599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallaxinc.com
Technical: stamptech@parallaxinc.com
Web Site: www.parallaxinc.com
Educational: www.stampsinclass.com

BS2p "Plus Pack" AppKit (#45184)

Introduction

The BS2p "Plus Pack" is a selection of components and ready-to-run source code to assist experimenters with mastering some of the exciting new features of the BS2p; specifically the use of parallel LCDs, Philips I²C™ components and Dallas Semiconductor 1-Wire® components.

Please note that this AppKit is designed for intermediate to advanced users. The schematics and source code have been carefully checked and are commented, but the expectation is that the user will consult the appropriate product data sheets (not duplicated here) for detailed explanation of each component's operation.

Each of the enclosed experiments was built, tested and run on the BS2p Demo Board (#45183). Should you desire more space for connecting components, please consider the NX-1000 lab board (#28135).

Packing List

Verify that your BS2p "Plus Pack" package is complete in accordance with the list below. The contents of the package include:

- Packing List (this page)
- Documentation/Source Code Diskette
- Parallel LCD module; 2 lines x 16 characters (HD44780-compatible)
- PCF8574 Remote 8-Bit I/O Expander
- PCF8583 Clock/Calendar with 240 x 8-Bit RAM
- PCF8591 8-Bit A/D and D/A Converter
- 24LC32 32K Serial EEPROM
- (2) DS1822 Econo-MicoLAN Digital Thermometer
- DS2405 Addressable Switch
- DS2890 1-Wire Digital Potentiometer
- (4) Jumper wires packs
- 220 ohm resistor
- (2) 1K resistor
- 10K resistor
- 100K potentiometer
- 0.01 uF capacitor
- (2) low-current LED
- Normally-open pushbutton switch
- 32.678 kHz crystal

PP_LCDDEMO1.BSP

- Connect LCD to the BS2p Demo Board X5
- Install jumper X6
- Adjust contrast pot for best display

```
' -----[ Title ]-----------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_LCDDEMO1.BSP
' Purpose... Basic LCD Demo - Single Line Mode
' Author.... Parallax, Inc.
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------
'
' This program demonstrates LCD basics using the BS2p.
'
' To run this program on the BS2p Demo Board, connect the LCD and install
' Jumper X6. Adjust contrast pot for best display.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.


' -----[ Revision History ]-------------------------------------------------
'


' -----[ I/O Definitions ]--------------------------------------------------
'
LCDpin  CON     0                               ' connect LCD to OutL


' -----[ Constants ]-------------------------------------------------------
'
NoCmd           CON     $00                     ' No command in LCDOUT
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control

DispCtrl        CON     %00001000               ' display control command

On              CON     1
Off             CON     0



' -----[ Variables ]-------------------------------------------------------
'
cmd             VAR     Byte                    ' command sent to LCD
```

```
display         VAR     cmd.Bit2                ' display on/off bit
cursor          VAR     cmd.Bit1                ' cursor on/off bit
blinking        VAR     cmd.Bit0                ' blinking on/off bit

char            VAR     cmd                     ' character sent to LCD
idx             VAR     Byte                    ' loop counter


' -----[ EEPROM Data ]----------------------------------------------------------
'


' -----[ Initialization ]-------------------------------------------------------
'
Initialize:
  PAUSE 500                                     ' let the LCD settle
  LCDCMD LCDpin,%00110000 : PAUSE 5             ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0             ' 4-bit mode
  LCDCMD LCDpin,%00001100 : PAUSE 0             ' no crsr, no blink
  LCDCMD LCDpin,%00000110                       ' inc crsr, no disp shift


' -----[ Main Code ]------------------------------------------------------------
'
Main:
  LCDCMD LCDpin,ClrLCD                          ' clear display
  PAUSE 500

Splash_Screen
  LCDOUT LCDpin,NoCmd,["THE BASIC STAMP!"]
  PAUSE 2000

Cursor_On:
  LCDCMD LCDpin,CrsrHm                          ' move the cursor home
  cmd = DispCtrl
  display = On
  cursor = On
  LCDCMD LCDpin,cmd
  PAUSE 500

Move_Cursor:
  FOR idx = 1 TO 15                             ' move the cursor across display
    LCDCMD LCDpin,CrsrRt
    PAUSE 150
  NEXT

  FOR idx = 14 TO 0                             ' go backward by moving cursor
    cmd = DDRam + idx                           '  to a specific address
    LCDCMD LCDpin,cmd
    PAUSE 150
  NEXT

  PAUSE 1000

Block_Cursor:
  cmd = DispCtrl
  display = On
  blinking = On                                 ' enable block cursor
  LCDCMD LCDpin,cmd
  PAUSE 2000
  blinking = Off                                ' turn it off
```

```
  LCDCMD LCDpin,cmd

Flash_Display:
  cmd = DispCtrl
  display = On

  FOR idx = 1 TO 10                             ' flash display by
    display = ~display                          '  toggling display bit
    LCDCMD LCDpin,cmd
    PAUSE 250
  NEXT

  PAUSE 1000

Shift_Display:
  FOR idx = 1 TO 16                             ' shift display to right
    LCDCMD LCDpin,DispRt
    PAUSE 100
  NEXT

  PAUSE 1000

  FOR idx = 1 TO 16                             ' shift display back
    LCDCMD LCDpin,DispLf
    PAUSE 100
  NEXT

  PAUSE 1000
  GOTO Main                                     ' do it all over
  END


' -----[ Subroutines ]------------------------------------------------------
'
```

PP_LCDDEMO2.BSP

- Connect LCD to the BS2p Demo Board X5
- Install jumper X6
- Adjust contrast pot for best display

```
' -----[ Title ]-----------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_LCDDEMO2.BSP
' Purpose... Basic LCD Demo - Multi-line mode with custom characters
' Author.... Parallax, Inc.
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------
'
' This program demonstrates the use of the multi-line initialization and
' the use of custom characters.  When using the standard 5x7 font, the LCD
' will hold up to eight customer characters.
'
' To run this program on the BS2p Demo Board, connect the LCD and install
' Jumper X6. Adjust contrast pot for best display.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.


' -----[ Revision History ]-------------------------------------------------
'


' -----[ I/O Definitions ]--------------------------------------------------
'
LCDpin          CON     0                       ' connect LCD to OutL


' -----[ Constants ]-------------------------------------------------------
'
NoCmd           CON     $00                     ' No command in LCDOUT
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control
CGRam           CON     $40                     ' Custom character RAM
Line1           CON     $80                     ' DDRAM address of line 1
Line2           CON     $C0                     ' DDRAM address of line 2


' -----[ Variables ]-------------------------------------------------------
'
```

```
cmd             VAR     Byte                    ' commnand sent to LCD
char            VAR     Byte                    ' character sent to LCD
newChr          VAR     Byte                    ' new character for animation
addr            VAR     Byte                    ' address in EE and display
cNum            VAR     Byte                    ' character number


' -----[ EEPROM Data ]----------------------------------------------------------
'
' custom character definitions

Mouth0          DATA    $0E,$1F,$1F,$1F,$1F,$1F,$0E,$00
Mouth1          DATA    $0E,$1F,$1F,$18,$1F,$1F,$0E,$00
Mouth2          DATA    $0E,$1F,$1C,$18,$1C,$1F,$0E,$00
Smile           DATA    $00,$0A,$0A,$00,$11,$0E,$06,$00

Msg             DATA    " IS VERY COOL! ",3     ' revealed message


' -----[ Initialization ]------------------------------------------------------
'
Initialize:
  PAUSE 500                                     ' let the LCD settle
  LCDCMD LCDpin,%00110000 : PAUSE 5             ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0             ' 4-bit mode
  LCDCMD LCDpin,%00101000 : PAUSE 0             ' 2-line mode
  LCDCMD LCDpin,%00001100 : PAUSE 0             ' no crsr, no blink
  LCDCMD LCDpin,%00000110                       ' inc crsr, no disp shift

DLChars:                                        ' download custom chars to LCD
  LCDCMD LCDpin,CGRam                           ' prepare to write CG data
  FOR addr = Mouth0 TO (Smile + 7)              ' build 4 custom chars
    READ addr,char                              ' get byte from EEPROM
    LCDOUT LCDpin,NoCmd,[char]                  ' put into LCD CGRAM
  NEXT


' -----[ Main Code ]------------------------------------------------------------
'
Main:
  LCDCMD LCDpin,ClrLCD
  PAUSE 1000
  LCDOUT LCDpin,NoCmd,["THE BASIC STAMP"]
  PAUSE 2000

  ' Animation by character replacement

  FOR addr = 0 TO 15                            ' cover 16 characters
    READ (Msg + addr),newChr                    ' get new char from message
    cmd = Line2 + addr                          ' set new DDRAM address
    FOR cNum = 0 TO 4                           ' 5 characters in cycle
      LOOKUP cNum,[2,1,0,1,newChr],char
      LCDOUT LCDpin,cmd,[char]                  ' write animation character
      PAUSE 100                                 ' delay between animation chars
    NEXT
  NEXT

  PAUSE 3000
  GOTO Main                                     ' do it all over
  END
```

```
' -----[ Subroutines ]------------------------------------------------------
'
```

PP_LCDFONT.BSP

- Connect LCD to the BS2p Demo Board X5
- Install jumper X6
- Adjust contrast pot for best display

```
' -----[ Title ]-----------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_LCDCFONT.BSP
' Purpose... Advanced LCD Demo - custom numeric font(s)
' Author.... Parallax, Inc.
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------
'
' This program demonstrates character definition replacement in order to create
' a custom font for numbers.  This program creates three custom characters that
' are used to display the tens, ones and tenths value of a counter.
'
' The program analyzes the counter and updates the screen by downloading the
' appropriate character map for each digit.
'
' To run this program on the BS2p Demo Board, connect the LCD and install
' Jumper X6. Adjust contrast pot for best display.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.


' -----[ Revision History ]-------------------------------------------------
'


' -----[ I/O Definitions ]--------------------------------------------------
'
LCDpin          CON     0                       ' connect LCD to OutL


' -----[ Constants ]-------------------------------------------------------
'
NoCmd           CON     $00                     ' No command in LCDOUT
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control
CGRam           CON     $40                     ' Custom character RAM
Line1           CON     $80                     ' DDRAM address of line 1
Line2           CON     $C0                     ' DDRAM address of line 2

CLines          CON     8                       ' lines per character
Space           CON     10
```

```
' -----[ Variables ]------------------------------------------------------------
'
char            VAR     Byte                    ' character sent to LCD
addr            VAR     Byte                    ' EE starting address of map
cNum            VAR     Nib                     ' character number
idx             VAR     Nib                     ' loop counter

counter         VAR     Word


' -----[ EEPROM Data ]-----------------------------------------------------------
'
' character definitions - digits 0 - 9 and space
'
Dig_0           DATA    $1F,$11,$11,$19,$19,$19,$1F,$00
Dig_1           DATA    $04,$04,$04,$0C,$0C,$0C,$0C,$00
Dig_2           DATA    $1F,$01,$01,$1F,$18,$18,$1F,$00
Dig_3           DATA    $1E,$02,$02,$1F,$03,$03,$1F,$00
Dig_4           DATA    $18,$18,$18,$19,$1F,$01,$01,$00
Dig_5           DATA    $1F,$18,$18,$1F,$01,$01,$1F,$00
Dig_6           DATA    $18,$10,$10,$1F,$19,$19,$1F,$00
Dig_7           DATA    $1F,$11,$01,$03,$03,$03,$03,$00
Dig_8           DATA    $0E,$0A,$0A,$1F,$13,$13,$1F,$00
Dig_9           DATA    $1F,$11,$11,$1F,$03,$03,$03,$00
Dig_Spc         DATA    $00,$00,$00,$00,$00,$00,$00,$00


' -----[ Initialization ]--------------------------------------------------------
'
Initialize:
  PAUSE 500                                     ' let the LCD settle
  LCDCMD LCDpin,%00110000 : PAUSE 5             ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0             ' 4-bit mode
  LCDCMD LCDpin,%00101000 : PAUSE 0             ' 2-line mode
  LCDCMD LCDpin,%00001100 : PAUSE 0             ' no crsr, no blink
  LCDCMD LCDpin,%00000110                       ' inc crsr, no disp shift

  FOR cNum = 0 TO 2                             ' initialize cust chars
    LOOKUP cNum,[Dig_0,Dig_0,Dig_Spc],addr
    GOSUB Update_CC
  NEXT


' -----[ Main Code ]--------------------------------------------------------------
'
Main:
  LCDOUT LCDpin,ClrLCD,["CUSTOM DIGITS"]        ' setup display
  LCDOUT LCDpin,(Line2 + 12),[2,1,".",0]

Show_Counter:
  FOR counter = 0 TO 999                        ' count in tenths 0 - 99.9
    FOR cNum = 0 TO 2
      addr = counter DIG cNum                   ' get a digit
      IF (cNum < 2) OR (addr > 0) THEN DigitOK
      addr = Space                              ' leading space if < 10
    DigitOK:
      addr = addr * CLines                      ' calculate map for this digit
      GOSUB Update_CC                           ' download to LCD
    NEXT
```

```
    PAUSE 100
  NEXT

  GOTO Main
  END


' -----[ Subroutines ]------------------------------------------------------
'
Update_CC:                                      ' update custom character
  LCDCMD LCDpin,(CGRam + (cNum * CLines))       ' point to character map
  FOR idx = 0 TO (CLines - 1)
    READ (addr + idx),char                      ' get data for character line
    LCDOUT LCDpin,NoCmd,[char]                  ' write to LCD CGRAM
  NEXT
  RETURN
```

PP_LCDODO.BSP

- Connect LCD to the BS2p Demo Board X5
- Install jumper X6
- Adjust contrast pot for best display

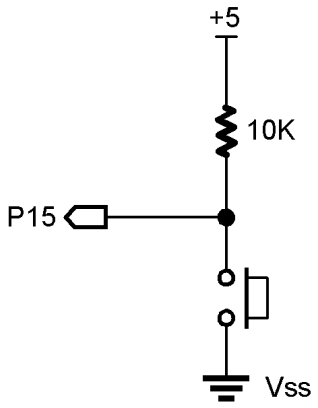
```
' -----[ Title ]-----------------------------------------------------------
'
' BS2p Plus Pack
'
' File..... PP_LCDODO.BSP
' Purpose... Advanced LCD Demo - rewriting CGRAM on the fly
' Author.... Parallax, Inc.
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------
'
' This program demonstrates LCD character animation by writing to the
' character map (in CGRAM) for a character that is already displayed. The
' refresh cycle of the LCD will cause the character to change when its
' map is changed.  This technique (originally by Scott Edwards) allows
' the programmer to create advanced animations by storing character (cell)
' definitions in the Stamp's EEPROM.
'
' This program displays a rolling odometer type reading (last digit
' "rolls").  Character definitions are copied from the standard set
' (using "LCD Character Creator" software from Parallax).
'
' Each character definition is separated by 2 blank lines in order to create
' 10 lines per "rolling" character.  This makes the math for calculating
' the starting line of the roller very easy.
'
' To run this program on the BS2p Demo Board, connect the LCD and install
' Jumper X6. Adjust contrast pot for best display.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.


' -----[ Revision History ]-------------------------------------------------
'


' -----[ I/O Definitions ]--------------------------------------------------
'
LCDpin          CON     0                       ' connect LCD to OutL


' -----[ Constants ]-------------------------------------------------------
'
NoCmd           CON     $00                     ' No command in LCDOUT
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
```

```
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control
CGRam           CON     $40                     ' Custom character RAM
Line1           CON     $80                     ' DDRAM address of line 1
Line2           CON     $C0                     ' DDRAM address of line 2

CLines          CON     8                       ' lines per character
OdoChar         CON     0                       ' animated odometer character


' -----[ Variables ]------------------------------------------------------
'
cmd             VAR     Byte                    ' commnand sent to LCD
char            VAR     Byte                    ' character sent to LCD
addr            VAR     Byte                    ' EE starting address of map
cNum            VAR     Nib                     ' character number
idx             VAR     Nib                     ' loop counter

counter         VAR     Word
hundreds        VAR     Byte                    ' hundredths value of counter

temp            VAR     Word                    ' temp value for RJ display
width           VAR     Nib                     ' width of rt justified
pos             VAR     Byte                    ' LCD display position
digits          VAR     Nib                     ' digits to display


' -----[ EEPROM Data ]----------------------------------------------------
'
' rolling odometer character definitions
'
Char0           DATA    $0E,$11,$13,$15,$19,$11,$0E,$00,$00,$00
Char1           DATA    $04,$0C,$04,$04,$04,$04,$0E,$00,$00,$00
Char2           DATA    $0E,$11,$01,$02,$04,$08,$1F,$00,$00,$00
Char3           DATA    $1F,$02,$04,$02,$01,$11,$0E,$00,$00,$00
Char4           DATA    $02,$06,$0A,$12,$1F,$02,$02,$00,$00,$00
Char5           DATA    $1F,$10,$1E,$01,$01,$11,$0E,$00,$00,$00
Char6           DATA    $06,$08,$10,$1E,$11,$11,$0E,$00,$00,$00
Char7           DATA    $1F,$01,$02,$04,$08,$08,$08,$00,$00,$00
Char8           DATA    $0E,$11,$11,$0E,$11,$11,$0E,$00,$00,$00
Char9           DATA    $0E,$11,$11,$0F,$01,$02,$0C,$00,$00,$00


' inverted character definitions (white on black)
'
Char0i          DATA    $11,$0E,$0C,$0A,$06,$0E,$11,$1F,$1F,$1F
Char1i          DATA    $1B,$13,$1B,$1B,$1B,$1B,$11,$1F,$1F,$1F
Char2i          DATA    $11,$0E,$1E,$1D,$1B,$17,$00,$1F,$1F,$1F
Char3i          DATA    $00,$1D,$1B,$1D,$1E,$0E,$11,$1F,$1F,$1F
Char4i          DATA    $1D,$19,$15,$0D,$00,$1D,$1D,$1F,$1F,$1F
Char5i          DATA    $00,$0F,$01,$1E,$1E,$0E,$11,$1F,$1F,$1F
Char6i          DATA    $19,$17,$0F,$01,$0E,$0E,$11,$1F,$1F,$1F
Char7i          DATA    $00,$1E,$1D,$1B,$17,$17,$17,$1F,$1F,$1F
Char8i          DATA    $11,$0E,$0E,$11,$0E,$0E,$11,$1F,$1F,$1F
Char9i          DATA    $11,$0E,$0E,$10,$1E,$1D,$13,$1F,$1F,$1F


MapStart        CON     Char0i


' -----[ Initialization ]------------------------------------------------
'
Initialize:
```

```
  PAUSE 500                                          ' let the LCD settle
  LCDCMD LCDpin,%00110000 : PAUSE 5                  ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0                  ' 4-bit mode
  LCDCMD LCDpin,%00101000 : PAUSE 0                  ' 2-line mode
  LCDCMD LCDpin,%00001100 : PAUSE 0                  ' no crsr, no blink
  LCDCMD LCDpin,%00000110                            ' inc crsr, no disp shift

  cNum = OdoChar
  addr = 0
  GOSUB Update_CC                                    ' put "0" into custom character


' -----[ Main Code ]----------------------------------------------------------
'
Main:
  LCDOUT LCDpin,ClrLCD,["ROLLER   COUNTER"]
  LCDOUT LCDpin,Line2, ["   0",OdoChar,"       0.00"]
  PAUSE 1000

Show_Counters:
  FOR counter = 0 TO 999
    FOR hundreds = 0 TO 99
      temp = counter                                 ' display odometer version
      width = 3
      pos = Line2 + 1
      GOSUB RJ_Print
      addr = hundreds
      GOSUB Update_CC                                ' update rolling character
      pos = Line2 + 10                               ' display digital version
      GOSUB RJ_Print
      LCDOUT LCDpin,NoCmd,["."DEC2 hundreds]
      PAUSE 100
    NEXT
  NEXT

  GOTO Main
  END


' -----[ Subroutines ]--------------------------------------------------------
'
Update_CC:                                           ' update custom character
  LCDCMD LCDpin,(CGRam + (cNum * CLines))            ' point to character map
  FOR idx = 0 TO (CLines - 1)
    READ MapStart + (addr + idx // 100),char
    LCDOUT LCDpin,NoCmd,[char]                       ' write to LCD CGRAM
  NEXT
  RETURN


RJ_Print:                                            ' right justified printing
  digits = width
  LOOKDOWN temp,<[0,10,100,1000,65535],digits
  LCDOUT LCDpin,pos,[REP " "\(width-digits),DEC temp]
  RETURN
```

PP_LCD5x10.BSP

- Connect LCD to the BS2p Demo Board X5
- Install jumper X6
- Adjust contrast pot for best display
- Assemble pushbutton circuit on breadboard



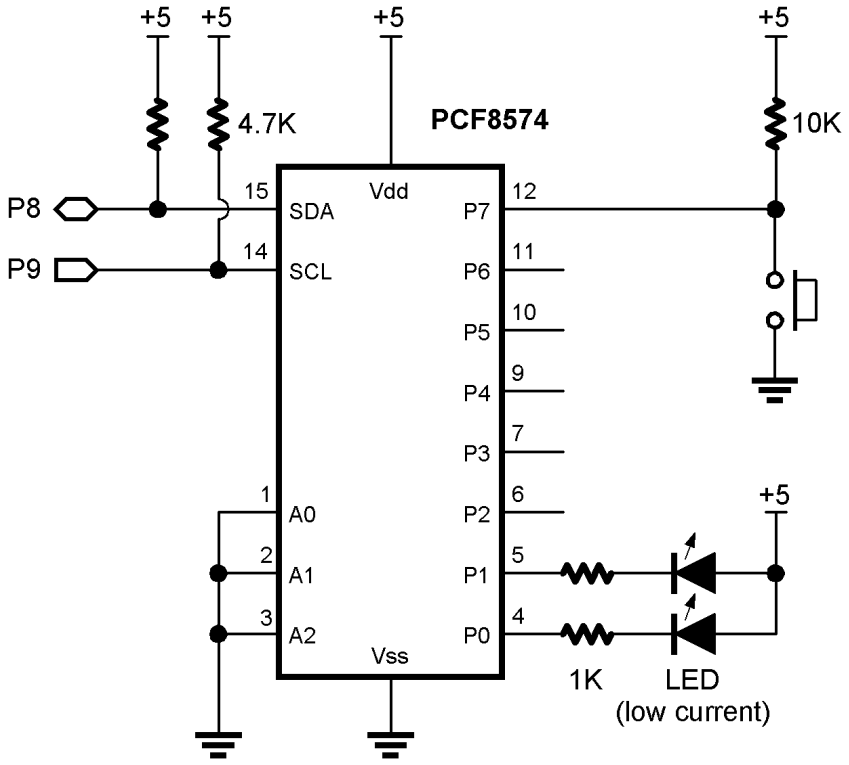

```
' -----[ Title ]-------------------------------------------------------------
'
' BS2p Plus Pack
'
' File..... PP_LCD5x10.BSP
' Purpose... Basic LCD Demo -- Using 5x10 font and descended characters
' Author.... Parallax, Inc.
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]-------------------------------------------------
'
' This program demostrates a method of intializing a 2x16 LCD so that it behaves
' like a single-line LCD that will display the 5x10 character set.  The LCD
' character map includes properly descended characters, but they are not mapped
' in the normal ASCII set.  A simple conversion routine can be used to replace
' "squishy" descended characters with proper ones.
'
' Stamp pin 15 is pulled up to Vdd (+5) through 10K.  This pin is connected to
' Vss (ground) through a N.O. pushbutton switch.  The pin will read 1 when the
' switch is open, 0 when pressed.
'
' To run this program on the BS2p Demo Board, assemble the the switch circuit on
' the breadboard, connect the LCD to X5 and install Jumper X6. Adjust contrast
' pot for best display.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.


' -----[ Revision History ]----------------------------------------------------
'
```

```
' -----[ I/O Definitions ]-------------------------------------------------
'
LCDpin          CON     0                       ' LCD is connected to OutL
AskBtn          VAR     In15                    ' Ask button input pin


' -----[ Constants ]-------------------------------------------------------
'
NoCmd           CON     $00                     ' No command in LCDOUT
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control

NumAns          CON     6                       ' 6 possible answers

_g              CON     $E7                     ' DDROM addresses of descenders
_j              CON     $EA
_p              CON     $F0
_q              CON     $F1
_y              CON     $F9

Pressed         CON     0                       ' button input is active low


' -----[ Variables ]------------------------------------------------------
'
char            VAR     Byte                    ' character sent to LCD
addr            VAR     Byte                    ' message address
answer          VAR     Nib                     ' answer pointer
clock           VAR     Nib                     ' animation clock
pntr            VAR     Nib                     ' pointer to animation character


' -----[ EEPROM Data ]-----------------------------------------------------
'
Prompt          DATA    "Ask a question",0      ' messages for LCD

Ans0            DATA    "Definitely YES",0
Ans1            DATA    "Possible...",0
Ans2            DATA    "Definitely NO",0
Ans3            DATA    "Not likely...",0
Ans4            DATA    "Answer uncertain",0
Ans5            DATA    "Please ask again",0


' -----[ Initialization ]-------------------------------------------------
'
Initialize:
  PAUSE 500                                     ' let the LCD settle
  LCDCMD LCDpin,%00110000 : PAUSE 5             ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0             ' 4-bit mode
  LCDCMD LCDpin,%00100100 : PAUSE 0             ' 5x10 font
  LCDCMD LCDpin,%00001100 : PAUSE 0             ' no crsr, no blink
  LCDCMD LCDpin,%00000110                       ' inc crsr, no disp shift
```

```
' -----[ Main Code ]-----------------------------------------------------------
'
Main:
  LCDCMD LCDpin,ClrLCD                          ' clear the LCD
  addr = Prompt
  GOSUB Show_Msg                                ' print prompt

Rollem:
  GOSUB Shuffle                                 ' shuffle until button pressed
  PAUSE 5
  IF (AskBtn = Pressed) THEN Show_Answer
  GOTO Rollem

Show_Answer:
  ' get address of answer message
  LOOKUP answer,[Ans0,Ans1,Ans2,Ans3,Ans4,Ans5],addr

  LCDCMD LCDpin,ClrLCD
  GOSUB Show_Msg

  PAUSE 2000                                    ' give time to read answer
  GOTO Main                                     ' do it all over
  END


' -----[ Subroutines ]---------------------------------------------------------
'
Show_Msg:
  READ addr,char                                ' read a character
  IF (char = 0) THEN Msg_Done                   ' if 0, message is complete
  GOSUB Translate                               ' fix letters with descenders
  LCDOUT LCDpin,NoCmd,[char]
  addr = addr + 1                               ' point to next character
  GOTO Show_Msg
Msg_Done:
  RETURN


' convert to descender font
' - does not change other characters

Translate:
  LOOKDOWN char,["g","j","q","p","y"],char      ' translate decended characters
  LOOKUP char,[_g,_j,_q,_p,_y],char
  RETURN


Shuffle:
  answer = answer + 1 // NumAns                 ' update answer pointer
  clock = clock + 1 // 15                       ' update pointer clock
  IF (clock > 0) THEN Shuffle_Done              ' time to update animation?
  LOOKUP pntr,["-+|*"],char                     ' load animation character
  LCDOUT LCDpin,DDRam + 15,[char]               ' write it at column 15
  pntr = pntr + 1 // 4                          ' update animation char
Shuffle_Done:
  RETURN
```

PP_PCF8574.BSP

- Assemble PCF8583574 circuit on breadboard
  -- use on-board 4.7K resistors (R1 and R2) for pull-ups



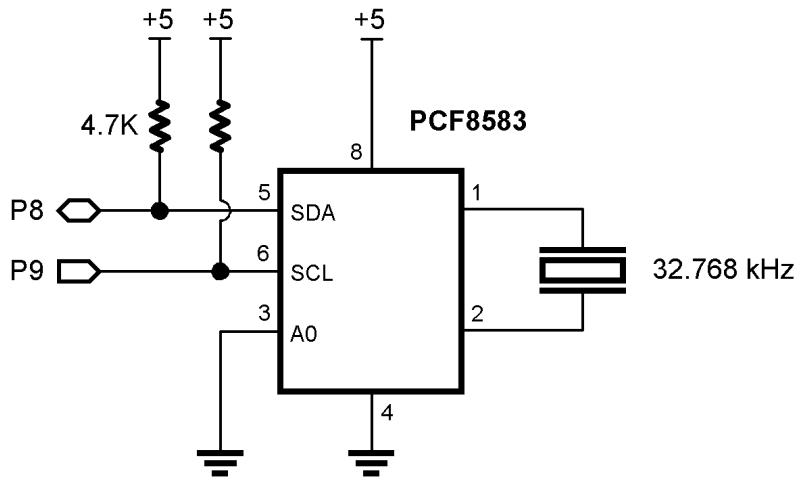

```
' -----[ Title ]-------------------------------------------------------------
'
' BS2p Plus Pack
'
' File..... PP_PCF8574.BSP
' Purpose... Reads remote input and updates 2 remote outputs on PCF8574
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]-------------------------------------------------
'
' This program reads bit 7 from the PCF8574.  If that bit is high (button is
' pressed), a counter is incremented and displayed via LEDs on PCF8574 bits
' 0 and 1.
'
' Note: Most (not all) I2C devices have multiple internal addresses, so the
' I2CIN and I2COUT commands support this with an address parameter (this byte
' comes after the Slave Address byte).  With the PCF8574, replace the address
' byte with a value that reflects the desired state of the I/O pins, where
```

```
' 1 is an input.  For example:
'
' %11100000 = Bits 0 - 4 are outputs, bits 5 - 7 are inputs
'
' For the PCF8574 the syntax becomes:
'
'    I2CIN  pin, ddr_value, [in_byte]
'    I2COUT pin, ddr_value, [out_byte]
'
' Special Note: When reading inputs while using the PCF8574 in mixed I/O mode,
' you must refresh the output bits during the read.  This is easily accomplished
' by ORing the state of the output pins with the DDR value.
'
'    I2CIN  pin, (ddr_value | out_bits), [out_byte]
'
' This program uses the bits in mixed mode and will use the syntax described
' immediately above.
'
' I/O Notes:
'
' The input bit is pulled up to Vdd (+5) through 10K.  This input is connected
' to Vss (ground) through a N.O. pushbutton switch.  The input will read 1 when
' the switch is open, 0 when pressed.
'
' PCF8574 can sink current, but provide almost no source current.  Outputs for
' this program are setup as active-low.  The tilde (~) in front of the variable
' cntr inverts the bits since we're using active low outputs.


' -----[ Revision History ]------------------------------------------------
'


' -----[ I/O Definitions ]-------------------------------------------------
'
I2Cpin          CON     8                       ' SDA on 8; SCL on 9


' -----[ Constants ]-------------------------------------------------------
'
DevType         CON     %0100 << 4              ' Device type
DevAddr         CON     %000 << 1               ' address = %000 -> %111
Wr8574          CON     DevType | DevAddr       ' write to PCF8574
Rd8574          CON     Wr8574 | 1              ' read from PCF8574

MixDDR          CON     %11111100               ' 1 = input, 0 = output


' -----[ Variables ]-------------------------------------------------------
'
ioByte          VAR     Byte                    ' i/o byte for PCF8574
btn             VAR     ioByte.Bit7             ' button input (0 = pressed)
cntr            VAR     Nib                     ' counter


' -----[ EEPROM Data ]-----------------------------------------------------
'


' -----[ Initialization ]--------------------------------------------------
'
Initialize:
  DEBUG CLS
```

```
  PAUSE 100
  DEBUG "PCF8574 Demo", CR
  DEBUG "Press button to update counter"


' -----[ Main Code ]----------------------------------------------------------
'
Main:
  I2CIN I2Cpin, Rd8574, (MixDDR | ~cntr), [ioByte]
  IF (btn) THEN Main                            ' wait for press
  cntr = cntr + 1 // 4                          ' update counter
  DEBUG Home, 10, 10, 10, BIN2 cntr             ' display on screen
  I2COUT I2Cpin, Wr8574, MixDDR, [~cntr]        ' send new value
  PAUSE 200
  GOTO Main


' -----[ Subroutines ]--------------------------------------------------------
'
```

PP_PCF8583.BSP

- Assemble PCF8583 circuit on breadboard
  -- use on-board 4.7K resistors (R1 and R2) for pull-ups




```
' -----[ Title ]-------------------------------------------------------------
'
' BS2p Plus Pack
'
' File..... PP_PCF8583.BSP
' Purpose... PCF8583 RTC Demo
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]-------------------------------------------------
'
' The program demonstrates the PCF8583 RTC/RAM.  When the program starts, you
' will be asked if you want to set the time.  If Yes, you'll enter the hours,
' minutes and day.  When running, the program displays the time and the day
' (by name) on a two-line LCD.

' To run this program on the BS2p Demo Board, connect the LCD and install
' Jumper X3.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.


' -----[ Revision History ]----------------------------------------------------
'




' -----[ I/O Definitions ]-----------------------------------------------------
'
```

```
LCDpin          CON     0                       ' LCD is connected to OutL
I2Cpin          CON     8                       ' SDA on 8; SCL on 9
RxD             CON     16                      ' serial receive (from DEBUG)


' -----[ Constants ]-------------------------------------------------------
'
DevType         CON     %1010 << 4              ' device type
DevAddr         CON     %000 << 1               ' address = %000 -> %001
Wr8583          CON     DevType | DevAddr       ' write to PCF8583
Rd8583          CON     Wr8583 | 1              ' read from PCF8583

' LCD control characters
'
NoCmd           CON     $00                     ' just print
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' cursor home
CrsrLf          CON     $10                     ' cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift display left
DispRt          CON     $1C                     ' shift displayright
DDRam           CON     $80                     ' Display Data RAM control
Line1           CON     $80                     ' address of line 1
Line2           CON     $C0                     ' address of line 2

Yes             CON     1
No              CON     0

Baud96          CON     240                     ' 9600-8-N-1 (matches DEBUG)


' -----[ Variables ]-------------------------------------------------------
'
seconds         VAR     Byte
minutes         VAR     Byte
hours           VAR     Byte
day             VAR     Nib                     ' 0 - 6 (day of week)
date            VAR     Byte                    ' 1 - 31
month           VAR     Nib
year            VAR     Nib                     ' 0 - 3 (LeapYear offset)

rawTime         VAR     Word                    ' minutes past midnight

regCtrl         VAR     Byte                    ' [0] control/status
regHuns         VAR     Byte                    ' [1] hundredths (bcd)
regSecs         VAR     Byte                    ' [2] seconds (bcd)
regMins         VAR     Byte                    ' [3] minutes (bcd)
regHrs          VAR     Byte                    ' [4] hours (bcd)
regYrDate       VAR     Byte                    ' [5] year & date (bcd+)
regMoDay        VAR     Byte                    ' [6] day & month (bcd+)

regAddr         VAR     Byte                    ' register address
regData         VAR     Byte                    ' data to/from register

eeAddr          VAR     Byte                    ' EE data pointer
char            VAR     Byte                    ' character from EE
idx             VAR     Byte                    ' loop counter

response        VAR     Byte


' -----[ EEPROM Data ]-----------------------------------------------------
'
```

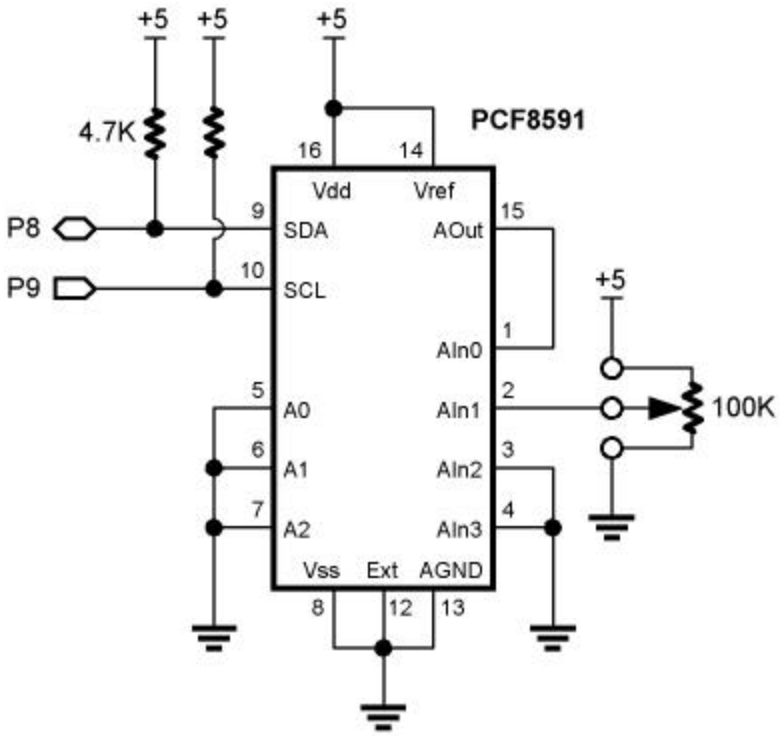
```
Su              DATA    "   SUNDAY",0
Mo              DATA    "   MONDAY",0
Tu              DATA    "  TUESDAY",0
We              DATA    "WEDNESDAY",0
Th              DATA    " THURSDAY",0
Fr              DATA    "   FRIDAY",0
Sa              DATA    " SATURDAY",0


' -----[ Initialization ]----------------------------------------------------
'
Initialize:
  DEBUG CLS                                     ' open DEBUG window
  PAUSE 500                                     ' let LCD settle


LCD_Setup:
  LCDCMD LCDpin,%00110000 : PAUSE 5             ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0             ' 4-bit mode
  LCDCMD LCDpin,%00101000 : PAUSE 0             ' 2-line mode
  LCDCMD LCDpin,%00001100 : PAUSE 0             ' no crsr, no blink
  LCDCMD LCDpin,%00000110                       ' inc crsr, no disp shift

  LCDOUT LCDpin,ClrLCD,["BSP <--> PCF8583"]     ' splash screen


Check_Set_Clock:
  DEBUG "Would you like to set the clock? (Y/N)"
  SERIN RxD,Baud96,10000,Main,[response]
  idx = 99
  LOOKDOWN response,["nNyY"],idx
  idx = idx / 2
  IF (idx = 0) THEN Main


Enter_Hours:
  DEBUG CR, "Hours (0..23): "
  SERIN RxD,Baud96,[DEC2 hours]
  IF (hours < 24) THEN Enter_Minutes
  hours = 6


Enter_Minutes:
  DEBUG CR, "Minutes (0..59): "
  SERIN RxD,Baud96,[DEC2 minutes]
  IF (hours < 60) THEN Enter_Day
  minutes = 0


Enter_Day:
  DEBUG CR, "Day (0..6 [0 = Sunday]): "
  SERIN RxD,Baud96,[DEC1 day]
  IF (day < 7) THEN Set_The_Clock
  day = 0


Set_The_Clock:
  month = 9
  date = 18
  year = 1
  GOSUB Put_Clock
```

```
' -----[ Main Code ]-----------------------------------------------------------
'
Main:
  DEBUG CLS, "The clock is running..."
  LCDCMD LCDpin, ClrLCD

Show_Clock:
  GOSUB Get_Time_And_Day
  LCDOUT LCDpin,Line1,[DEC2 hours,":",DEC2 minutes,":",DEC2 seconds]
  LCDCMD LCDpin, (Line2 + 7)
  GOSUB Print_Day
  GOTO Show_Clock


' -----[ Subroutines ]----------------------------------------------------------
'
Put_Register:
  I2COUT I2Cpin,Wr8583,regAddr,[regData]        ' send data to register
  RETURN


Get_Register:
  I2CIN I2Cpin,Rd8583,regAddr,[regData]         ' get data from register
  RETURN


Put_Raw_Clock:                                  ' set with rawTime
  minutes = rawTime // 60
  hours = rawTime / 60

Put_Clock:
  regSecs = 0
  regMins.HighNib = minutes / 10                ' convert regs to BCD
  regMins.LowNib = minutes // 10
  regHrs.HighNib = hours / 10
  regHrs.LowNib = hours // 10
  regMoDay.HighNib = month / 10
  regMoDay.LowNib = month // 10
  regMoDay = regMoDay | (day << 5)              ' pack weekday in
  I2COUT I2Cpin,Wr8583,2,[STR regSecs\5]        ' write time & day
  RETURN


Get_Time_And_Day:
  I2CIN I2Cpin,Rd8583,0,[STR regCtrl\7]

  ' convert from BCD

  seconds = (regSecs.HighNib * 10) + regSecs.LowNib
  minutes = (regMins.HighNib * 10) + regMins.LowNib
  hours = (regHrs.HighNib * 10) + regHrs.LowNib
  rawTime = (hours * 60) + minutes
  day = regMoDay >> 5
  RETURN


Print_Day:
  LOOKUP day,[Su,Mo,Tu,We,Th,Fr,Sa],eeAddr      ' point to EE string
Print_Loop:
  READ eeAddr,char                              ' read a character
  IF (char = 0) THEN Print_Done                 ' done?
```

```
  LCDOUT LCDpin,NoCmd,[char]                    ' print the character
  eeAddr = eeAddr + 1                           ' point to next
  GOTO Print_Loop:                              ' go get it
Print_Done:
  RETURN
```

PP_PCF8591.BSP

- Assemble PCF8591 circuit on breadboard
  -- use on-board 4.7K resistors (R1 and R2) for pull-ups



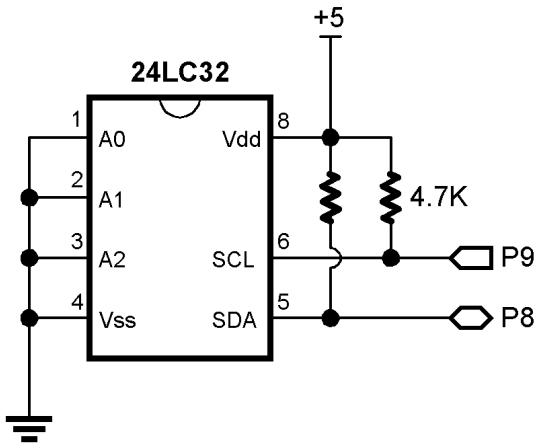

```
' -----[ Title ]-------------------------------------------------------------
'
' BS2p Plus Pack
'
' File..... PP_PCF8591.BSP
' Purpose... PCF8591 A2D/D2A Demo
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]-------------------------------------------------
'
' This program demonstates the Philips PCF8591 4-channel A2D plus 1-channel
' D2A.  Channel 0 input is tied to the output of the D2A pin.  Channel 1 input
' is tied to the wiper of a pot.  Channes 2 and 3 are tied to Vss.
'
' The PCF85591 uses a control byte after the Slave Address.  The control byte
' data (see details in PCF8591 documentation) is used to enable the analog
' output bit and set the kind of analog inputs.  In this demo, the analog output
' bit is enabled and four single-ended analog inputs are used.
'
' Note that the first byte transmitted in a read cycle contains the conversion
```

```
' result code of the previous read cycle, so a dummy byte is placed ahead of
' the analog input array in the I2CIN command.


' -----[ Revision History ]------------------------------------------------
'


' -----[ I/O Definitions ]--------------------------------------------------
'
I2Cpin          CON     8                       ' SDA on 8; SCL on 9


' -----[ Constants ]-------------------------------------------------------
'
DevType         CON     %1001 << 4              ' device type
DevAddr         CON     %000 << 1               ' address = %000 -> %111
Wr8591          CON     DevType | DevAddr       ' write to PCF8591
Rd8591          CON     Wr8591 | 1              ' read from PCF8591

D2A_Enable      CON     %01000000               ' enable analog output
Auto_Inc        CON     %00000100               ' auto inc a2d channels

MVPB            CON     $139C                   ' millivolts per bit factor


' -----[ Variables ]-------------------------------------------------------
'
aOut            VAR     Byte                    ' analog out value
aIn             VAR     Byte(4)                 ' analog input channels
mVolts          VAR     Word                    ' convert to millivolts
dummy           VAR     mVolts.LowByte
chan            VAR     Nib                     ' channel


' -----[ EEPROM Data ]-----------------------------------------------------
'


' -----[ Initialization ]--------------------------------------------------
'
Initialize:
  DEBUG CLS                                     ' call DEBUG window
  PAUSE 250                                     ' let it open


' -----[ Main Code ]-------------------------------------------------------
'
Main:
  DEBUG Home, "PCF8591 Demo"

Set_D2A:
  DEBUG Home, CR, CR, "D2A Out........ ", DEC aOut, "  ", CR
  I2COUT I2Cpin, Wr8591, D2A_Enable, [aOut]


Get_A2D:
  I2CIN I2Cpin, Rd8591, (D2A_Enable | Auto_Inc), [dummy, STR aIn\4]

  FOR chan = 0 TO 3
    DEBUG "Channel ", DEC1 chan, " In... ", DEC aIn(chan), "  ", Tab
    mVolts = aIn(chan) */ MVPB
    DEBUG "(", DEC mVolts DIG 3, ".", DEC3 mVolts, " volts)", CR
```

```
  NEXT

  PAUSE 500                                     ' delay between updates
  aOut = aOut + 1                               ' increment analog output
  GOTO Set_D2A                                  ' go again

  END


' -----[ Subroutines ]------------------------------------------------------
'
```

PP_24LC32.BSP

- Connect LCD to the BS2p Demo Board X5
- Install jumper X6
- Adjust contrast pot for best display
- Assemble 24LC32 ciruit on breadboard
  -- use on-board 4.7K resistors (R1 and R2) for pull-ups




```
' -----[ Title ]-----------------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_24LC32.BSP
' Purpose... Demonstrates I2CIN, I2COUT and using an LCD
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------------
'
' This program writes to and reads from a 24LC32 I2C EEPROM.  The status of the
' program and data are displayed on a 2x16 LCD.
'
' To run this program on the BS2p Demo Board, install the 24LC32 in the bread-
' board and wire connect to the BS2p with jumper wires.  Connect the LCD to X5
' and enable by installing Jumper X6.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.


' -----[ I/O Definitions ]--------------------------------------------------------
'
LCDpin          CON     0                       ' LCD is connected to OutL
I2Cpin          CON     8                       ' SDA on 8; SCL on 9
```

```
' -----[ Constants ]-------------------------------------------------------
'
DevType         CON     %1010 << 4              ' device type
DevAddr         CON     %000 << 1               ' address = %000 -> %111
Wr2432          CON     DevType | DevAddr       ' write to 24LC32
Rd2432          CON     Wr2432 | 1              ' read from 24LC32

MaxEE           CON     4095                    ' highest EE address


' LCD control characters
'
NoCmd           CON     $00                     ' No command in LCDOUT
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control
CGRam           CON     $40                     ' Custom character RAM
Line1           CON     $80                     ' DDRAM address of line 1
Line2           CON     $C0                     ' DDRAM address of line 2


' -----[ Variables ]-------------------------------------------------------
'
addr            VAR     Word                    ' EE address
addrHi          VAR     addr.HighByte
addrLo          VAR     addr.LowByte
rVar            VAR     Word                    ' for random number
tOut            VAR     Byte                    ' test value to LCD
tIn             VAR     Byte                    ' test value read from LCD
temp            VAR     Word                    ' temp value for display
width           VAR     Nib                     ' width of rt justified
pos             VAR     Byte                    ' column position
digits          VAR     Nib                     ' digits to display


' -----[ EEPROM Data ]-----------------------------------------------------
'
Super2          DATA    %01100                  ' superscript 2 (custom char)
                DATA    %00010
                DATA    %00100
                DATA    %01000
                DATA    %01110
                DATA    %00000
                DATA    %00000
                DATA    %00000


' -----[ Initialization ]--------------------------------------------------
'
LCD_Setup:
  PAUSE 500                                     ' let the LCD settle
  LCDCMD LCDpin,%00110000 : PAUSE 5             ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0             ' 4-bit mode
  LCDCMD LCDpin,%00101000 : PAUSE 0             ' 2-line mode
  LCDCMD LCDpin,%00001100 : PAUSE 0             ' no crsr, no blink
  LCDCMD LCDpin,%00000110                       ' inc crsr, no display shift
```

```
  ' download custom character map to LCD

  LCDCMD LCDpin,(CGRam + (2 * 8))               ' write to CGRAM (character 2)
  FOR addr = Super2 TO (Super2 + 7)             ' build custom char
    READ addr,temp                              ' get byte from EEPROM
    LCDOUT LCDpin,NoCmd,[temp]                  ' put into LCD CG RAM
  NEXT


' -----[ Main Code ]----------------------------------------------------------
'
Splash:
  LCDOUT LCDpin,ClrLCD,["  BS2P <-> I",2,"C"]
  LCDOUT LCDpin,Line2, [" Communications"]
  PAUSE 2000

Main:
  LCDOUT LCDpin,ClrLCD,["I",2,"C:     Out="]
  LCDOUT LCDpin,(Line2 + 10),["In="]

  FOR addr = 0 TO MaxEE STEP 5                  ' create addresses
    RANDOM rVar                                 ' create "random" value
    tOut = rVar.HighByte

    ' write value then read it back

    I2COUT I2Cpin,Wr2432,addrHi\addrLo,[tOut]
    PAUSE 100
    I2CIN I2Cpin,Rd2432,addrHi\addrLo,[tIn]

    ' display results

    LCDOUT LCDpin,(Line1 + 4),[DEC addr]
    temp = tOut : width = 3 : pos = Line1 + 13
    GOSUB RJ_Print
    temp = tIn : width = 3 : pos = Line2 + 13
    GOSUB RJ_Print
    PAUSE 250
  NEXT

  END


' -----[ Subroutines ]---------------------------------------------------------
'
RJ_Print:                                       ' right justified printing
  digits = width
  LOOKDOWN temp,<[0,10,100,1000,65535],digits
  LCDOUT LCDpin,pos,[REP " "\(width-digits),DEC temp]
  RETURN
```

PP_OWID.BSP

```
' -----[ Title ]-----------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_OWID.BSP
' Purpose... Reads ROM data from 1-Wire device
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p, PP_OWNAMES.BSP}


' -----[ Program Description ]---------------------------------------------
'
' Reads 1-Wire device ROM pattern.  Data is displayed in DEBUG window
'
' If using the iButton socket on the BS2p demo board, install Jumper iB1.
'
' If using a "Blue Dot Receptor" (RJ-11 connection) with the BS2p Demo
' Board, install Jumper iB2
'
' Do not connect more than one device.


' -----[ I/O Definitions ]--------------------------------------------------
'
OWpin           CON     15                      ' 1-wire device pin


' -----[ Constants ]-------------------------------------------------------
'
' 1-Wire Support
'
OW_FERst        CON     %0001                   ' Front-End Reset
OW_BERst        CON     %0010                   ' Back-End Reset
OW_BitMode      CON     %0100
OW_HighSpd      CON     %1000

ReadROM         CON     $33                     ' read ID, serial num, CRC
SearchROM       CON     $F0                     ' search

NoDevice        CON     %11                     ' no device present
NamesPgm        CON     1                       ' names are stored in slot 1


' -----[ Variables ]-------------------------------------------------------
'
idx             VAR     Byte                    ' loop counter
romData         VAR     Byte(8)                 ' ROM data from device
devType         VAR     romData
devCheck        VAR     Nib                     ' device check return ocde
addr            VAR     Word                    ' address of string pointer
strPtr          VAR     Word                    ' string pointer (device address)
char            VAR     Byte                    ' character for LCD
```

```
' -----[ Initialization ]----------------------------------------------------
'
Initialize:
  PAUSE 500                                     ' let DEBUG window open


' -----[ Main Code ]----------------------------------------------------------
'
Main:
  DEBUG CLS
  GOSUB Device_Check                            ' look for device
  IF (devCheck <> NoDevice) THEN Display_ROM

No_Device_Found:
  DEBUG "No 1-Wire device present."
  END

Display_ROM:
  OWOUT OWpin,OW_FERst,[ReadROM]                ' send Read ROM command
  OWIN  OWpin,OW_BERst,[STR romData\8]          ' read serial number & CRC

  IF (romData(7) < $FF) THEN Show_Device
  DEBUG CLS,"Bad device?"
  END

Show_Device
  DEBUG "Dallas 1-Wire ID : ", HEX2 romData(0), " ("
  GOSUB Display_Device_Type
  DEBUG ")",CR

  DEBUG "   Serial Number : "

  FOR idx = 6 TO 1
    DEBUG HEX2 romData(idx)
  NEXT

  DEBUG CR, "        Checksum : ", HEX2 romData(7),CR,CR

  DEBUG "      Stamp Data : ",IHEX2 romData(0)
  FOR idx = 1 TO 7
    DEBUG ",",IHEX2 romData(idx)
  NEXT

  END


' -----[ Subroutines ]--------------------------------------------------------
'

' This subroutine checks to see if any 1-Wire devices are present on the
' bus.  It does NOT search for ROM codes
'
Device_Check:
  devCheck = 0
  OWOUT OWpin,OW_FERst,[SearchROM]              ' reset and start search
  OWIN  OWpin,OW_BitMode,[devCheck.Bit1,devCheck.Bit0]
  RETURN


' This subroutine is used to display the part number of a 1-Wire device.
' The text data and pointers to it are stored in the EE of a different
' program slot.
```

```
Display_Device_Type:
  addr = devType * 2 + $600                    ' calulate string pointer addr
  STORE NamesPgm                               ' point to names EEPROM
  READ addr,strPtr.LowByte                     ' get the string location
  READ addr+1,strPtr.HighByte
Read_Char:
  READ strPtr,char                             ' read character from string
  IF (char = 0) THEN Dev_Type_Done             ' at end? (0 = Yes)
  DEBUG char                                   ' no, print the char
  strPtr = strPtr + 1                          ' point to next char
  GOTO Read_Char
Dev_Type_Done:
  STORE 0                                      ' point to main pgm slot
  RETURN
```

```
' -----[ Title ]-------------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_OWNAMES.BSP
' Purpose... Device names for Dallas 1-Wire components
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]-------------------------------------------
'
' There is no actual code in this module. This program stores the names
' of various Dallas Semiconductor 1-Wire devices.  The device family code
' is used to map the strings in EEPROM.  The pointer to a device's string
' discription is stored at the location determined by this formula:
'
'    pointer = device_id * 2 + $600
'
' "pointer" is actually the low-byte of the address location.  The high
' byte is at pointer+1.


' -----[ Constants ]-----------------------------------------------------
'
PntrBase        CON     $600


' -----[ EEPROM Data ]---------------------------------------------------
'
' Store strings first so labels can be used in address pointer table
'
Unknown         DATA    "Unknown device",0

' shared family codes

FCode01         DATA    "DS1990/DS2401",0
FCode04         DATA    "DS1994/DS2404",0
FCode10         DATA    "DS1920/DS18S20",0
FCode14         DATA    "DS1971/DS2430",0
FCode23         DATA    "DS1973/DS2433",0

' iButtons
```

```
DS1920          DATA    "DS1920",0
DS1921          DATA    "DS1921",0
DS1963          DATA    "DS1963",0
DS1971          DATA    "DS1971",0
DS1973          DATA    "DS1973",0
DS1982          DATA    "DS1982",0
DS1985          DATA    "DS1985",0
DS1986          DATA    "DS1986",0
DS1990          DATA    "DS1990",0
DS1991          DATA    "DS1991",0
DS1992          DATA    "DS1992",0
DS1993          DATA    "DS1993",0
DS1994          DATA    "DS1994",0
DS1995          DATA    "DS1995",0
DS1996          DATA    "DS1996",0

' 1-Wire chips

DS1822          DATA    "DS1822",0
DS18B20         DATA    "DS18B20",0
DS18S20         DATA    "DS18S20",0
DS2401          DATA    "DS2401",0
DS2404          DATA    "DS2404",0
DS2405          DATA    "DS2405",0
DS2406          DATA    "DS2406",0
DS2417          DATA    "DS2417",0
DS2430          DATA    "DS2430",0
DS2433          DATA    "DS2433",0
DS2450          DATA    "DS2450",0
DS2505          DATA    "DS2505",0
DS2506          DATA    "DS2506",0
DS2890          DATA    "DS2890",0


' string pointers

Pointers        DATA    @$01*2+PntrBase,Word FCode01
                DATA    @$04*2+PntrBase,Word FCode04
                DATA    @$10*2+PntrBase,Word FCode10
                DATA    @$14*2+PntrBase,Word FCode14
                DATA    @$23*2+PntrBase,Word FCode23

                DATA    @$02*2+PntrBase,Word DS1991
                DATA    @$06*2+PntrBase,Word DS1993
                DATA    @$08*2+PntrBase,Word DS1992
                DATA    @$09*2+PntrBase,Word DS1982
                DATA    @$0A*2+PntrBase,Word DS1995
                DATA    @$0C*2+PntrBase,Word DS1996
                DATA    @$1A*2+PntrBase,Word DS1963
                DATA    @$21*2+PntrBase,Word DS1921

                DATA    @$05*2+PntrBase,Word DS2405
                DATA    @$0B*2+PntrBase,Word DS2505
                DATA    @$0F*2+PntrBase,Word DS2506
                DATA    @$12*2+PntrBase,Word DS2406
                DATA    @$20*2+PntrBase,Word DS2450
                DATA    @$22*2+PntrBase,Word DS1822
                DATA    @$27*2+PntrBase,Word DS2417
                DATA    @$28*2+PntrBase,Word DS18B20
                DATA    @$2C*2+PntrBase,Word DS2890
```

PP_OWID-LCD.BSP

```
' -----[ Title ]-----------------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_OWID-LCD.BSP
' Purpose... Reads ROM data from 1-Wire device
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p, PP_OWNAMES.BSP}


' -----[ Program Description ]---------------------------------------------------
'
' Reads 1-Wire device ROM pattern.  Data is displayed on a 2x16 LCD.
'
' To run this program on the BS2p Demo Board, connect the LCD and install
' Jumper X6. Adjust contrast pot for best display.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.
'
' If using the iButton socket on the BS2p demo board, install Jumper X1-iB1.
'
' If using a "Blue Dot Receptor" (RJ-11 connection) with the BS2p Demo Board,
' install Jumper X1-iB2
'
' Do not connect more than one device.


' -----[ I/O Definitions ]--------------------------------------------------------
'
LCDpin          CON     0                       ' connect LCD to OutL
OWpin           CON     15                      ' 1-wire device pin


' -----[ Constants ]-------------------------------------------------------------
'
NoCmd           CON     $00                     ' No command in LCDOUT
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control
CGRam           CON     $40                     ' Custom character RAM
Line1           CON     $80                     ' DDRAM address of line 1
Line2           CON     $C0                     ' DDRAM address of line 2

' 1-Wire Support
'
OW_FERst        CON     %0001                   ' Front-End Reset
OW_BERst        CON     %0010                   ' Back-End Reset
OW_BitMode      CON     %0100
OW_HighSpd      CON     %1000
```

```
ReadROM         CON     $33                     ' read ID, serial num, CRC
SearchROM       CON     $F0                     ' search

NoDevice        CON     %11                     ' no device present
NamesPgm        CON     1                       ' names are stored in slot 1


' -----[ Variables ]------------------------------------------------------
'
idx             VAR     Byte                    ' loop counter
romData         VAR     Byte(8)                 ' ROM data from device
devType         VAR     romData
devCheck        VAR     Nib                     ' device check return ocde
addr            VAR     Word                    ' address of string pointer
strPtr          VAR     Word                    ' string pointer (device address)
char            VAR     Byte                    ' character for LCD


' -----[ Initialization ]--------------------------------------------------
'
Initialize:
  PAUSE 500                                     ' let the LCD settle
  LCDCMD LCDpin,%00110000 : PAUSE 5             ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0             ' 4-bit mode
  LCDCMD LCDpin,%00101000 : PAUSE 0             ' 2-line mode
  LCDCMD LCDpin,%00001100 : PAUSE 0             ' no crsr, no blink
  LCDCMD LCDpin,%00000110                       ' inc crsr, no disp shift


' -----[ Main Code ]-------------------------------------------------------
'
Main:
  LCDOUT LCDpin,ClrLCD,["1-Wire"]               ' splash screen
  LCDOUT LCDpin,(Line2 + 6),["Identifier"]
  PAUSE 2500

  GOSUB Device_Check                            ' look for device
  IF (devCheck <> NoDevice) THEN Get_ROM

No_Device_Found:
  LCDOUT LCDpin,ClrLCD,["No 1-Wire device."]
  END

Get_ROM:
  OWOUT OWpin,OW_FERst,[ReadROM]                ' send Read ROM command
  OWIN  OWpin,OW_BERst,[STR romData\8]          ' read serial number & CRC

  IF (romData(7) < $FF) THEN Show_Device
  LCDOUT LCDpin,ClrLCD,["Bad device?"]
  END

Show_Device:
  LCDCMD LCDpin,ClrLCD
  GOSUB Display_Device_Type

  ' serial number
  LCDCMD LCDpin,Line2
  FOR idx = 6 TO 1
    LCDOUT LCDpin,NoCmd,[HEX2 romData(idx)]
  NEXT
```

```
  ' checksum
  LCDOUT LCDpin,NoCmd,["  ",HEX2 romData(7)]

  END


' -----[ Subroutines ]------------------------------------------------------
'

' This subroutine checks to see if any 1-Wire devices are present on the
' bus.  It does NOT search for ROM codes
'
Device_Check:
  devCheck = 0
  OWOUT OWpin,OW_FERst,[SearchROM]              ' reset and start search
  OWIN  OWpin,OW_BitMode,[devCheck.Bit1,devCheck.Bit0]
  RETURN


' This subroutine is used to display the part number of a 1-Wire device.
' The text data and pointers to it are stored in the EE of a different
' program slot.

Display_Device_Type:
  addr = devType * 2 + $600                     ' calulate string pointer addr
  STORE NamesPgm                                ' point to names EEPROM
  READ addr,strPtr.LowByte                      ' get the string location
  READ addr+1,strPtr.HighByte

Read_Char:
  READ strPtr,char                              ' read character from string
  IF (char = 0) THEN Dev_Type_Done              ' at end? (0 = Yes)
  LCDOUT LCDpin,NoCmd,[char]                    ' no, print the char
  strPtr = strPtr + 1                           ' point to next char
  GOTO Read_Char
Dev_Type_Done:
  STORE 0                                       ' point to main pgm slot
  RETURN
```

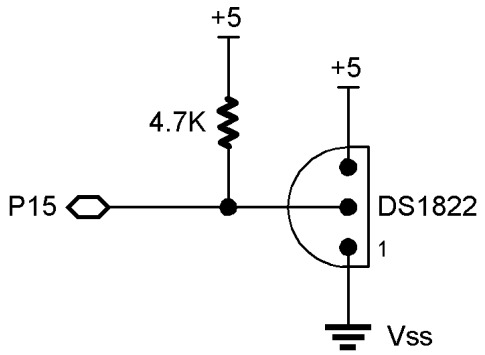
```
' -----[ Title ]-----------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_OWNAMES.BSP
' Purpose... Device names for Dallas 1-Wire components
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------
'
' There is no actual code in this module. This program stores the names
' of various Dallas Semiconductor 1-Wire devices.  The device family code
' is used to map the strings in EEPROM.  The pointer to a device's string
' discription is stored at the location determined by this formula:
'
'    pointer = device_id * 2 + $600
'
```

```
' "pointer" is actually the low-byte of the address location.  The high
' byte is at pointer+1.


' -----[ Constants ]-------------------------------------------------------
'
PntrBase        CON     $600


' -----[ EEPROM Data ]-----------------------------------------------------
'
' Store strings first so labels can be used in address pointer table
'
Unknown         DATA    "Unknown device",0

' shared family codes

FCode01         DATA    "DS1990/DS2401",0
FCode04         DATA    "DS1994/DS2404",0
FCode10         DATA    "DS1920/DS18S20",0
FCode14         DATA    "DS1971/DS2430",0
FCode23         DATA    "DS1973/DS2433",0

' iButtons

DS1920          DATA    "DS1920",0
DS1921          DATA    "DS1921",0
DS1963          DATA    "DS1963",0
DS1971          DATA    "DS1971",0
DS1973          DATA    "DS1973",0
DS1982          DATA    "DS1982",0
DS1985          DATA    "DS1985",0
DS1986          DATA    "DS1986",0
DS1990          DATA    "DS1990",0
DS1991          DATA    "DS1991",0
DS1992          DATA    "DS1992",0
DS1993          DATA    "DS1993",0
DS1994          DATA    "DS1994",0
DS1995          DATA    "DS1995",0
DS1996          DATA    "DS1996",0

' 1-Wire chips

DS1822          DATA    "DS1822",0
DS18B20         DATA    "DS18B20",0
DS18S20         DATA    "DS18S20",0
DS2401          DATA    "DS2401",0
DS2404          DATA    "DS2404",0
DS2405          DATA    "DS2405",0
DS2406          DATA    "DS2406",0
DS2417          DATA    "DS2417",0
DS2430          DATA    "DS2430",0
DS2433          DATA    "DS2433",0
DS2450          DATA    "DS2450",0
DS2505          DATA    "DS2505",0
DS2506          DATA    "DS2506",0
DS2890          DATA    "DS2890",0


' string pointers

Pointers        DATA    @$01*2+PntrBase,Word FCode01
                DATA    @$04*2+PntrBase,Word FCode04
```

```
                DATA    @$10*2+PntrBase,Word FCode10
                DATA    @$14*2+PntrBase,Word FCode14
                DATA    @$23*2+PntrBase,Word FCode23

                DATA    @$02*2+PntrBase,Word DS1991
                DATA    @$06*2+PntrBase,Word DS1993
                DATA    @$08*2+PntrBase,Word DS1992
                DATA    @$09*2+PntrBase,Word DS1982
                DATA    @$0A*2+PntrBase,Word DS1995
                DATA    @$0C*2+PntrBase,Word DS1996
                DATA    @$1A*2+PntrBase,Word DS1963
                DATA    @$21*2+PntrBase,Word DS1921

                DATA    @$05*2+PntrBase,Word DS2405
                DATA    @$0B*2+PntrBase,Word DS2505
                DATA    @$0F*2+PntrBase,Word DS2506
                DATA    @$12*2+PntrBase,Word DS2406
                DATA    @$20*2+PntrBase,Word DS2450
                DATA    @$22*2+PntrBase,Word DS1822
                DATA    @$27*2+PntrBase,Word DS2417
                DATA    @$28*2+PntrBase,Word DS18B20
                DATA    @$2C*2+PntrBase,Word DS2890
```

PP_DS1822.BSP

- Connect LCD to the BS2p Demo Board X5
- Install jumper X6
- Adjust contrast pot for best display
- Assemble DS1822 ciruit on breadboard
  -- use on-board 4.7K resistor (R1) for pull-up



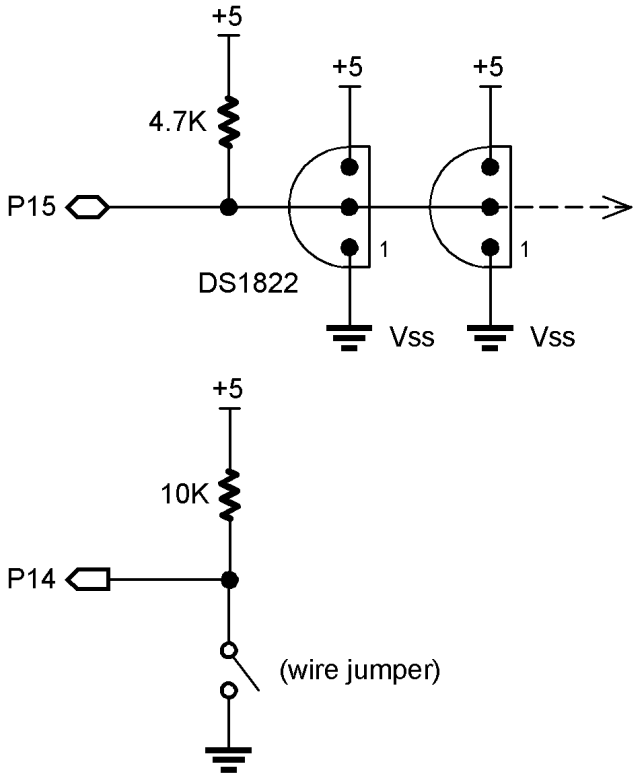

```
' -----[ Title ]-----------------------------------------------------------------
'
' BS2p Plus Pack
'
' File..... PP_DS1822.BSP
' Purpose... Reads and displays information from a Dallas DS1822
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------------
'
' This program demonstrates using the DS1822 in its simplest form for direct
' temperature measurement.  With only one sensor, we can use SkipROM and ignore
' the device serial number.
'
' Program output is via DEBUG.


' -----[ Revision History ]------------------------------------------------------
'


' -----[ I/O Definitions ]-------------------------------------------------------
'
OWpin           CON     15



' -----[ Constants ]-------------------------------------------------------------
'
```

```
' 1-Wire Support
'
OW_FERst        CON     %0001                   ' Front-End Reset
OW_BERst        CON     %0010                   ' Back-End Reset
OW_BitMode      CON     %0100
OW_HighSpd      CON     %1000

ReadROM         CON     $33                     ' read ID, serial num, CRC
MatchROM        CON     $55                     ' look for specific device
SkipROM         CON     $CC                     ' skip rom (one device)
SearchROM       CON     $F0                     ' search

' DS1822 control
'
CnvrtTemp       CON     $44                     ' do temperature conversion
RdScratch       CON     $BE                     ' read scratchpad

NoDevice        CON     %11                     ' no device present
DS1822          CON     $22                     ' device code
DegSym          CON     176


' -----[ Variables ]------------------------------------------------------------
'
devCheck        VAR     Nib                     ' device check return ocde
idx             VAR     Byte                    ' loop counter
romData         VAR     Byte(8)                 ' ROM data from DS1820
tempIn          VAR     Word                    ' raw temperature
sign            VAR     tempIn.Bit11            ' 1 = negative temperature
tLo             VAR     tempIn.LowByte
tHi             VAR     tempIn.HighByte
tSign           VAR     Bit
tempC           VAR     Word                    ' Celsius
tempF           VAR     Word                    ' Fahrenheit


' -----[ EEPROM Data ]----------------------------------------------------------
'


' -----[ Initialization ]-------------------------------------------------------
'
Initialize:
  DEBUG CLS
  PAUSE 250                                     ' allow DEBUG screen to open


' -----[ Main Code ]------------------------------------------------------------
'
Main:
  GOSUB Device_Check                            ' look for device
  IF (devCheck <> NoDevice) THEN Get_ROM

No_Device_Found:
  DEBUG CLS,"No DS1822 present.", CR
  DEBUG "-- Insert device and re-start."
  END

Get_ROM
  OWOUT OWpin,OW_FERst,[ReadROM]                ' send Read ROM command
  OWIN  OWpin,OW_BERst,[STR romData\8]          ' read serial number & CRC

  IF (romData(0) = DS1822) THEN Show_Data
```

```
  DEBUG "Installed device is not DS1822", CR
  DEBUG "-- Code = ",HEX2 romData(0)
  END

Show_Data:
  DEBUG Home, "DS1822 Data",CR,CR
  DEBUG "Serial Number : "
  FOR idx = 6 TO 1
    DEBUG HEX2 romData(idx)
  NEXT
  DEBUG CR,"     Checksum : ",HEX2 romData(7),CR,CR

Show_Raw:
  GOSUB Get_Temp
  DEBUG "    Raw Input : ",BIN16 tempIn,CR,CR

Display_Temperatures:
  DEBUG "       Temp C : ", SDEC tempC,DegSym,CR
  DEBUG "       Temp F : ", SDEC tempF,DegSym,CR

  PAUSE 1000
  GOTO Main
  END


' -----[ Subroutines ]------------------------------------------------------
'
' This subroutine checks to see if any 1-Wire devices are present on the
' bus.  It does NOT search for ROM codes
'
Device_Check:
  devCheck = 0
  OWOUT OWpin,OW_FERst,[SearchROM]              ' reset and start search
  OWIN  OWpin,OW_BitMode,[devCheck.Bit1,devCheck.Bit0]
  RETURN


Get_Temp:
  OWOUT OWpin,OW_FERst,[SkipROM,CnvrtTemp]      ' send conversion command
  PAUSE 500                                     ' give it some time
  OWOUT OWpin,OW_FERst,[SkipROM,RdScratch]      ' go get the temperature
  OWIN  OWpin,OW_BERst,[tLo,tHi]

  tSign = sign                                  ' save sign bit
  tempC = tempIn
  tempC = tempC >> 4                            ' round to whole degrees
  IF (tSign = 0) THEN NoNegC
  tempC = tempC | $FF00                         ' extend sign bits for negs

NoNegC:
  tempF = tempC */ $01CD                        ' multiply by 1.8
  IF tSign = 0 THEN NoNegF                      ' if neg, extend sign bits
  tempF = tempF | $FF00

NoNegF:
  tempF = tempF + 32                            ' finish C -> F conversion
  RETURN
```

PP_DS1822-2.BSP

- Connect LCD to the BS2p Demo Board X5
- Install jumper X6
- Adjust contrast pot for best display
- Assemble DS1822 ciruit on breadboard
  -- use on-board 4.7K resistor (R1) for 1-Wire pull-up




```
' -----[ Title ]-----------------------------------------------------------------
'
' BS2p Plus Pack
'
' File..... PP_DS1822-2.BSP
' Purpose... Reads and displays information from two Dallas DS1822
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------------
'
' This program demonstrates individual device addressing with the Dallas
' 1-Wire bus.  Before running the program, the individual device addresses
' must be determined and stored in the EEPROM.
'
```

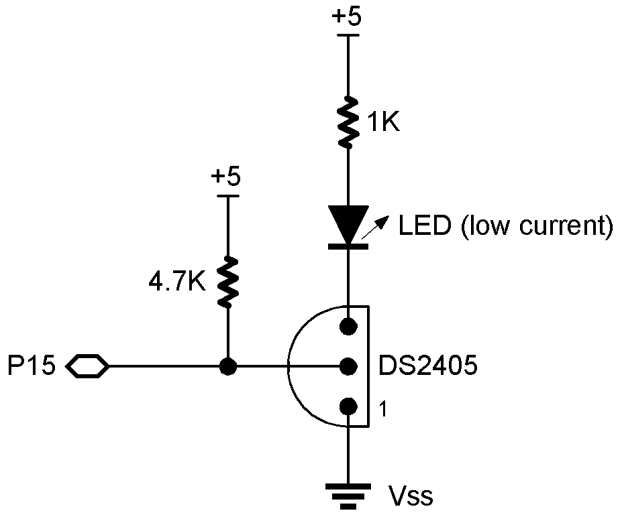
```
' Use the program PP_OWID.BSP to determine the data table for your
' DS1822 sensors.
'
' To run this program on the BS2p Demo Board, connect the LCD and
' install Jumper X6. Adjust contrast pot for best display.
'
' Refer to the Hitachi HD44780 documentation for details on LCD control.


' -----[ Revision History ]---------------------------------------------------
'


' -----[ I/O Definitions ]----------------------------------------------------
'
OWpin           CON     15
TempType        VAR     In14                    ' 1 = Fahrenheit
LCDpin          CON     0                       ' connect LCD to OutL


' -----[ Constants ]----------------------------------------------------------
'
NoCmd           CON     $00                     ' No command in LCDOUT
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control
CGRam           CON     $40                     ' Custom character RAM
Line1           CON     $80                     ' DDRAM address of line 1
Line2           CON     $C0                     ' DDRAM address of line 2

DegSym          CON     223                     ' degrees symbol

' 1-Wire Support
'
OW_FERst        CON     %0001                   ' Front-End Reset
OW_BERst        CON     %0010                   ' Back-End Reset
OW_BitMode      CON     %0100
OW_HighSpd      CON     %1000

ReadROM         CON     $33                     ' read ID, serial num, CRC
MatchROM        CON     $55                     ' look for specific device
SkipROM         CON     $CC                     ' skip ROM (one device)
SearchROM       CON     $F0                     ' search

' DS1822 control
'
CnvrtTemp       CON     $44                     ' do temperature conversion
RdScratch       CON     $BE                     ' read scratchpad

TC              CON     0                       ' read in Celcius
TF              CON     1                       ' read in Fahrenheit
NumSensors      CON     2                       ' inside and outside

' -----[ Variables ]---------------------------------------------------------
'
sensor          VAR     Nib                     ' sensor number to process
idx             VAR     Nib                     ' loop counter
eeAddr          VAR     Byte                    ' ee address of ROM match
romData         VAR     Byte(8)                 ' ROM data to DS1822
```

```
tempIn          VAR     Word                    ' raw temperature
sign            VAR     tempIn.Bit11            ' 1 = negative temperature
tInLow          VAR     tempIn.LowByte
tInHigh         VAR     tempIn.HighByte
tSign           VAR     Bit

tempC           VAR     Word                    ' Celsius
tempF           VAR     Word                    ' Fahrenheit
char            VAR     Byte                    ' character for LCD
rjNum           VAR     tempIn                  ' right justified number
rjSign          VAR     Bit                     ' sign for rj number
pos             VAR     Byte                    ' position to print
digits          VAR     Nib                     ' digits in rjNum
width           VAR     Nib                     ' width of display


' -----[ EEPROM Data ]------------------------------------------------------
'
                ' ROM codes for connected sensors
                ' -- these values MUST be edited for your sensors

T_ID0           DATA    $22,$30,$34,$01,$00,$00,$00,$6C
T_ID1           DATA    $22,$85,$42,$01,$00,$00,$00,$71


                ' labels for temperature sensors

T_Label0        DATA    "INSIDE ",0
T_Label1        DATA    "OUTSIDE ",0


' -----[ Initialization ]--------------------------------------------------
'
Initialize:
  PAUSE 500                                     ' let the LCD settle
  LCDCMD LCDpin,%00110000 : PAUSE 5             ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0             ' 4-bit mode
  LCDCMD LCDpin,%00101000 : PAUSE 0             ' 2-line mode
  LCDCMD LCDpin,%00001100 : PAUSE 0             ' no crsr, no blink
  LCDCMD LCDpin,%00000110                       ' inc crsr, no disp shift


' -----[ Main Code ]--------------------------------------------------------
'
Main:
  LCDOUT LCDpin,ClrLCD,["    DS1822"]           ' splash screen
  LCDOUT LCDpin,Line2, ["  Thermometer"]
  PAUSE 2000
  LCDCMD LCDpin,ClrLCD


Show_Sensors:
  LOOKUP sensor,[T_ID0,T_ID1],eeAddr            ' point to ROM code
  GOSUB Get_Temp                                ' get temperature

  LOOKUP sensor,[T_Label0,T_Label1],eeAddr      ' display sensor label
  LCDCMD LCDpin, Line1
  GOSUB Print_Label

  width = 4
```

```
  pos = Line2 + 10
  rjNum = tempC
  IF (TempType = TC) THEN  Print_Temp
  rjNum = tempF

Print_Temp:
  GOSUB RJ_Print
  LCDOUT LCDpin,NoCmd,[DegSym,TempType * ("F" - "C") + "C"]

Next_Sensor:
  sensor = sensor + 1 // NumSensors
  PAUSE 5000
  GOTO Show_Sensors

  END


' -----[ Subroutines ]------------------------------------------------------
'
Get_Temp:
  FOR idx = 0 TO 7                              ' load ROM pattern
    READ (eeAddr+idx),romData(idx)
  NEXT

  OWOUT OWpin,OW_FERst,[MatchROM,STR romData\8,CnvrtTemp]
  PAUSE 500
  OWOUT OWpin,OW_FERst,[MatchROM,STR romData\8,RdScratch]
  OWIN  OWpin,OW_BERst,[tInLow,tInHigh]

  tSign = sign                                  ' save sign bit
  tempC = tempIn
  tempC = tempC >> 4                            ' round to whole degrees
  IF (tSign = 0) THEN NoNeg1
  tempC = tempC | $FF00                         ' extend sign bits for negs

NoNeg1:
  tempF = tempC */ $01CD                        ' multiply by 1.8
  IF (tSign = 0) THEN NoNeg2                    ' if neg, extend sign bits
  tempF = tempF | $FF00

NoNeg2:
  tempF = tempF + 32                            ' finish C -> F conversion
  RETURN


Print_Label:
  READ eeAddr,char                              ' get a character
  IF (char = 0) THEN Print_Done                 ' if 0, exit
  LCDOUT LCDpin,NoCmd,[char]                    ' send to LCD
  eeAddr = eeAddr + 1                           ' move to next char address
  GOTO Print_Label
Print_Done:
  RETURN


RJ_Print:
  rjSign = rjNum.Bit15                          ' save sign
  rjNum = ABS(rjNum)                            ' convert to positive
  digits = width
  LOOKDOWN rjNum,<[0,10,100,1000,65535],digits
  LCDOUT LCDpin,pos,[REP " "\(width-digits-1),13 * rjSign + " ",DEC rjNum]
  RETURN
```

PP_DS2405.BSP

- Assemble DS2405 circuit on breadboard
  -- use on-board 4.7K resistor (R1) for 1-Wire pull-up



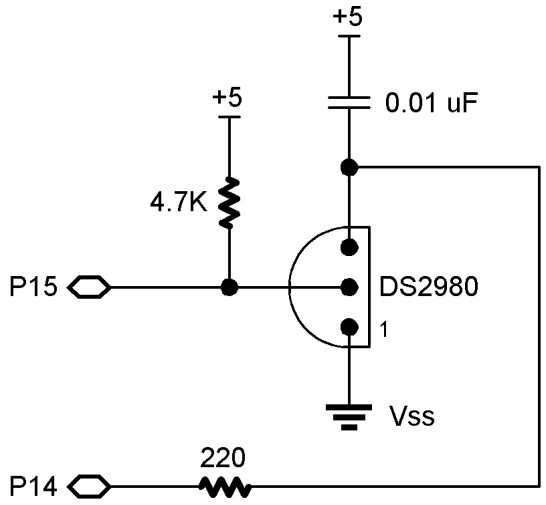

```
' -----[ Title ]-----------------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_DS2405.BSP
' Purpose... Simple DS2405 addressable switch demo
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------------
'
' Flashes an LED with a DS2405 addressable switch.  In this demo, the LED
' status follows a bit in a randomly generated variable.

' Note that the sink capability of the DS2405 is only 4 mA.  Use at least 1K
' in series with the LED.
'
' Note: The DS2405 is toggled.  Its current status is available only after
' addressing the device -- which also toggles it.  Status (0 = on) of the
' device is read in bit mode after addressing it with Match ROM.
'
' Refer to PP_DS1822-2.BSP for an example of dealing with multiple 1-Wire
' devices on the same pin.


' -----[ Revision History ]------------------------------------------------------
'
```

```
' -----[ I/O Definitions ]-------------------------------------------------
'
OWpin           CON     15                      ' 1-Wire bus


' -----[ Constants ]-------------------------------------------------------
'
On              CON     0                       ' pull cathode to ground
Off             CON     1

' 1-Wire Support
'
OW_FERst        CON     %0001                   ' Front-End Reset
OW_BERst        CON     %0010                   ' Back-End Reset
OW_BitMode      CON     %0100
OW_HighSpd      CON     %1000

ReadROM         CON     $33                     ' read ID, serial num, CRC
MatchROM        CON     $55                     ' look for specific device


' -----[ Variables ]-------------------------------------------------------
'
romData         VAR     Byte(8)                 ' ROM data from device
idx             VAR     Nib
rndValue        VAR     Word                    ' random value
flags           VAR     rndValue.LowByte        ' flags byte
ledFlag         VAR     flags.Bit7              ' LED flag
status          VAR     bit


' -----[ EEPROM Data ]-----------------------------------------------------
'


' -----[ Initialization ]--------------------------------------------------
'
Get_SN:
  DEBUG CLS
  PAUSE 250                                     ' let debug window open
  DEBUG "DS2405 - Reading serial number"
  OWOUT OWpin,OW_FERst,[ReadROM]                ' send Read ROM command
  OWIN  OWpin,OW_BERst,[STR romData\8]          ' read serial number & CRC
  DEBUG CR, "         "
  FOR idx = 6 TO 1                              ' display serial number
    DEBUG HEX2 romData(idx)
  NEXT
  PAUSE 1000


' -----[ Main Code ]-------------------------------------------------------
'
Main:
  DEBUG CLS
  DEBUG "DS2405 Digital Switch Demo (LED = flags.Bit7)"

Show_Flags
  PAUSE 1000                                    ' delay between tests
  GOSUB Shake_Flags                             ' randomize output
  DEBUG Home, CR, CR, "Flags... ", BIN8 flags, CR
  IF (ledFlag) THEN LED_On
```

```
LED_Off:
  DEBUG "LED..... Off"
  IF (status = Off) THEN Show_Flags             ' if off, shake flags
  GOTO Toggle_LED                               ' otherwise, turn off

LED_On:
  DEBUG "LED..... On "
  IF (status = On) THEN Show_Flags              ' if on, shake flags
  GOTO Toggle_LED                               ' otherwise, turn on


' -----[ Subroutines ]------------------------------------------------------------
'
Shake_Flags:
  FOR idx = 0 TO 15                             ' let all bits change
    RANDOM rndValue
  NEXT
  RETURN


Toggle_LED:
  OWOUT OWpin, OW_FERst, [MatchROM, STR romData\8]
  OWIN OWpin, OW_BitMode + OW_BERst, [status]   ' get new status
  GOTO Show_Flags
```

PP_DS2890.BSP

- Assemble DS2890 circuit on breadboard
  -- use on-board 4.7K resistor (R1) for 1-Wire pull-up




```
' -----[ Title ]-----------------------------------------------------------------
'
' BS2p Plus Pack
'
' File...... PP_DS2890.BSP
' Purpose... Demonstrates DS2980 digital pot by adjusting an RC circuit
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 26 SEP 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------------
'
' This program demonstrates the DS2980 digital pot by placing it in an RC
' circuit that would typically be used with RCTIME.  This program assumes only
' one device on the 1-Wire bus and reads the ROM code for the connected
' device.
'
' Refer to PP_DS1822-2.BSP for an example of dealing with multiple 1-Wire
' devices on the same pin.


' -----[ Revision History ]------------------------------------------------------
'


' -----[ I/O Definitions ]-------------------------------------------------------
'
OWpin           CON     15                      ' 1-Wire bus
RCpin           CON     14                      ' RCTIME pin
```

```
' -----[ Constants ]-------------------------------------------------------
'
' 1-Wire Support
'
OW_FERst        CON     %0001                   ' Front-End Reset
OW_BERst        CON     %0010                   ' Back-End Reset
OW_BitMode      CON     %0100
OW_HighSpd      CON     %1000

ReadROM         CON     $33                     ' read ID, serial num, CRC
MatchROM        CON     $55                     ' look for specific device

' DS2890 control
'
RdPot           CON     $F0                     ' read pot position
WrPot           CON     $0F                     ' write pot position
RdCtrl          CON     $AA                     ' read control register
WrCtrl          CON     $55                     ' write control register
IncPot          CON     $C3                     ' increment resistance
DecPot          CON     $99                     ' decrement resistance
Release         CON     $96                     ' release new pot setting


' -----[ Variables ]-------------------------------------------------------
'
romData         VAR     Byte(8)                 ' ROM data from device
idx             VAR     Byte
temp            VAR     Byte
potLevel        VAR     Byte
rcValue         VAR     Word


' -----[ EEPROM Data ]-----------------------------------------------------
'


' -----[ Initialization ]--------------------------------------------------
'
Get_SN:
  DEBUG CLS
  PAUSE 250                                     ' let debug window open
  DEBUG "DS2980 - Reading serial number"
  OWOUT OWpin,OW_FERst,[ReadROM]                ' send Read ROM command
  OWIN  OWpin,OW_BERst,[STR romData\8]          ' read serial number & CRC
  DEBUG CR, "         "
  FOR idx = 6 TO 1                              ' display serial number
    DEBUG HEX2 romData(idx)
  NEXT
  PAUSE 1000

  DEBUG CLS
  DEBUG "DS2980 - Setting control register"

Set_Ctrl:
  ' wiper 1, charge pump off
  OWOUT OWpin, OW_FERst, [MatchROM, STR romData\8, WrCtrl, %00001100]
  OWIN OWpin, OW_BERst, [temp]                  ' read back ctrl data
  IF (temp = $FF) THEN Set_Ctrl                 ' $FF = invalid ctrl reg val
  PAUSE 1000
```

```
' -----[ Main Code ]-----------------------------------------------------------
'
Main:
  DEBUG CLS
  DEBUG "DS2980 1-Wire Pot Demo", CR, CR
  DEBUG "Pot Setting... ",CR
  DEBUG "RC Value..... "

Demo_Pot:
  FOR potLevel = 255 TO 0 STEP 5
    DEBUG Home, CR, CR, "Pot Setting... ", DEC potLevel, "   ", CR
    ' send new level to pot
    OWOUT OWpin, OW_FERst, [MatchROM, STR romData\8, WrPot, potLevel]
    OWIN OWpin, 0, [temp]                         ' read new level back
    IF (temp <> potLevel) THEN Pot_Error          ' check for correct receipt
    OWOUT OWpin, OW_BERst, [Release]              ' release new value
    PAUSE 100

    HIGH RCpin                                    ' dicharge RC cap
    PAUSE 1
    RCTIME RCpin, 1, rcValue                      ' read pot
    DEBUG "RC Value..... ", DEC rcValue, "    "
    PAUSE 500
  NEXT

  GOTO Main
  END

' -----[ Subroutines ]----------------------------------------------------------
'
Pot_Error:
  DEBUG CLS
  DEBUG "Error -- could not set pot"
  END
```