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 Professional Starter Kit (#27235)

Introduction

The BS2p Professional Starter Kit is designed to provide the engineer with all the parts required to get started with Parallax's BS2p microcontroller. The kit includes a small selection of components and ready-to-run source code that will help you master some of the exciting new features of the BS2p; specifically the use of Philips I^2C™ components and Dallas Semiconductor 1-Wire® components.

Please note that this AppKit is designed for intermediate to advanced users. The schematics and source code (available for download from www.parallaxinc.com) 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) that is included in the kit. Should you desire more space for connecting components, please consider the NX-1000 lab board (#28135).

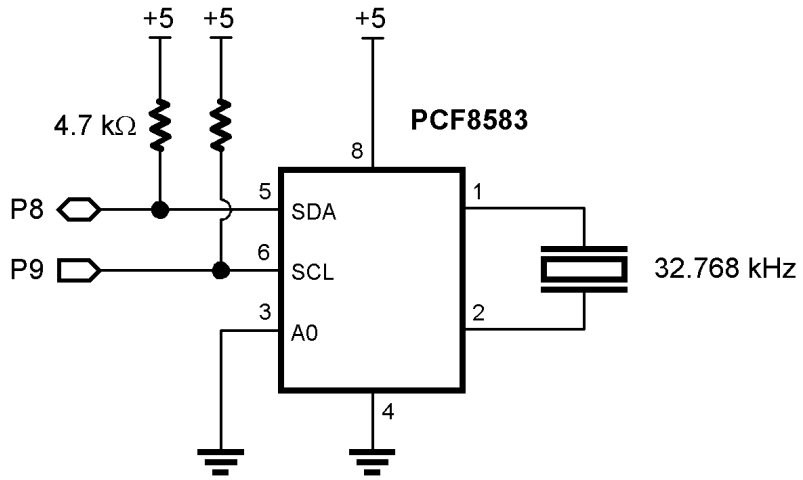
Packing List

Verify that your BS2p Professional Starter Kit package is complete in accordance with the list below. The contents of the package include:

- Packing List (this page)
- BASIC Stamp Programming Manual, Version 2.0
- BS2p Demo Board
- BS2p-24 Microcontroller
- Programming cable (DB9-to-DB9 serial)
- PCF8583 Clock/Calendar with 240 x 8-Bit RAM (I^2C)
- 32.678 kHz crystal
- 24LC32 Serial EEPROM (I2C)
- DS1822 Thermometer (1-Wire)
- (3) 4.7 kΩ resistor
- Pack of jumper wires

PSK_PCF8583.BSP

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




```
' -----[ Title ]-----------------------------------------------------------------
'
' BS2p Professional Starter Kit
'
' File..... PSK_PCF8583.BSP
' Purpose... PCF8583 RTC Demo
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 10 DEC 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 day (by name) and
' time in the DEBUG window


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


' -----[ I/O Definitions ]-------------------------------------------------------
'
I2Cpin          CON     8                       ' SDA on 8; SCL on 9
RxD             CON     16                      ' serial receive (from DEBUG)
```

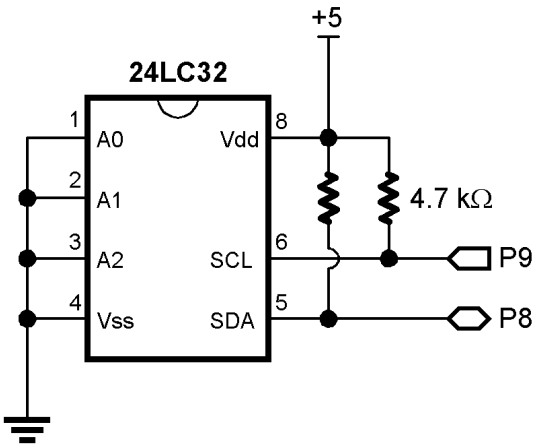
```
' -----[ Constants ]-------------------------------------------------------
'
DevType         CON     %1010 << 4              ' device type
DevAddr         CON     %000 << 1               ' address = %000 -> %001
Wr8583          CON     DevType | DevAddr       ' write to PCF8583
Rd8583          CON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

Yes             CON     1
No              CON     0

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

LF              CON     10                      ' linefeed


' -----[ 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 ]---------------------------------------------------
'
Splash:
  PAUSE 250                                     ' let DEBUG window open
  DEBUG CLS
  DEBUG "BS2p <--> PCF8583", CR, CR
```

```
Check_Set_Clock:
  DEBUG "Would you like to reset 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, CR, "Hours (0..23): "
  SERIN RxD, Baud96, [DEC2 hours]
  IF (hours < 24) THEN Enter_Minutes
  hours = 6                                     ' default to 6 AM on error


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                                       ' default to Sunday


Set_The_Clock:
  month = 12
  date = 10
  year = 1
  GOSUB Put_Clock


' -----[ Main Code ]----------------------------------------------------------
'
Main:
  DEBUG CLS, "BS2p <--> PCF8583"

Show_Clock:
  DEBUG Home, LF, LF
  GOSUB Get_Time_And_Day
  GOSUB Print_Day
  DEBUG DEC2 hours, ":", DEC2 minutes, ":", DEC2 seconds
  PAUSE 200
  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?
  DEBUG char
  eeAddr = eeAddr + 1                           ' point to next
  GOTO Print_Loop                               ' go get it

Print_Done:
  DEBUG CR
  RETURN
```

PSK_24LC32.BSP

- Assemble 24LC32 ciruit on breadboard
  -- use on-board 4.7 kΩ resistors (R1 and R2) for pull-ups



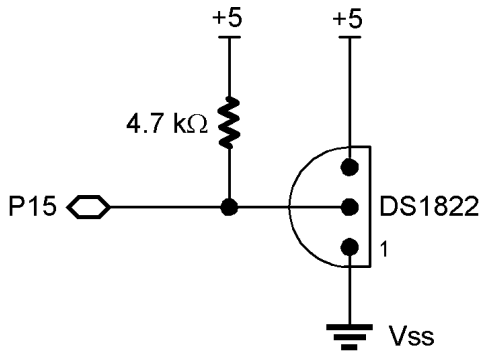

```
' -----[ Title ]-----------------------------------------------------------------
'
' BS2p Professional Starter Kit
'
' File...... PSK_24LC32.BSP
' Purpose... Demonstrates I2CIN and I2COUT
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 10 DEC 2001

' {$STAMP BS2p}


' -----[ Program Description ]---------------------------------------------------
'
' This program writes to and reads from a 24LC32 I2C EEPROM.  The status of the
' program and data are displayed in the DEBUG window.
'
' To run this program on the BS2p Demo Board, install the 24LC32 in the bread-
' board and connect to the BS2p with jumper wires.


' -----[ I/O Definitions ]--------------------------------------------------------
'
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
LF              CON     10                      ' linefeed


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


' -----[ Initialization ]--------------------------------------------------------
'
Splash:
  PAUSE 250                                     ' let DEBUG window open
  DEBUG CLS
  DEBUG "BS2p <--> I2C Memory", CR


' -----[ Main Code ]-------------------------------------------------------------
'
Main:
  FOR addr = 0 TO MaxEE STEP 5                  ' create addresses
    RANDOM rVar                                 ' create "random" value
    tOut = rVar.HighByte

    ' write valeu to 24LC32 then read it back

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

    ' display results

    DEBUG Home, LF, LF
    DEBUG "Addr... "                            ' display EE address
    width = 4
    temp = addr
    GOSUB RJ_Print
    DEBUG CR

    DEBUG " Out... "                            ' display value sent
    width = 4
    temp = tOut
    GOSUB RJ_Print
    DEBUG CR

    DEBUG "  In... "                            ' display value in
    width = 4
    temp = tIn
    GOSUB RJ_Print
    DEBUG CR
```

```
    PAUSE 250
  NEXT

  END


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

PSK_DS1822.BSP

- Assemble DS1822 ciruit on breadboard
  -- use on-board 4.7 kΩ resistor (R1) for pull-up




```
' -----[ Title ]-----------------------------------------------------------
'
' BS2p Professional Starter Kit
'
' File..... PSK_DS1822.BSP
' Purpose... Reads and displays information from a Dallas DS1822
' Author.... Parallax
' E-mail.... stamptech@parallaxinc.com
' Started...
' Updated... 10 DEC 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
```