN"};
char scolor[16][13]
  ={"BLACK","BLUE","GREEN","CYAN","RED","MAGENTA","BROWN","LIGHTGRAY",
    "DARKGRAY","LIGHTBLUE","LIGHTGREEN","LIGHTCYAN","LIGHTRED","LIGHTMAGENTA",
    "YELLOW","WHITE"};
int y=line*14+32;
bar(4,y,476,y+12);
outtextxy(posn[0],y,setup[line].value);
outtextxy(posn[1],y,onoff[setup[line].on_off]);
outtextxy(posn[2],y,scolor[setup[line].screen_col]);
outtextxy(posn[3],y,pcolor[setup[line].plot_col]);
outtextxy(posn[4],y,dotfull[setup[line].line]);
if (line<10) { outtextxy(posn[5],y,onoff[setup[line].limits]);
               if (setup[line].mx) outtextxy(posn[6],y,itoa(setup[line].mx,tempstr,10));
               if (setup[line].mn) outtextxy(posn[7],y,itoa(setup[line].mn,tempstr,10));
               }
}


byte setup_comms(void) {
char pos=0;
int oldcom=comms.comno,oldbaud=comms.baud,key;
message("Use left/right & SPACE to set,ENTER to finish",TWOLINE,NOWAIT,BLUE);
outtextxy(64,17,"COM ");
outtextxy(160,17," Baud, 8 bits, No parity, 1 Stop bit");
do{
 setfillstyle(SOLID_FILL,(DARKGRAY*(pos==1)+RED*(pos==0)));
 bar(88,17,100,31);
 setfillstyle(SOLID_FILL,(DARKGRAY*(pos==0)+RED*(pos==1)));
 bar(110,17,160,31);
 outtextxy(92,17,itoa(comms.comno+1,tempstr,10));
 outtextxy(118,17,itoa(comms.baud,tempstr,10));
 key = bioskey(0);
 switch (key){
           case CURLT:        pos=0;break;
           case CURRT:        pos=1;break;
           case SPACE:        if (!pos) comms.comno=(1-comms.comno);
                              else {if (comms.baud >= 19200)       comms.baud=300;
                                    else comms.baud*=2;}
                              break;
           }

 }while (key!=ESC && key!=ENTER);
 if (key==ENTER) return(1);
comms.baud=oldbaud;comms.comno=oldcom;
return(1);
}



byte alter_notes(void) {
char pos=0;
int nlen,key;
do{
 disnotes(pos);
 if (!pos) nlen=strlen(dataxtra.title);
           else nlen=strlen(dataxtra.notes);
 key = bioskey(0);
 switch (key){
           case CURUP:        pos=0;break;
           case CURDN:        pos=1;break;
           case DEL:  if (nlen)
```

```
                                {if (!pos) dataxtra.title[--nlen] = '\0';
                                   else dataxtra.notes[--nlen] = '\0';}
                            break;
          default:   if (isprint(key%0x100))
                                {if (!pos && nlen<0x3E)
                                   {dataxtra.title[nlen++]=(key%0x100);
                                   dataxtra.title[nlen] = '\0';}
                                 if (pos && nlen<0x4E)
                                   {dataxtra.notes[nlen++]=(key%0x100);
                                   dataxtra.notes[nlen] = '\0';}
                                }
  }
 }while (key !=ENTER && key != ESC);
if (key==ENTER) return(1);

return(1);
}

void disnotes(char pos) {
message("Title:-                                                                    notes:-"
                                ,TWOLINE,NOWAIT,BLUE);
setfillstyle(SOLID_FILL,(DARKGRAY*(pos==1)+RED*(pos==0)));
bar(58,1,570,15);
setfillstyle(SOLID_FILL,(DARKGRAY*(pos==0)+RED*(pos==1)));
bar(8,17,632,31);
 setcolor(YELLOW);
 outtextxy(68,1,dataxtra.title);
 outtextxy(10,17,dataxtra.notes);
}


byte plotter(void) {
long tempy,offsetx=1100,offsety=400,ypos,li,t,terr,pow,xpos,yscale=1100;
byte j,anyton=0;
int mn,mx,persec=dataxtra.per_sec;
if (datalength==0) return(1);
if (getfilename(PLOT)) return(1);
setbkcolor(BLACK);
if ((filehd = fopen(filename, "wt")) == NULL) exit(1);
strcpy(tempstr,"Plotting to ");
message(strcat(tempstr,filename),ONELINE,NOWAIT,MAGENTA);
fprintf(filehd,"IN;PA;SI 0.1,0.15;\n");  /* changed from 0.15,.225 */
/* pen1, line type 1..dots  for time grid   */
fprintf(filehd,"SP1;LT1,0.4;DI0,1;LO18;\n");
for (li=((disdatamin/persec)+1)* persec;li<=((disdatamax/persec)* persec);li+=persec) {
 fprintf(filehd,"PU%li,%i;PR;",(offsetx+scale(li,disdatamax,disdatamin,PLOTX)),offsety);
 fprintf(filehd,"LB%s\3\n",itoa(li/persec,tempstr,10));
 fprintf(filehd,"PD0,%i;PA;\n",PLOTY);
 }
fprintf(filehd,"SI 0.1,0.15;PU%li,%li;LO2;LB(C) S.Lawther'93\003\n",offsetx+PLOTX,offsety);
fprintf(filehd,"DI1,0;"); /* taken out SI 0.15,0.225;  */
for (j=0;j<8;j++) {
 if (setup[j].on_off == ON)
  {
  t=data[vtoa(disdatamin)].t[j];
  fprintf(filehd,"PU;LT;SP%i;PU%li,%li;\n",
           setup[j].plot_col,offsetx,(offsety+scale(t,maxtx,mintx,PLOTY)));
  anyton++;
  if (setup[j].line==OFF) fprintf(filehd,"LT2,0.2;\n");
  for (li=disdatamin;li<=disdatamax;li++)
   {
   xpos=scale(li,disdatamax,disdatamin,PLOTX);
   t=data[vtoa(li)].t[j];
   ypos=(offsety + scale(t,maxtx,mintx,PLOTY));
   fprintf(filehd,"PD%li,%li;\n",(xpos+offsetx),ypos);
/*(offsety+scale(t,maxtx,mintx,PLOTY)));*/
   }
  fprintf(filehd,"LO2;LB T%.7s\003\n",setup[j].value);
  if (setup[j].limits == ON)
    {
    fprintf(filehd,"LT1,0.6;");
    mx=setup[j].mx*256;mn=setup[j].mn*256;
    if (mx && mx>mintx && mx<maxtx)
           fprintf(filehd,"PU%li,%li;PR;PD%i,0;PA;\n",
                      offsetx,offsety+scale(mx,maxtx,mintx,PLOTY),PLOTX);
    if (mn && mn>mintx && mn<maxtx)
           fprintf(filehd,"PU%li,%li;PR;PD%i,0;PA;\n",
                      offsetx,offsety+scale(mn,maxtx,mintx,PLOTY),PLOTX);
    }
  }
 }
if (setup[POWS].on_off == ON)
```

```
 {
 pow=data[vtoa(disdatamin)].power;
 if (pow < 0) pow=(-(pow & 0xFF));
 fprintf(filehd,"PU;LT;SP%i;PU%li,%li\n",
          setup[POWS].plot_col,offsetx,(offsety+scale(pow,maxpow,minpow,PLOTY)));
 if (setup[POWS].line==OFF) fprintf(filehd,"LT1;\n");
 for (li=disdatamin;li<=disdatamax;li++)
  {
  xpos=scale(li,disdatamax,disdatamin,PLOTX);
  pow=data[vtoa(li)].power;
  if ( pow < 0) pow=(-(pow & 0xFF));
  fprintf(filehd,"PD%li,%li;\n",
          (xpos+offsetx),offsety+scale(pow,maxpow,minpow,PLOTY));
  }
 }

if (setup[TERRS].on_off == ON)
 {
 terr=terror(disdatamin);
 fprintf(filehd,"PU;LT;SP%i;PU%li,%li\n",
          setup[TERRS].plot_col,offsetx,(offsety+scale(terr,maxe,mine,PLOTY)));
 if (setup[TERRS].line==OFF) fprintf(filehd,"LT1;\n");
 for (li=disdatamin;li<=disdatamax;li++)
  {
  xpos=scale(li,disdatamax,disdatamin,PLOTX);
  terr=terror(li);
  fprintf(filehd,"PD%li,%li;\n",
          (xpos+offsetx),offsety+scale(terr,maxe,mine,PLOTY));
  }
 }

//show bookmarks
if (setup[BOOKMARKS].on_off == ON)
 {
 for (li=disdatamin;li<=disdatamax;li++)
  {
   if ((data[vtoa(li)].bookmark) || (data[vtoa(li)].automark))
          {
          xpos=scale(li,disdatamax,disdatamin,PLOTX);
                    {
                    if (data[vtoa(li)].bookmark) fprintf(filehd,"PU;LT;SP1;");
                                  else fprintf(filehd,"PU;LT;SP2;");
                    fprintf(filehd,"PU%li,%li;PR;PD0,100;PA;",xpos+offsetx,offsety);
                    }
          }
  }
 }

if (setup[STATMARKS].on_off == ON)
 {
 int counts;
 char codes[8][5] = {"ADC","I2C","Offs","Pow","Step","Test","SReq","SAck"};
 fprintf(filehd,"PU;SP%i;",setup[STATMARKS].plot_col);
 for (j=2;j<8;j++)
   if ((data[vtoa(disdatamin)].status >> j) % 2)
outtextxy(scale(li,disdatamax,disdatamin,SCREENX),370- 8 * counts++,codes[j]);
 for (li=disdatamin+1;li<=disdatamax;li++) {
  xpos=scale(li,disdatamax,disdatamin,PLOTX);
  counts=0;
  for (j=2;j<8;j++)
  {
   switch (((data[vtoa(li)].status >> j) %2) - ((data[vtoa(li-1)].status >> j) %2))
          {
          case -1:        fprintf(filehd,"PU%li,%li;",xpos+offsetx,offsety+300*counts++);
                             fprintf(filehd,"PU;LT;");

          fprintf(filehd,"PR;PD0,100;PD-50,100;PD50,-100;PD50,100;PD-50,-100;PD0,150;PA;")
;
                             fprintf(filehd,"LB%s\003\n",codes[j]);
                             break;
          case 0:    break;
          case 1:         fprintf(filehd,"PU%li,%li;",xpos+offsetx,offsety+300*counts++);
                             fprintf(filehd,"PU;LT;");

          fprintf(filehd,"PR;PD0,150;PD-50,-100;PD50,100;PD50,-100;PD-50,100;PD0,100;PA;")
;
                             fprintf(filehd,"LB%s\003\n",codes[j]);

          }
  }
 }
}
```

```
fprintf(filehd,"PU;LT;SP1;PU%li,%li;PR;PD%i,0;PA;\n",offsetx,offsety,PLOTX);
plotln_n_txt(offsetx,offsety-100,decimal(disdatamin,persec),0,1);
plotln_n_txt((offsetx+PLOTX),offsety-100,decimal(disdatamax,persec),0,1);
fprintf(filehd,"PU%li,%li;LBSeconds (approx)\003\n",offsetx+PLOTX/2,offsety-100);
j=0;
if (anyton)
 {
 fprintf(filehd,"SP1;LT1,0.4;LO18;\n");
 tempy=((((maxtx-mintx)/(25*256))!=0)+1)*(5*256);
 for (li=((mintx/tempy)+1)*tempy;li<=((maxtx+1)/tempy)*tempy;li+=tempy) {
  fprintf(filehd,"PU%li,%i;PR;",offsetx,offsety+scale(li,maxtx,mintx,PLOTY));
  fprintf(filehd,"LB%s\3\n",itoa(li/256,tempstr,10));
  fprintf(filehd,"PD%i,0;PA;\n",PLOTX);
  }
 fprintf(filehd,"LT;PU%li,%li;PR;PD0,%i;PA:",yscale,offsety,PLOTY);
 plotln_n_txt(yscale,offsety+PLOTY,decimal(maxtx,256),-1,0);
 plotln_n_txt(yscale,offsety,decimal(mintx,256),-1,0);
 plotln_n_txt(yscale,offsety+PLOTY,"TEMP degC",0,-1);
 if (labs(mintx)==maxtx) {plotln_n_txt(yscale,offsety+(PLOTY/2),"0",-1,0);j++;};
 yscale-=330;
 }
if (setup[POWS].on_off ==ON)
 {
 fprintf(filehd,"SP%i;PU%li,%li;PR;PD0,%i;PA;",
                              setup[POWS].plot_col,yscale,offsety,PLOTY);
 plotln_n_txt(yscale,offsety+PLOTY,itoa((maxpow),tempstr,10),-1,0);
 plotln_n_txt(yscale,offsety,itoa((minpow),tempstr,10),-1,0);
 plotln_n_txt(yscale,offsety+PLOTY,"Power",0,-1);
 if (labs(minpow)==maxpow) {plotln_n_txt(yscale,offsety+(PLOTY/2),"0",-1,0);j++;}
 yscale-=360;
 }
if (setup[TERRS].on_off ==ON)
 {
 fprintf(filehd,"SP%i;PU%li,%li;PR;PD0,%i;PA;",
                              setup[TERRS].plot_col,yscale,offsety,PLOTY);
 plotln_n_txt(yscale,offsety+PLOTY,decimal(maxe,256),-1,0);
 plotln_n_txt(yscale,offsety,decimal(mine,256),-1,0);
 plotln_n_txt(yscale,offsety+PLOTY,"Error",0,-1);
 if (labs(mine)==maxe) {plotln_n_txt(yscale,offsety+(PLOTY/2),"0",-1,0);j++;}
 }
if (csronflag)
 {
 for (i=0;i<2;i++)
  {

fprintf(filehd,"PU%li,%i;PR;",(offsetx+scale(cursor[i].X,disdatamax,disdatamin,PLOTX)),offs
ety);
  fprintf(filehd,"SP4;LO2;LT5,1.2;PD0,%i;",PLOTY);

fprintf(filehd,"PU0,-%li;PU-100,0;PD200,0;PA;",((cursor[i].Y*(long)PLOTY)/(long)SCREENY));
  fprintf(filehd,"DI1,1;LBC%i\003\nDI1,0;",i);
  fprintf(filehd,"PU%i,6900;LBC%i: %s\003\n",3000+(i*2000),i,setup[cursor[i].line].value);
  fprintf(filehd,"PU%i,6770;LBValue:
%s\003\n",3000+(i*2000),dec_and_units(value[i],cursor[i].line));
  fprintf(filehd,"PU%i,6640;LBTime: %smS\003\n",3000+(i*2000),itoa(timet[i],tempstr,10));
  }
 fprintf(filehd,"PU7000,6640;LBdt= %smS\003\n",itoa(timet[1]-timet[0],tempstr,10));
 if (cursor[1].line==cursor[0].line)
    fprintf(filehd,"PU7000,6770;LBd=
%s\003\n",dec_and_units(value[1]-value[0],cursor[1].line));
 }
if (j) fprintf(filehd,"SP1;PU%li,%li;PR;PD%i,0;PA;\n",offsetx,offsety+(PLOTY/2),PLOTX);
fprintf(filehd,"SP1;PU5000,7200;LO6;LB%.80s\003\n",dataxtra.notes);
fprintf(filehd,"PU9000,7200;LO2;LBCOM%i: %i Baud,\003\n",
                                 (dataxtra.comno+1),dataxtra.baud);
fprintf(filehd,"PU9000,7050;LB8 bit,No P,1Sbit\003\n");
fprintf(filehd,"PU9000,6900;LBAT %2i:%2i, ON %2i/%2i/%4i \003\n",
                              smh.ti_hour,smh.ti_min,dmy.da_day,dmy.da_mon,dmy.da_year);
fprintf(filehd,"PU9000,6750;LB%i Readings/sec approx\003\n",dataxtra.per_sec);
fprintf(filehd,"LT;PA;PU0,0;PD;EA10700,7600;PU;");
fprintf(filehd,"LO5;PU5000,7400;SR;LB%.63s\003\nSP0;",dataxtra.title);
fclose(filehd);
return(1);
}


void plotln_n_txt(int x,int y,char* text,char xdir,char ydir) {
fprintf(filehd,"PU%i,%i;",x,y);
fprintf(filehd,"PR;PD %i,%i;",100*xdir,100*(-ydir));
fprintf(filehd,"LO%i;LB%s\003\nPA;PU;\n",(15-(xdir*3)+ydir),text);
}
```

```
char *dec_and_units(int val,int line) {
if (line==POWS) return(strcat(itoa(val,tempstr2,10),"%"));
if (line>=0 && line <=TERRS) return(strcat(decimal(val,256),"deg.C"));
abort();
return(0);
}


/* save data routine (altered 26/10/94)

                      .TDA file format (version 2.0)
byte
00-2F hex - text sting to identify file
30        - min
31        - hour
32        - day
33        - month
34,35     - year
36,37     - comno (I think!)
38,39     - baud
3A-6F     - junk
70-AF     - Title
B0-FF     - notes
100-end   - data
               format:
                      byte    - status byte
                      word 0  - Temp0
                        |         |
                      word 7  - Temp7
                      word 8  - power
                      word 9  - Timer
                      byte    - bookmark (bit 0) + higher timer (bits 1,2)
                      byte    - junk (0)
                      byte    - junk (newline chr)
*/

byte save_data(void) {
int arraypos;
byte j;
if (datalength==0) return(1);
if (getfilename(SAVE)) return(1);
_fmode = O_BINARY;
if ((filehd = fopen(filename,"wb")) == NULL) exit(1);
fprintf(filehd,"Temperature and Texture     S.Lawther 1994 V2.0\n");
fputc(smh.ti_min,filehd);
fputc(smh.ti_hour,filehd);
fputc(dmy.da_day,filehd);
fputc(dmy.da_mon,filehd);
putw(dmy.da_year,filehd);
putw(dataxtra.comno,filehd);
putw(dataxtra.baud,filehd);

/* setup display data to go heresoon */

for (i=10;i<0x40;i++) fputc(0,filehd);
for (i=0;i<0x40;i++) fputc(dataxtra.title[i],filehd);
for (i=0;i<0x50;i++) fputc(dataxtra.notes[i],filehd);
for (i=0;i<datalength+1;i++) {
           arraypos = vtoa(i);
           fputc(data[arraypos].status,filehd);
           for (j=0;j<8;j++) putw(data[arraypos].t[j],filehd);
           putw(data[arraypos].power,filehd);
           putw(data[arraypos].timer,filehd);
           fputc(((data[arraypos].hitimer) + (data[arraypos].bookmark << 2)
                                    + (data[arraypos].automark << 3)),filehd);
           fputc(0,filehd);
           if (i!=datalength) fputc('\n',filehd);
           }

fclose(filehd);
saveflag=0;
return(0);
}

byte qp_data(void) {
char tempfilename[10] = "QPTXXXXXX";
int arraypos;
byte j;
if (datalength==0) return(1);
if (mktemp(tempfilename)==NULL)
            {message("failed mktemp",1,PAUSE,RED);return(0);}
```

```
//this filename needs sorting out
_fmode = O_TEXT;
if ((filehd = fopen(tempfilename,"wt")) == NULL) exit(1);
fprintf(filehd,"min,hour,day,month,year,baud\n");
fprintf(filehd,"%u,%u,%u,%u,%u,%u\n",smh.ti_min,smh.ti_hour,dmy.da_day,dmy.da_mon,dmy.da_ye
ar,dataxtra.baud);
fprintf(filehd,"T4,T5,T6,T7,Time\n");
for (i=0;i<datalength+1;i++) {
          arraypos = vtoa(i);
          for (j=4;j<8;j++) fprintf(filehd,"%u,",data[arraypos].t[j]);
          fprintf(filehd,"%lu\n",data[arraypos].timer+(data[arraypos].hitimer*0x10000));
          }

fclose(filehd);
return(0);
}


byte load_from_disk(void) {
byte j;
i=0;
_fmode = O_BINARY;
if (getfilename(LOAD)) return(1);
if ((filehd = fopen(filename, "rb")) == NULL ) exit(1); /* weird! it was there!*/
strcpy(tempstr,"Loading  ");
message(strcat(tempstr,filename),ONELINE,NOWAIT,BLUE);
len = fread(dat,1,0x30,filehd);
if ((len != 0x30)
  || (strnicmp(dat,"Temperature and Texture     S.Lawther 19",40)))
          {fclose(filehd);
          message(_strerror(NULL),ONELINE,WAIT,RED);
/* {message("File not TDA format! Press any key...",ONELINE,WAIT,RED);*/
          return(1);}
smh.ti_min= fgetc(filehd);
smh.ti_hour=fgetc(filehd);
dmy.da_day= fgetc(filehd);
dmy.da_mon= fgetc(filehd);
dmy.da_year=getw(filehd);
dataxtra.comno=getw(filehd);
if ((dataxtra.baud= getw(filehd))==0) dataxtra.baud=2400;
dataxtra.per_sec=(dataxtra.baud/10)/23;
/* setup graph to go here soon */
for (i=10;i<0x40;i++) fgetc(filehd);
if(fread(dataxtra.title,1,0x40,filehd) != 0x40)
          {fclose(filehd);
          message("File Corrupt.. Press any key...",ONELINE,WAIT,RED);
          return(1);}
dataxtra.title[0x3F]='\0';
if(fread(dataxtra.notes,1,0x50,filehd) != 0x50)
          {fclose(filehd);
          message("File Corrupt...Press any key...",ONELINE,WAIT,RED);
          return(1);}
dataxtra.notes[0x4F]='\0';
/* data in start from here on
Note - data is loaded in from start of data array, so no convertion required*/
i=0;
while ((i<LENGTH) && (!feof(filehd))) {
          data[i].status=getc(filehd);
          for (j=0;j<8;j++)       data[i].t[j]=getw(filehd);
          data[i].power=getw(filehd);
          data[i].timer=getw(filehd);
          j=getc(filehd);
          data[i].hitimer=(j%4);
          data[i].bookmark=(j/4)%2;
          data[i++].automark=(j/8)%2;
          getc(filehd);
          getc(filehd);
          }
datalength=i-1;
disdatamax=i-1;
disdatamin=0;
poszero=0;
if (fclose(filehd) != 0) exit(1);
outtext("ok!");
findmaxmin();
return(0);
}


char getfilename(int service) {
char set_ext[]=".TDA";
int key,keyc=256,flen,flag;
```

```
setcolor(YELLOW);
switch(service){
          case(LOAD):     message("File to Load?",TWOLINE,NOWAIT,BLUE);
                             break;
          case(SAVE):        message("File to Save to?",TWOLINE,NOWAIT,BLUE);
                             break;

          case(PLOT):        message("File to Plot to?",TWOLINE,NOWAIT,BLUE);
                             strcpy(set_ext,".PGL");
                             break;
          }
if(strchr(filename,'.')!=NULL) strcpy(strchr(filename,'.'),"\0");
do{
 setcolor(DARKGRAY);
 setfillstyle(SOLID_FILL,DARKGRAY);
 bar(8,17,630,30);
 setcolor(YELLOW);
 outtextxy(8,17,filename);
 key = bioskey(0);
 if (keyc==256 && ((key==ENTER)|| (key==ESC))) continue;
 if (keyc==256 && key!=DEL) strcpy(filename,"\0");
 keyc=key%0x100;
 flen=strlen(filename);
 while (filename[--flen] == ' ');
 if (key==DEL && (flen+1)) filename[flen] = '\0';
 if ((isalnum(keyc)|| (keyc=='\\') || (keyc=='.')
          ||(keyc==':'&& isalpha(filename[0]) && flen==0)) && flen < MAXPATH-2)
                             {filename[++flen]=toupper(keyc);
                             filename[++flen] = '\0';
                             }
 }while (key!=ESC && (key != ENTER || strlen(filename)==0));
 if (key==ESC) return(1);
flag=fnsplit(filename,drive,dir,file,ext);
if (flag & DRIVE)     setdisk(drive[0]-'A');
if (flag & DIRECTORY)
  {if (strlen(dir)>1) dir[(strlen(dir)-1)]='\0';
  if(chdir(dir))
          {message("Invalid Directory, press any key...",ONELINE,WAIT,RED);
          return(1);}}
if(!(flag & FILENAME))
          {message("No Filename Specified, Press any key...",ONELINE,WAIT,RED);
          return(1);}
if(!(flag & EXTENSION)) strcpy(ext,set_ext);
fnmerge(filename,NULL,NULL,file,ext);
filehd=fopen(filename,"rb");
if(errno!=ENOENT && errno!=EZERO)
  {if (errno != EFAULT)
          {strcpy(tempstr,"Error: ");
          strcpy(tempstr+7,strerror(errno));
          message(tempstr,ONELINE,WAIT,RED);
          return(1);}
   else {message("no disk in drive! Press any key...",ONELINE,WAIT,RED);
          return(1);}}
if (filehd!=NULL && service!=LOAD)
  {fclose(filehd);
  return(tolower(message("file exists!, overwrite? (y/N)"
                                                    ,ONELINE,WAIT,BLUE))!='y');}
if (filehd==NULL && service==LOAD)
  {message("file doesn't exist!, Press any key...",ONELINE,WAIT,BLUE);
  return(1);}
return(0);
}


/* This is the live data-in section, which is almost a program in its self!!

This was rewritten 26/10/94
material guessing code added 21/12/94
*/

byte live_data_in(void) {

register int ap;
char *ptr;
int tlen=0, key=0, keyc=0, garbage=0, delay=50, enterno=32000;
unsigned lasttime=0, hitime=0;
//new stuff for expert sys test
double roomtemp=22.4, heattemp=32.03, present, last1, last2, last3,
part1equ,part2equ,possible;
struct {char name[12];
          double percent,pcstddev;
          int score;} material[13] = {{"Aluminium ",  .90,.03,0},
```

```
                                        {"Card "     ,  .34,.0133,0},
                                        {"Eraser "   ,  .5016,.015,0},
                                        {"Wood "     ,  .275,.0248,0},
                                        {"Glass "    ,  .67,.0191,0},
                                        {"Tufnol "   ,  .55,.014,0},
                                        {"Diecast "  ,  .80,.01,0},
                                        {"Foam (blk)",  .12,.01,0},
                                        {"Foam (wht)",  .15,.01,0},
                                        {"Expanded P",  .07,.01,0},
                                        {"Plastic "  ,  .42,.01,0},
                                        {"hot object",-1.0,.3,0},
                                        {"ice / cold", 2.0,.25,0}};
float
predics[44]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0};
int start=0,startno,matno,count,option;
char matchar;
byte ok=1, pport;
pport=(biosprint(2, 0, 0));
if (comms.baud==2400) delay=100;
setviewport(0,32,639,479,1);
option = message("what option? ENTER - ignore, M - material recogn, C - computer
generate",ONELINE,WAIT,BLUE);
if (option == (ESC%0x100)) return(0);
option=toupper(option);
if (toupper(option) != (ENTER%0x100))
 {
 matchar = message("which material (A,C,E,W,G,T,F,H,I,X,B,D,P)...or ENTER to
ignore",ONELINE,WAIT,BLUE);
 strcpy(tempstr,"ACEWGTDBWXPHI");
 if ((matchar%0x100)==matchar) if ((ptr = strchr(tempstr,toupper(matchar))) == NULL)
{matchar=ENTER; message("Wrong letter, ignored",ONELINE,PAUSE,BLUE);}
 matno=ptr-tempstr;
 if (matchar == (ENTER%0x100)) start=4;
   else if (matchar == (ESC%0x100)) return(0);
            else {
                 do {
                       strcpy(tempstr,"Room Temp = ");
                       strcat(tempstr,gcvt(roomtemp,3,tempstr2));
                       i = message(tempstr,ONELINE,WAIT,BLUE);
                       if (i=='+') roomtemp+=0.1;
                       if (i=='-') roomtemp-=0.1;
                       } while (i!=(ENTER%0x100));
                 }
 }
 else  start=4;
if(initialcomm())
                      if(initialcomm())
                                      return(1); //give it 2 chances to lock on
message(" Sunk!......",ONELINE,NOWAIT,RED);
/* from here on is the main loop */
tempstr[0]='\0';
setviewport(0,0,639,479,1);
if (bioskey(1)) bioskey(0);
i=0;
do
 {
 ap=vtoa(i);

// testint = cQueueLen(comms.comno);
 bar(0,0,40,35);
 outtextxy(1,1,itoa(i,tempstr2,10));
 testint=cQueueLen(comms.comno);
 outtextxy(1,20,itoa(testint,tempstr2,10));
 if ((testint > 200) || (testint < 0)) do{}while (0==0);
 //if its timer data
 data[ap].timer = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno)); // need soln to
wraparound
 if(data[ap].timer < lasttime)
                     hitime++;
 lasttime = data[ap].timer;
 data[ap].hitimer = hitime;

 // clear the marks (to be marked later if required)
 data[ap].bookmark=data[ap].automark=0;
 //mod to get long term data
 //for (count=0;count<207;count++) {ctgetc(comms.comno); if (bioskey(1)==ESC) break;}
 //if its temp0
 data[ap].t[0] = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));
 if(data[ap].t[0] > (42*256)) sound(880);

 //if its temp1
```

```
 data[ap].t[1] = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));

 //if its temp2
 data[ap].t[2] = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));
 if (data[ap].t[2] > (55*256)) sound(880);
 //if its temp3
 data[ap].t[3] = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));

 //if its temp4
 data[ap].t[4] = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));

 //if its temp5
 last3=last2;
 last2=last1;
 last1=present;
 data[ap].t[5] = present = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));
 if ((i>2) && (option=='M')) switch (start) {
           case 0 :   if (fabs(last1 - present) > 100)
                                 {data[ap].automark =1;
                                  start++;
                                  startno= i-1;
                                 }
                           else if (fabs(last2 - present) > 100)
                                 {data[ap].automark =1;
                                  start++;
                                  startno= i-2;
                                 }
                           else if ((fabs(last3 - present) > 100) && (delay==50))
                                 {data[ap].automark =1;
                                  start++;
                                  startno= i-3;
                                 }

                           break;
           case 1 :   part1equ=exp((double)(-(i-startno)*delay)/375.0);  //changed from 850
3/1/95
                           part2equ=(((double)data[ap].t[5]/256.0 - heattemp*part1equ)
/(1-part1equ));

           predics[i-startno]=(possible=((heattemp-part2equ)/(heattemp-roomtemp)));
                           for (matno=0;matno<13;matno++)
                                 {int stddevsout;

                                 stddevsout = ((int)
fabs((possible-material[matno].percent)
                                                                 /
(material[matno].pcstddev)));
                                 if (stddevsout < 3)
material[matno].score+=(9-(stddevsout*stddevsout))*(1+((i-startno) / (20/(delay/50))));
                                 }
                           if ((i-startno) > 40/(delay/50)) start++;
                           break;
           case 2:    if (1)
                                 {int a,b=0;
                                 start=3;
                                 for (a=1;a<13;a++) if (material[a].score >
material[b].score) b=a;
                                 //outtextxy(40,1,material[b].name);
                                 for (a=0;a<13;a++)
                                       {outtextxy(40,(1+a*10),material[a].name);

outtextxy(120,(1+a*10),itoa(material[a].score,tempstr2,10));
                                        if (a==b) outtextxy(150,(1+a*10),"*");
                                       }

                                 }

           default:        break;
                           }
 if ((option=='C') && (start == 1))
  {
  float absvalue;
  cputc(comms.comno,'A');
  part1equ = exp((double)(-(i-startno) * delay)/375.0);
  part2equ = material[matno].percent * part1equ + (1-material[matno].percent);
  absvalue = part2equ * ((heattemp) - roomtemp) + roomtemp;
  cputc(comms.comno,((int)(absvalue*4.0)%256));
  if ((i-startno) ==35)
                           start=2;
  }
 //if its temp6
 data[ap].t[6] = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));
```

```
 DrainOutQueue(comms.comno);

 //if its temp7
 data[ap].t[7] = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));
 DrainOutQueue(comms.comno);

 //if its power data
 data[ap].power = ((ctgetc(comms.comno)<<8)+ctgetc(comms.comno));

 //if its status data, and the gash (syncing) bytes
 data[ap].status = (ctgetc(comms.comno));
 //time to update screen?
 //if ((i%(comms.baud/120)) == 0) updatescreen(garbage);
 //Check parallel port (Out of Paper error pin) for bookmark
 if (pport!=(biosprint(2, 0, 0))) {data[ap].bookmark = 1;pport=(biosprint(2, 0, 0));}

 if ( ((ctgetc(comms.comno)!=0) && bioskey(1)!=ESC)
                      || ((ctgetc(comms.comno)!=255) && bioskey(1)!=ESC) ) ok=0;

 if (++i >= LENGTH)
           {
           i=LENGTH-1;
           if (++poszero >= LENGTH) poszero=0;
           }


   if ((bioskey(1)) || (enterno == i))
           {
           if (enterno != i) key=bioskey(0);
                     else key=ENTER;
           garbage=0;
           tlen=strlen(tempstr);
           switch(key)
                     {
                     case ESC:       break;
                     case ENTER:     //time to decode and send instruction
                                     if (tlen == 0) {key = 0; break;}
                                     garbage = decode_instr();
                                     if ((option == 'C') && start==1) start=2;
                                     tempstr[0]='\0';       //clear tempstr and set
automark
                                     if (!garbage) data[ap].automark = 1;
                                     key = 0;
                                     break;

                     case TAB:       if ((tlen == 0) && (option =='C'))
                                            {key = 0;start=1;startno=i;
                                            data[ap].automark = 1;
                                            }
                                     break;

                     case DEL:       if (tlen != 0) tempstr[--tlen] = '\0';
                                     if ((tlen > 3) && tempstr[tlen-2] == '.')
                                                    tempstr[--tlen] = '\0';
                                     key = 0;
                                     break;

                     case SPACE:      data[ap].bookmark = 1;
                                     key = 0;
                                     break;

                     case F1:         randomize();
                                     enterno = (i + (random(450)));
                                     key = 0;
                                     data[ap].automark=1;
                                     break;
                                     //need to sort out nos after d.p. and how
                                     // to get negative nos

                     default:         if ((keyc=toupper(key%0x100)) == 0) {key = 0;break;}
                                     if (tlen == 0)         {
                                                    if (strchr("JOTAPSCB@X",keyc) !=NULL)
break;
                                                           else {key=0;break;}
                                                    }
                                     switch(tempstr[0])
                                            {
                                            case 'X':
                                            case '@':
                                            case 'S':
                                            case 'C':       key=0;
                                                           break;
```

```
                                                case 'B':
                                                case 'P':       if ((tlen < 4) &&
isdigit(keyc)) break;
                                                                //to get here key must be bad
                                                                key= 0;
                                                                break;
                                                case 'J':
                                                case 'O':       if ((tlen==1) && (keyc == '-'))
break;
                                                case 'T':
                                                case 'A':        if (tempstr[tlen-1] == '.')
                                                                        {
                                                                        switch(keyc)
                                                                         {
                                                                         case '0':
tempstr[tlen++]='0';
                                                                                  break;
                                                                         case '2':
tempstr[tlen++]='2';
                                                                                  keyc = '5';
                                                                                  break;
                                                                         case '5':
tempstr[tlen++]='5';
                                                                                  keyc = '0';
                                                                                  break;
                                                                         case '7':
tempstr[tlen++]='7';
                                                                                  keyc = '5';
                                                                                  break;
                                                                         default:  key=0;
                                                                         }
                                                                        break;
                                                                        }
                                                                if (keyc=='.')
                                                                        {
                                                                        if (strchr(tempstr,'.')
!= NULL) {key = 0;break;}
                                                                        if
(!isdigit(tempstr[tlen-1])) tempstr[tlen++]='0';
                                                                        break;
                                                                        }
                                                                if (isdigit(keyc) && ((tlen <
3) || ((tlen < 4) && (tempstr[1] == '-'))))
                                                                                break;
                                                                key=0;
                                                }


                        }
                if (key)
                        {
                        tempstr[tlen++] = keyc;
                        tempstr[tlen] = '\0';
                        }

                bar(10,350,50,370);
                if (garbage) outtextxy(10,350,"Eh???");
                else outtextxy(10,350,tempstr);
                }
 DrainOutQueue(comms.comno);
 } while(ok && (key != ESC));

if (key == ESC) {cputc(comms.comno,'C');nosound();}
cclose(comms.comno);
if (!ok) {message("comms error, I guess!..press any key",ONELINE,WAIT,BLACK);}

if (i < 10) return(1);
datalength=i-2;
disdatamin=0;disdatamax=i-2;
gettime(&smh);
getdate(&dmy);
dataxtra.baud=comms.baud;
dataxtra.comno=comms.comno;
dataxtra.per_sec=(comms.baud/10)/23;

/*  note that ints garage and startno, and floats part1equ and part2equ
 are recycled here for this development code */
if (start ==3)
                {
                part1equ=part2equ=0.0;
                startno=garbage=20/(delay/50);
                if ((filehd = fopen("RECOGRES.DAT","at")) == NULL) exit(1);
```

```
           for (i=0;i<13;i++) fprintf(filehd,"%s,",material[i].name);
           fprintf(filehd,"roomtemp,heattemp,delay,avg sec1,avg sec2,time,day,act matl\n");
           for (i=0;i<13;i++) fprintf(filehd,"%u,",material[i].score);
           fprintf(filehd,"%4.2f,%4.2f,%i,",roomtemp,heattemp,delay);
           for (i=0;i<garbage;i++)
                              {
                              part2equ+=predics[i+garbage];
                              if (predics[i] != 0.0) part1equ+=predics[i];
                                    else startno--;
                              }
           fprintf(filehd,"%5.3f,%5.3f,",part1equ/startno,part2equ/garbage);
           fprintf(filehd,"%u%u,%c\n",smh.ti_hour,smh.ti_min,matchar);
           for (i=0;i<(garbage*2);i++) fprintf(filehd,"%4.2f,",predics[i]);
           fprintf(filehd,"\n");
           fclose(filehd);
           }

saveflag=1;
findmaxmin();
return(0);

}


/*
void updatescreen(int whichmessage)
 {
 bar(0,370,640,448);
 outtextxy(10,375,"  Heater Temp=   °C    Touch Temp=     °C   Steady Temp=     °C");
 outtextxy(10,400,"Heatsink Temp=   °C  Peltier Temp=     °C   Finger Temp=     °C");
 outtextxy(10,425,"        Power=     Queue Length=");
/**************i got to here       */
/* } */




int decode_instr(void) {
byte firstbyte,secondbyte;
int novalue, decs=0, tlen;
int garbage=1;
firstbyte = tempstr[0];
tlen=strlen(tempstr);
switch(tempstr[0])
           {
           case 'J':  //its a jump +/- X from current value(holds value)
           case 'O':  //its an offset from current value(no hold)
                              //2nd byte to be Suuuuudd S=sign, u=unit, d=decimal
                              // ie +/-31.75 max
                              if (tlen < 2) break;
                              if (tempstr[1] == '-')
                                    {
                                    sscanf(tempstr,"%*c-%3d.%2d",&novalue,&decs);
                                    if ((novalue > 31) || (tlen < 3))  break;
                                    novalue+= 32;
                                    }
                                 else {
                                    sscanf(tempstr,"%*c%3d.%2d",&novalue,&decs);
                                    if (novalue > 31) break;
                                    }
                              cputc(comms.comno,firstbyte);
                              cputc(comms.comno,(novalue << 2) + (decs / 25)%4);
                              garbage=0;
                              break;

           case 'T':  //sets value of tsteady
           case 'A':  //sets an absolute value(holds value)
                              //2nd byte to be uuuuuudd u=unit, d=decimal
                              // ie 0 to 63.75 max
                              if (tlen < 2) break;
                              if (tempstr[1] == '-') break;
                              sscanf(tempstr,"%*c%3d.%2d",&novalue,&decs);
                              if (novalue > 63) break;

                              cputc(comms.comno,firstbyte);
                              cputc(comms.comno,(novalue << 2) + (decs / 25)%4);
                              garbage=0;
                              break;


           case 'P':  //sets maximum proportional power level
```

```
           case 'B':       //sets maximum bang-bang power level
                              //2nd byte to be pppppppp p=power ie 0 to 255
                              sscanf(tempstr,"%*c%d",&novalue);
                              if ( (tlen > 1) && (novalue >=0) && (novalue < 256) )
                                           {
                                           cputc(comms.comno,firstbyte);
                                           cputc(comms.comno,novalue);
                                           garbage=0;
                                           }

                              break;

           case 'X':
           case '@':  //sets new baud rate (and causes the comms reading to die)
           case 'S':  //sets absolute value of 5' C(holds value)
           case 'C':  //clears effects of jump, offset, absolute and shock
                              //no 2nd byte
                              cputc(comms.comno,firstbyte);
                              garbage=0;
           }
return(garbage);
}



int initialcomm(void){
int lastdatain=0,datain, check;
byte rawin[23]= {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
byte count=0,i,sync,synccount;
message("Trying to open comms..",ONELINE,NOWAIT,BLUE);
check = copen(comms.comno,CBUFFER,comms.baud,NONE,DATA8,STOP1,0,0,0);
if (check !=0)
 {
  message("comms open failed.. press any key..",ONELINE,WAIT,BLACK);
  return(1);
 }
/* get rid of 1st 10 bytes in case of a framing error and to check
connection to THE LUMP*/
cputc(comms.comno,25);
for (i=0;i<10;i++)  ctgetc(comms.comno);
           if ((lastdatain=ctgetc(comms.comno)) <0)
           {
               cclose(comms.comno);
               if (bioskey(1)==ESC)
                      message("escape'd ",ONELINE,PAUSE,BLACK);
               else message("Timed out on comms, press any key...",ONELINE,WAIT,BLACK);
           return (-11);
           }

setviewport(1,1,639,479,0);
do
 {
 synccount=0;
 for (i=0;i<23;i++){
  if ((datain=ctgetc(comms.comno)) <0 )
  {
      cclose(comms.comno);
      message("Timed out / Escape pressed! ..press any key.. ",ONELINE,WAIT,BLACK);
      return(-11);
  }
  outtextxy((24*i),count*8,itoa(datain,tempstr,10));
  if ((datain==255) && (!lastdatain) && (rawin[i]) )
                     {rawin[i]=1;
                     synccount++;
                     sync=i;}
           else rawin[i]=0;
  lastdatain=datain;
  }
 if (synccount==0) for (i=0;i<23;i++) rawin[i]=1;
 }
 while ((synccount!=1) && (++count != 128) && (!(bioskey(1)==ESC)));
if (bioskey(1)==ESC) {bioskey(0);return(-10);}
if (count ==128)
           {cclose(comms.comno);
           message("Can't sync in to data packet!..press any key",ONELINE,WAIT,BLACK);
           return(-13);}
if (sync==0) return(0);
for (i=0;i<(sync+1);i++) if (ctgetc(comms.comno) < 0)
                              {
                              cclose(comms.comno);
                              message("Failed on sunk unwind...press any
key",ONELINE,WAIT,BLACK);
```

```
                                return(-13);
                                }
return(0);}


void line_n_text(int x,int y,char* text,char xdir,char ydir) {
int tx,ty;
moveto(x,y);
linerel(4*xdir,6*ydir);
tx = textwidth (text);
ty = textheight(text);
if (xdir == 1) moverel (0,0);
if (xdir == 0) moverel ((-tx/2),0);
if (xdir ==-1) moverel ((-tx),0);  /* add 4 */
if (ydir == 1) moverel (0,2);
if (ydir == 0) moverel (0,(-ty/2));
if (ydir ==-1) moverel (0,(-2-ty));
outtext(text);
}



char message(char *mess,char posn,char wait,int col) {
long tx;
winmangled=1;
setviewport(1,1,639,33,1);
setfillstyle(SOLID_FILL,col);
bar(1,1,639,33);
settextstyle(SMALL_FONT,HORIZ_DIR,5);
tx=textwidth(mess);
outtextxy((320-(tx/2)),posn,mess);
if (bioskey(1)) bioskey(0);
if (wait==PAUSE) while (tx>0) tx+=8192;
if (wait==WAIT)       return(bioskey(0)%0x100);
return(0);
}


int vtoa(int virtno) {
virtno=virtno%LENGTH;
return((virtno+poszero)%LENGTH);
}
```

ROUTINE - ASYNCH.C 15/1/95

```
/*
 * ASYCNCH.C A simple, low-level, interrupt driven comm driver for
 * Turbo/Borland C. No assembly required!
 *
 * Portions (C) Copyright 1991 Steve Resnick, Asylum Software.
 * License granted for personal or educational use.
 * Commercial use licensed only through registration with the author.
 *
 * Steve Resnick - 530 Lawrence Expressway, Box 374, Sunnyvale, Ca 94086
 * resnicks@netcom.com, steve@apple!camphq,
 * steve.resnick@f105.n143.z1.FIDONET.ORG
*/
#include <stdio.h>
#include <dos.h>
#include <bios.h>
#include "asynch.h"

#define ESC        0x011B

/*
 * This table is used to translate a BPS rate to a baud rate divisor for the
 * UART
*/
struct _baud
{
         word     Rate, Divisor;
} BaudTable[] =
{
         110,1040,150,768,300,384,600,192,1200,96,2400,48,4800,24,9600,12,
         19200,6,         /* I have no idea if this works */
} ;
#if DEBUG
byte PICVals[2][2];       /* Mirror PIC values to these arrays */
byte DataTable[2][20];          /* Mirror UART values to these arrays */
byte portinb(word port);  /* Use our own port I/O functions */
void portoutb(word port, byte data);
#else
#define portinb(p) inportb(p)
#define portoutb(p,b) outportb(p,b)
#endif
uart cports[4] = {COM1_DEFAULTS,COM2_DEFAULTS};

/*
 * SetPortParms sets up the UART line parameters and baud rate.
 * If Base, IRQ, and BreakLength are specified as 0, defaults are used.
 * (See ASYNCH.H)
 * This may only be called for an open port.
*/
int SetPortParms(int PortNo, int Baud, int Parity, int Data, int Stop,
                    int Base, int IRQ, int BreakLength)
{
         int i, b;
         if (PortNo > 3 || PortNo < 0)
                  return EINVPORT;
         if (cports[PortNo].status & (PS_ENABLED == 0))
                  return EPORTNOTOPEN;

         if (Base)
                  cports[PortNo].addr = Base;
         if (IRQ)
                  cports[PortNo].irqline = IRQ;
         if (BreakLength)
                  cports[PortNo].break_length = BreakLength;

         cports[PortNo].parms = 0;
```

```
        cports[PortNo].parms |= Data;
        cports[PortNo].parms |= Stop;
        cports[PortNo].parms |= Parity;
        if (Baud != 0)
        {
                for (i = 0, b = -1; i < dim(BaudTable); i++)
                        if (BaudTable[i].Rate == Baud)
                                b = i;
                if (b == -1)
                        return EINVBAUDRATE;

                cports[PortNo].baud = BaudTable[b].Divisor;
                InitBaud(PortNo);
        }
        else if (cports[PortNo].baud == 0)
                return EINVBAUDRATE;
        portoutb(cports[PortNo].addr+LCR_REG,cports[PortNo].parms);
        return 0;
}
/*
 * copen opens a comm port spcified by PortNo. Input and Output buffers are
 * allocated according to the BufSiz parameter.
 * If Base, IRQ, and BreakLength are specified as 0, defaults are used.
 * (See ASYNCH.H)
 *
 * When the port is opened, the following takes place:
 * Buffer memory is allocated
 * CommPort options are set
 * The interrupt vector is set based on IRQ+8
 * The 8259 is enabled to recieve interrupts.
 * The 8250 IE mask is set
 * The 8250 OUT2 is asserted, enabling the interrupt line from the UART to the
 * PIC.
*/
int copen(int PortNo, int BufSiz, int Baud, int Parity, int Data, int Stop,
                      int Base, int IRQ, int BreakLength)
{
        uart * Cp;
        int Rc;
        if (PortNo > 3 || PortNo < 0)
                return EINVPORT;
        if (cports[PortNo].status & PS_ENABLED)
                return EPORTALREADYOPEN;
        Cp = &cports[PortNo];
        if ((Cp->InBuffer = Qalloc(BufSiz)) == NULL)
                return EMEMALLOC;
        if ((Cp->OutBuffer = Qalloc(BufSiz)) == NULL)
        {
                Qfree(Cp->InBuffer);
                return EMEMALLOC;
        }
        Cp->status |= PS_ENABLED;
        Rc = SetPortParms(PortNo,Baud,Parity,Data,Stop,Base,IRQ,BreakLength);
        if (Rc)
        {
                Qfree(Cp->InBuffer);
                Qfree(Cp->OutBuffer);
                Cp->status = 0;
                return Rc;
        }
        SetIntVector(PortNo);
        disable();
        SetPIC(PortNo,1);
        SetUARTIe(PortNo);
        enable();
        ControlDTR(PortNo,ON);
        return 0;
}
/*
 * cclose closes a serial port by disabling interrupts on the UART, then on
 * the PIC and finally, returns the original vector to the IVT.
```

```
*/
int cclose(int PortNo)
{
          uart * Cp;
          /*int Rc;  */
          if (PortNo > 3 || PortNo < 0)
                    return EINVPORT;
          if (cports[PortNo].status & (PS_ENABLED == 0))
                    return EPORTNOTOPEN;
          Cp = &cports[PortNo];
          disable();
          ClrIntVector(PortNo);
          SetPIC(PortNo,0);
          enable();
          Cp->status = 0;
          Qfree(Cp->InBuffer);
          Qfree(Cp->OutBuffer);
          portoutb(Cp->addr + IE_REG,0);
          portoutb(Cp->addr + MCR_REG,1);
          return 0;
}
/*
 * InitBaud sets the baudrate on the UART by setting the divisor latch access
 * bit (DLAB) then writing the divisor's low byte,then the divisors high byte.
 * For baudrates >= 600 BPS, the high byte should be 0. (See ASYNCH.H)
*/
int InitBaud(int PortNo)
{
          uart * Cp;
          byte bdata;
          if (PortNo > 3 || PortNo < 0)
                    return EINVPORT;
          if (cports[PortNo].status & (PS_ENABLED == 0))
                    return EPORTNOTOPEN;
          Cp = &cports[PortNo];
          bdata = portinb(Cp->addr + LCR_REG);
          portoutb(Cp->addr + LCR_REG,bdata | DLAB);
          portoutb(Cp->addr + DIV_LOW,LOBYTE(Cp->baud));
          portoutb(Cp->addr + DIV_HI, HIBYTE(Cp->baud));
          portoutb(Cp->addr + LCR_REG,Cp->parms);
          return 0;
}
/*
 * SetPIC sets or clears the interrupt enable mask on the PIC for a
 * particular interrupt. This function will work for both the master
 * and the slave 8259A's in an AT.
*/
SetPIC(int PortNo, int State)
{
static    int States[] = {-1,-1,-1,-1};
          int PICAddr = PIC;
          byte mask;
          if (PortNo > 3 || PortNo < 0)
                    return EINVPORT;
          if(State == 0 && States[PortNo] == -1)
                    return -1;
          if (cports[PortNo].irqline > 7)
          {
                    mask = ( 1 << (cports[PortNo].irqline - 8));
                    PICAddr = 0xA0;
          }
          else
                    mask = (1 << cports[PortNo].irqline);

          if (State == 0)
          {
                    States[PortNo] = portinb(PIC+1);
                    portoutb(PICAddr+1,(byte)States[PortNo]|mask);
                    return 0;
          }
          mask = ~mask;
```

```
          States[PortNo] = portinb(PICAddr+1);
          portoutb(PICAddr+1,States[PortNo] & mask);
          portinb(PICAddr+1);
          return 0;
}
/*
 * EOI Sends a non-specific end of interrupt to the 8259A specific to the
 * IRQ line of a com port
*/
int EOI(int PortNo)
{
          if (PortNo > 3 || PortNo < 0)
                    return EINVPORT;
          if (cports[PortNo].status & (PS_ENABLED == 0))
                    return EPORTNOTOPEN;
          if (cports[PortNo].irqline > 7)
                    portoutb(0xA0,0x20);
          else
                    portoutb(0x20,0x20);
          return 0;
}
/*
 * This is the interrupt handler for ALL comm ports. Each ISR for a comm
 * port calls this routine passing the index into the UART definition array
 * so that values may be extracted for things like UART address, etc.
 *
 * This is a re-entrant function, although interrupts are disabled to ensure
 * that interrupts are handled one at a time for each UART. This may be
 * over-kill and may need to be changed in the future.
*/
void IsrHandler(int PortNo)
{
          uart * Cp;
          byte Idv, IIDvalue=0;
          disable();
          Cp = &cports[PortNo];
          IIDvalue = portinb(Cp->addr+ IID_REG);
          if ((IIDvalue & IID_PENDING) == 0)
          {
                    IIDvalue &= IID_MASK ;
                    Idv = IIDvalue >> 1;
                    if (Idv & IID_DATA)
                              QinsertChar(Cp->InBuffer,portinb(Cp->addr));

                    if (Idv & IID_TEMPTY)
                              DrainOutQueue(PortNo);

                    if (Idv & IID_LSTAT)
                              Cp->lstat = portinb(Cp->addr+ LSR_REG);

                    if (Idv & IID_MSTAT)
                              Cp->mstat = portinb(Cp->addr+ MSR_REG);

                    portinb(Cp->addr+5);
                    portinb(Cp->addr+6);
          }
          enable();
          SetUartOUT2(PortNo);
          EOI(PortNo);
}
/*
 * Dummy ISR's
 * Each comm port needs to have it's own ISR, but we need only one real
 * handler. Since an ISR shared my multiple interrupts cannot determine
 * it's source, we set up these dummy ISR's to call IsrHandler specifying
 * the port which needs service.
*/
void interrupt com1isr()
{
          IsrHandler(0);
}
```

```
void interrupt com2isr()
{
         IsrHandler(1);
}
void interrupt com3isr()
{
         IsrHandler(2);
}
void interrupt com4isr()
{
         IsrHandler(3);
}
/*
 * SetIntVector sets the appropriate interrupt vector (correctly for both
 * (master and slaved IRQ's) based on the IRQ. The original vector is saved
 * in the UART structure so that it may be reset upon termination.
*/
SetIntVector(int PortNo)
{
         int NewInt;
static   void interrupt (*isrs[])() = {com1isr,com2isr,com3isr,com4isr};
         if (PortNo > 3 || PortNo < 0)
                  return EINVPORT;
         if (cports[PortNo].status & (PS_ENABLED == 0))
                  return EPORTNOTOPEN;
         cports[PortNo].oldisr = getvect(IRQINT(cports[PortNo].irqline));
         NewInt = IRQINT(cports[PortNo].irqline);
         setvect(NewInt,isrs[PortNo]);
         return 0;
}
/*
 * ClrIntVector revectors interrupts back to the original vector before we
 * loaded up.
*/
ClrIntVector(int PortNo)
{
         if (PortNo > 3 || PortNo < 0)
                  return EINVPORT;
         if (cports[PortNo].status & (PS_ENABLED == 0))
                  return EPORTNOTOPEN;
         setvect(IRQINT(cports[PortNo].irqline),cports[PortNo].oldisr);
         return 0;
}
/*
 * SetUARTIe sets the interrupt enable mask on the UART.
 * SetUartOUT2 is then called to unmask interrupts from the UART to the PIC
*/
SetUARTIe(int PortNo)
{
         int i;
         word IntFlags = IIV_LSTAT | IIV_RDATA | IIV_MSTAT | IIV_XMITE;
         if (PortNo > 3 || PortNo < 0)
                  return EINVPORT;
         if (cports[PortNo].status & (PS_ENABLED == 0))
                  return EPORTNOTOPEN;
         for (i = 5; i < 8; i++)
                  inportb(cports[PortNo].addr+i);

         portoutb(cports[PortNo].addr + IE_REG, IntFlags);
         SetUartOUT2(PortNo);
         return 0;
}
/*
 * SetUARTOut2 set's the OUT2 line on the UART high, allowing interrupts
 * to pass from the UART to the PIC (thanks, IBM)
*/
SetUartOUT2(int PortNo)
{
         byte IoData;
         IoData = portinb(cports[PortNo].addr + MCR_REG);
         portoutb(cports[PortNo].addr + MCR_REG,IoData | UARTIEMASK);
```

```
        return 0;
}
/*
 * DrainOutQueue reads characters from the output queue to the UART, as
 * long as the UART is ready for data. If it is not ready, we return
 * immediately.
*/
DrainOutQueue(int PortNo)
{
        byte IoData;
        char Ch;
        if (PortNo > 3 || PortNo < 0)
                return EINVPORT;
        if (cports[PortNo].status & (PS_ENABLED == 0))
                return EPORTNOTOPEN;

        while(1)
        {
                IoData = portinb(cports[PortNo].addr + LSR_REG);
                if (IoData & 0x20 || IoData & 0x40)
                {
                        Ch = QreadChar(cports[PortNo].OutBuffer);
                        if (Ch == -1)
                                break ;
                        portoutb(cports[PortNo].addr,Ch);
                        continue ;
                }
                break ;
        }
        return 0;
}
/*****************************************************************************
 ***************************** User Functions ******************************/
/*
 * cQueueLen calculates the no of byte waiting in I/P queue returns it,
 *
*/
cQueueLen(int PortNo)
{
        if (PortNo > 3 || PortNo < 0)
                return -10 - EINVPORT;
        if (cports[PortNo].status & (PS_ENABLED == 0))
                return -10 - EPORTNOTOPEN;
        return QLen(cports[PortNo].InBuffer);
}
/*
 * cgetc reads a character from the input queue and returns it, or -1 if no
 * character is available.
*/
cgetc(int PortNo)
{
        if (PortNo > 3 || PortNo < 0)
                return -10 - EINVPORT;
        if (cports[PortNo].status & (PS_ENABLED == 0))
                return -10 - EPORTNOTOPEN;
        return QreadChar(cports[PortNo].InBuffer);
}
/*
 * ctgetc reads a character from the input queue and returns it; if no chr
 * is available, it waits for one or until timeout, when it returns -1
*/
ctgetc(int PortNo)
{
        int Ch;
        int Timeout=0;
        if (PortNo > 3 || PortNo < 0)
                return -10 - EINVPORT;
        if (cports[PortNo].status & (PS_ENABLED == 0))
                return -10 - EPORTNOTOPEN;
        do
        {
```

```
                Ch=QreadChar(cports[PortNo].InBuffer);
        }while ((Ch == -1) && ((Timeout+=16) > 0) && (bioskey(1)!=ESC));
        return Ch;
}
/*
 * cputc writes a character to the output queue
*/
cputc(int PortNo, char Ch)
{
        if (PortNo > 3 || PortNo < 0)
                return -10 - EINVPORT;
        if (cports[PortNo].status & (PS_ENABLED == 0))
                return -10 - EPORTNOTOPEN;

        QinsertChar(cports[PortNo].OutBuffer,Ch);
        DrainOutQueue(PortNo);
        return 0;
}
/*
 * SendBreak sets a break condition on the serial line and holds it
 * for break_lenght miliseconds
*/
int SendBreak(int PortNo)
{
        byte IoData;
        if (PortNo > 3 || PortNo < 0)
                return EINVPORT;
        if (cports[PortNo].status & (PS_ENABLED == 0))
                return EPORTNOTOPEN;
        IoData = portinb(cports[PortNo].addr + LCR_REG);
        portoutb(cports[PortNo].addr + LCR_REG,IoData | 0x40);
        delay(cports[PortNo].break_length);
        portoutb(cports[PortNo].addr + LCR_REG,IoData);
        return 0;
}
/*
 * ControlDTR raises or lowers DTR on a specific port
*/
int ControlDTR(int PortNo, int State)
{
        byte IoData;
        if (PortNo > 3 || PortNo < 0)
                return EINVPORT;
        if (cports[PortNo].status & (PS_ENABLED == 0))
                return EPORTNOTOPEN;
        IoData = portinb(cports[PortNo].addr + MCR_REG);
        if (State == ON)
                portoutb(cports[PortNo].addr + MCR_REG,(byte)(IoData | 1));
        else
                portoutb(cports[PortNo].addr + MCR_REG,(byte)(IoData &
~1)&0xFF);
        return 0;
}
/*****************************************************************************
***************************** Debugging Functions *************************/
#if DEBUG
void portoutb(word port, byte data)
{
        int Idx = port - 0x3f8;
        if (Idx > -1 && Idx < 20)
                DataTable[0][Idx] = data;
        if (port >0x1F && port < 0x22)
        {
                Idx = port - 0x20;
                PICVals[0][Idx] = data;
        }
        outportb(port,data);
}
byte portinb(word port)
{
        int Idx = port - 0x3F8;
```

```
        byte Data;
        Data = inportb(port);
        if (port >0x1F && port < 0x22)
        {
                Idx = port - 0x20;
                PICVals[1][Idx] = Data;
        }
        else if (Idx > -1 && Idx < 20)
                DataTable[1][Idx] = Data;
        return Data;
}
#endif
```

ROUTINE - ASYNCH.H 29/10/94

```
#if !defined(ASYNCH_INCLUDED)    /* Prevent multiple includes */
#define ASYNCH_INCLUDED
#include "circleq.h"
typedef  unsigned char    byte;
typedef unsigned short  word;
typedef unsigned long     dword;
typedef struct _lcr
{
          word     databits     : 2;
          word     stopbits     : 1;
          word    parity        : 2;
          word     stuckparity  : 1;
          word     breakenable  : 1;
          word     dlab         : 1;
} lcr;
typedef struct _uart
{
          word     addr;
          byte     irqline;
          word     baud;
          byte     lstat;

          byte     mstat;
          byte     status;
          byte     break_length;
          byte     parms;
          CircleQ *InBuffer;
          CircleQ *OutBuffer;
          void interrupt (*oldisr)();
} uart;
#define LineStatus(n) ((n >= 0 && n <4)  ? cports[n].lstat : EINVPORT)
#define ModemStatus(n) ((n >= 0 && n <4)  ? cports[n].mstat : EINVPORT)

extern uart cports[4];
#define COM1_DEFAULTS\
          {0x3F8,4,0,0,0,0,250,\
          DATA8|STOP1|NONE,\
          NULL,NULL}

#define COM2_DEFAULTS \
          {0x2F8,3,0,0,0,0,250,\
          DATA8|STOP1|NONE,\
          NULL,NULL}

#define PIC        0x20               /* 8259A PIC Address */
#define IRQINT(n) ((n < 9) ? n+8 : (n-9)+0x70)
                                      /* Make int value from IRQ */
#define IID(n)     (n & 7)            /* Get Interrupt ID from reg val */
/*
 * Define Interrupt ID values from the interrupt ID register
*/
#define IID_PENDING0x01
#define IID_LSTAT  0x00               /* Line status change */
#define IID_DATA   0x02               /* Data is available from 8250 */
#define IID_TEMPTY 0x04               /* Transmitter buffer empty */
#define IID_MSTAT  0x06               /* New modem status */
#define IID_MASK   0x07

#define IIV_LSTAT  0x01
#define IIV_XMITE  0x02
#define IIV_RDATA  0x04
#define IIV_MSTAT  0x08
/*
 * Define some offsets from the UART base address for various registers
*/
```

```
#define XMT_REG           0x00         /* Transmitter holding buffer */
#define RCV_REG           0x00         /* Reciever holding buffer */
#define DIV_LOW           0x00         /* Divisor low byte when DLAB=1 */
#define DIV_HI            0x01         /* Devisor high byte when DLAB=1 */

#define IE_REG            0x01         /* Interrupt enable register */
#define IID_REG           0x02         /* Interrupt ID register */
#define LCR_REG           0x03         /* line control register */
#define MCR_REG           0x04         /* Modem control register */
#define LSR_REG           0x05         /* line status register */
#define MSR_REG           0x06         /* modem status register */

/*
 *       Stuff for the line control register
*/
#define NONE       0x00                /* No Parity */
#define ODD        0x08                /* Odd Parity */
#define EVEN       0x18                /* Even Parity */
#define STOP1      0x00                /* Need I explain these? */
#define STOP2      0x04
#define DATA5      0x00
#define DATA6      0x01
#define DATA7      0x02
#define DATA8      0x03
#define DLAB       0x80

#define UARTIEMASK 0x08          /* Mask used to enable interrupts on
                                    the 8250 OUT2 line */
#define PS_ENABLED 0x01
#define PS_INTERRUPT      0x02         /* Not used */

enum errors {EPORTNOTOPEN=1,EPORTALREADYOPEN,EINVBAUDRATE,EINVPORT,EMEMALLOC};
#define COM1       0
#define COM2       1
#define ON         1
#define OFF        0
#define HIBYTE(n) ((n >> 8) & 0xFF)
#define LOBYTE(n) (n & 0xFF)

#if !defined(dim)
#define dim(x) (sizeof(x)/sizeof(x[0]))
#endif
int SetPortParms(int PortNo, int Baud, int Parity, int Data, int Stop,
                     int Base, int IRQ, int BreakLength);
int copen(int PortNo, int BufSiz, int Baud, int Parity, int Data, int Stop,
                     int Base, int IRQ, int BreakLength);
int cclose(int PortNo);
int InitBaud(int PortNo);
SetIntVector(int);
ClrIntVector(int);
SetUARTIe(int);
SetUartOUT2(int);
DrainOutQueue(int);
ControlDTR(int,int);
SetPIC(int PortNo, int State);
cQueueLen(int PortNo);
cgetc(int PortNo);
ctgetc(int POrtNo);
cputc(int PortNo, char Ch);

#endif
```

ROUTINE - CIRCLEQ.C 29/11/94

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "circleq.h"

/*
 * Qalloc allocates space for a Queue Control Structure (CircleQ) and
 * a buffer to be contained within.
 * If space is not available for this structure, or it's buffer, NULL
 * is returned, otherwise a pointer to a CirclQ is returned.
*/
CircleQ * Qalloc(int Size)
{
          CircleQ * Qptr;
          if ((Qptr = calloc(sizeof(CircleQ),1)) == NULL)
          {
                    errno = ENOMEM;
                    return NULL;
          }
          if ((Qptr->Buffer = calloc(Size,1)) == NULL)
          {
                    free(Qptr);
                    errno = ENOMEM;
                    return NULL ;
          }
          Qptr->Head = Qptr->Tail = Qptr->Buffer;
          Qptr->Size = Size;
          Qptr->BytesWaiting = 0;
          return Qptr;
}
/*
 * QinsertChar inserts a character on a queue. The tail pointer is used
 * to indicate where the character shall be placed. Once a character is
 * written, the tail pointer is updated. If the tail pointer overruns the
 * head pointer, the overrun flag is set. Once the overrun flag is set, the
 * head pointer will be updated on all subsequent writes to ensure that
 * the buffer always contains Qptr->Size characters. If Qptr->Tail or
 * Qptr->Head reach the end of the buffer, they are wrapped around to
 * the beginning of the buffer.
*/
void QinsertChar(CircleQ * Qptr, char Ch)
{
          *Qptr->Tail = Ch;
          if ((Qptr->Tail + 1) >= (Qptr->Buffer+Qptr->Size))
                    Qptr->Tail = Qptr->Buffer;
          else
                    Qptr->Tail ++;

          if (Qptr->Head == Qptr->Tail)
                    Qptr->Flag = 1;           /* Queue buffer overflow */
            else Qptr->BytesWaiting ++;
          if (Qptr->Flag)
                    Qptr->Head ++ ;
          if (Qptr->Head >= Qptr->Buffer + Qptr->Size)
                    Qptr->Head = Qptr->Buffer;
}
/*
 * QLen. returns the number of bytes presently in specified input buffer.
 *
 * As CircleQ means that the tail could be before the head, in memory,
 * the extra +size %size should correct the value.
 */
int QLen(CircleQ * Qptr)
{
```

```
          if (Qptr->Flag) return(1025);
          return (Qptr->BytesWaiting);
}
/*
 * QpeekChar returns a character from the specified queue without removing
 * it. If the queue is empty -1 is returned.
*/
int QpeekChar(CircleQ * Qptr)
{
          byte Ch;
          if (Qptr->Head == Qptr->Tail)
          {
                    Qptr->Flag = 0;
                    return -1;
          }
          Ch = *Qptr->Head;
          return Ch;
}
/*
 * QreadChar returns a character from the specified queue. The queue is then
 * updated to reflect the last read. If the queue is empty, -1 is returned.
*/
int QreadChar(CircleQ * Qptr)
{
          byte Ch;
          if (Qptr->Head == Qptr->Tail)
          {
                    Qptr->Flag = 0;
                    return -1;
          }
          Ch = *Qptr->Head;
          Qptr->Head ++ ;
          Qptr->BytesWaiting --;
          if (Qptr->Head >= Qptr->Buffer + Qptr->Size)
                    Qptr->Head = Qptr->Buffer;
          return Ch;
}
/*
 * Qfree frees a queue control structure (CircleQ) and all buffers
 * associated with it (Qptr->Buffer). This Head and Tail pointers are
 * set to NULL, and Size and Flag (overrun flag) are set to 0.
 * Qptr is then freed, releasing all memory associated with the queue.
 * No subsequent operations may be made on this queue.
*/
void Qfree(CircleQ * Qptr)
{
          if (Qptr != NULL)
          {
                    free(Qptr->Buffer);
                    Qptr->Buffer = Qptr->Head = Qptr->Tail = NULL;
                    Qptr->Size = Qptr->Flag = Qptr->BytesWaiting = 0;
                    free(Qptr);
          }
}
```

ROUTINE - CIRCLEQ.H 03/11/94

```
#if !defined(CIRCLEQ_INCLUDE)
#define CIRCLEQ_INCLUDE
typedef struct _circq
{
         char *    Head;
         char *    Tail;
         char *    Buffer;
         int       Size;
         int       BytesWaiting;
         int       Flag;
} CircleQ;

CircleQ *          Qalloc(int size);
void               Qfree(CircleQ * qptr);
void               QinsertChar(CircleQ * qptr, char ch);
int                QLen(CircleQ * Qptr);
int                QpeekChar(CircleQ * qptr);
int                QreadChar(CircleQ * qptr);
#endif
```

Objects used in Testing

1). **Aluminium Block** - 101 x 37.5 x 12.5 mm smooth surface;

2). **Glass** - approx. 60 mm diameter by 5.5 mm thick smooth surface, slightly curved; Bottom of a 'Spitfire' pint real-ale bottle.

3). **Eraser** - 57 x 24 x 12.5 mm smooth surface; 'COSTS' plastic eraser.

4). **Plastic** - 120 x 64 x 2 mm smooth surface; casing of a logic analyser probe pod.

5). **Wood** - 102 x 27.5 x 9.5 mm planed surface; pine.

6). **Card** - 145 x 100 x 2 mm lined smooth surface, on 10mm paper; cover of pocket notebook.

7). **Expanded polystyrene** - 120 x 100 x 8 mm smooth surface; packing block.

8). **White Foam** - 102 x 27.5 x 9.5 mm slightly rough surface; packing block.

9). **Black Foam** - 100 x 70 x 25 mm smooth surface

10). **Tufnol** - 125 x 40 x 13 mm smooth surfaced block;

11). **Diecast** - 170 x 40 x 2.5 mm smooth surface; section of a diecast box lid

12). **Ice/cold** - 100x75x20 mm rough surface; Ice brick, plastic covered.

13). **Hot** - 70 mm diameter by 80 mm curved smooth surface; Side of a hot cup of Tea, Coffee. (whatever I was drinking at the time!.)

Test Results

Title:
BGI Graphics
Creator:
BGI by Borland International
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
BGI Graphics
Creator:
BGI by Borland International
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
BGI Graphics
Creator:
BGI by Borland International
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
BGI Graphics
Creator:
BGI by Borland International
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
BGI Graphics
Creator:
BGI by Borland International
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
BGI Graphics
Creator:
BGI by Borland International
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
BGI Graphics
Creator:
BGI by Borland International
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
BGI Graphics
Creator:
BGI by Borland International
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Operator's Thermal generater - 60g total
20g weight on finger (ie connecter held)
Operator's Texout generater - 5g

Manipulator Thermal sensor - 50g total
25g weight on gripper (ie connecter held)
Manipulator Texin sensor - 15g

4 Way Cable - 90g

8 Way Cable - 50g

RS232 unit + cables - 255g

Powerc - 105g

Manipulator electronics unit - 95g

LCD - 130g

Operator's electronics unit - 180g

All weights measured on a 'total' postweigh 30 Wp postal scales
Resolution and accurace of 5g - 28/3/95

subject 1	subject 2	subject 3
JH	SL	MH
jump +5C		
1292	1241	944
1489	1442	844
844	1391	1044
745	1243	1193
944	1243	1243
avg = 1062.8	avg = 1312	avg = 1053.6
jump -2C		
794	547	645
745	645	644
845	547	598
596	547	596
447	598	746
avg = 685.4	avg = 576.8	avg = 645.8
jump -5C		
396	547	497
497	447	496
397	496	645
347	497	546
347	747	547
avg = 396.8	avg = 546.8	avg = 546.2
jump -10C		
349	447	498
349	545	447
497	498	597
398	646	598
396	596	448
avg = 397.8	avg = 546.4	avg = 517.6
	response.wq1	
	04/13/95	