APPLICATION NOTE

LCD6402B and BASIC Stamp-2 Interface

1 INTRODUCTION

This application note presents an example interface between an LCD6402B-P intelligent graphic LCD and a Parallax BASIC Stamp-2 (BS2).

1.1 HARDWARE INTERFACE

The application circuit shown in APPENDIX A consists of an LCD6402B-P intelligent graphic LCD and 5V powered BS2. The LCD6402B has a logic supply voltage of 3.3V so care is required when interfacing with 5V powered devices. This is achieved using the zener diode and resistor network formed by R1-4 and D1-2. Graphics can be programmed to the LCD6402B via the 'RS232 PROG PORT' using LCDLAB. Graphic objects (0-127) are recalled by writing the appropriate value to the LCD6402B command register.

1.2 SOFTWARE INTERFACE

The application code in this example is written in Parallax BASIC. See APPENDIX B for source code listing.

The LCD6402B is communicated to via the I2C bus and the functions for this are show in the code headed 'LCD64 INTERFACE FUNCTIONS '

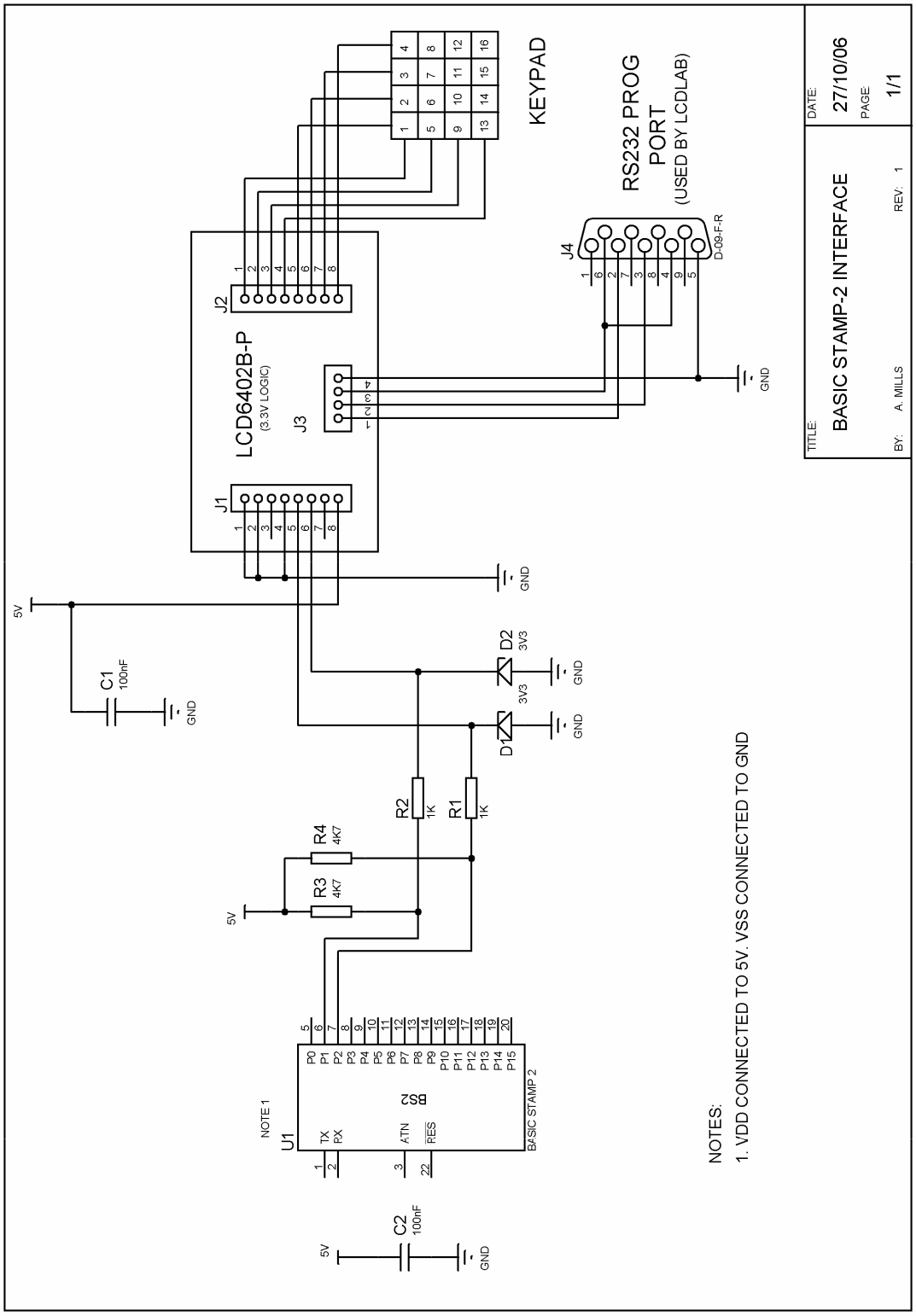
In general if a graphic function is used more than once then it's usually worth writing a wrapper function (see code headed 'LCD WRAPPER FUNCTIONS') to both save code and provide clarity. Examples of this are the functions LCD_Putc & LCD_PutData. More wrapper functions can be creates as required.

When power is applied, the backlight is switched on and the displayed cleared. The display is then rendered with a rectangle and some introductory text.

In the main program loop key presses are detected by reading the NEWKEY register. When the user presses a key its scan code is printed on the LCD.

This code could, with minimal modification, be made to work with other microcontrollers and compilers.

APPENDIX A. CIRCUIT SCHEMATIC



APPENDIX B. APPLICATION CODE

```
'/////////////////////////////////////////////////////////////////////
'////                          AN1002.BS2                         ////
'////                                                             ////
'////  See application note AN1002                                ////
'////                                                             ////
'////                                                             ////
'/////////////////////////////////////////////////////////////////////



'/////////////////////////////////////////////////////
'//                                                 //
'//                Declarations                     //
'//                                                 //
'/////////////////////////////////////////////////////

' I2C declarations
'
SCL         con 2                   ' I2C clock
SDA         con 1                   ' I2C data
SDAin       var in1
SDAout      var out1
SDAdir      var dir1

I2cBuf      var byte                ' I2c read/write buffer
I2cAddr     var byte                ' Address of I2C device
I2cReg      var byte                ' Register number within I2C
device
I2cData     var byte                ' Data to read/write
I2cAck      var bit                 ' Acknowledge bit
I2cAddr = $a0                       ' display device address


' LCD64 register
'
rCMD        con $20
rSTATUS     con $21
rNEWKEY     con $22
rINS        con $2B


' LCD64 commands
'
cmdCLEAR_SCREEN   con $94
cmdBACKLIGHT_ON   con $92


' LCD64 instructions
'
gRECT       con $5C
gPUTC       con $71
gFONT       con $69
gPUTS       con $80
gCURSOR     con $3A
```



```
' Application vars
'
byCMD       var byte
byINS       var byte
wdPointer   var word
byLoop      var byte                ' General purpose loop var
byResult    var byte                ' General purpose result var
pData       var word                ' Data pointer
byPar1      var byte                ' General purpose parameter var


' Data
'
dIntro1     data 5,gRECT,0,0,127,63
dIntro2     data 21,gCURSOR,16,4, gFONT,1, gPUTS,"Welcome to the",0
dIntro3     data 15,gCURSOR,30,16, gFONT,2, gPUTS,"LCD6402B",0
dIntro4     data 32,gCURSOR,8,42, gFONT,1
            data gPUTS,"You pressed key #:",0
dTextCur    data 3,gCURSOR,56,52

'////////////////////////////////////////////////////////
'//                                                    //
'//                 Start                              //
'//                                                    //
'////////////////////////////////////////////////////////

' Initialise
'
byCMD = cmdCLEAR_SCREEN : gosub LCD64_CMD ' clear screen
byCMD = cmdBACKLIGHT_ON : gosub LCD64_CMD ' backlight on


' Display introduction graphics and text
'
pData = dIntro1: gosub LCD_PutData
pData = dIntro2: gosub LCD_PutData
pData = dIntro3: gosub LCD_PutData
pData = dIntro4: gosub LCD_PutData




' Main program loop
'
MainLoop:
      ' Wait for key press
      gosub LCD64_KEY
      if byResult = 0 then MainLoop

      ' Display key no.
      pData = dTextCur: gosub LCD_PutData
      byPar1 = byResult dig 1 +"0": gosub LCD_Putc
      byPar1 = byResult dig 0 +"0": gosub LCD_Putc


goto MainLoop
```



```
'/////////////////////////////////////////////////////////////////////
'////                          AN1002.BS2                         ////
'////                                                             ////
'////  See application note AN1002                                ////
'////                                                             ////
'////                                                             ////
'/////////////////////////////////////////////////////////////////////


'/////////////////////////////////////////////////////
'//                                                 //
'//                Declarations                     //
'//                                                 //
'/////////////////////////////////////////////////////

' I2C declarations
'
SCL         con 2                   ' I2C clock
SDA         con 1                   ' I2C data
SDAin       var in1
SDAout      var out1
SDAdir      var dir1

I2cBuf      var byte                ' I2c read/write buffer
I2cAddr     var byte                ' Address of I2C device
I2cReg      var byte                ' Register number within I2C
device
I2cData     var byte                ' Data to read/write
I2cAck      var bit                 ' Acknowledge bit
I2cAddr = $a0                       ' display device address


' LCD64 register
'
rCMD        con $20
rSTATUS     con $21
rNEWKEY     con $22
rINS        con $2B


' LCD64 commands
'
cmdCLEAR_SCREEN   con $94
cmdBACKLIGHT_ON   con $92


' LCD64 instructions
'
gRECT       con $5C
gPUTC       con $71
gFONT       con $69
gPUTS       con $80
gCURSOR     con $3A


' Application vars
'
byCMD       var byte
```



```
byINS       var byte
wdPointer   var word
byLoop      var byte                ' General purpose loop var
byResult    var byte                ' General purpose result var
pData       var word                ' Data pointer
byPar1      var byte                ' General purpose parameter var


' Data (SizeOfData,Data0,Data1,Data2...)
'
dIntro1     data 5,gRECT,0,0,127,63
dIntro2     data 21,gCURSOR,16,4, gFONT,1, gPUTS,"Welcome to the",0
dIntro3     data 15,gCURSOR,30,16, gFONT,2, gPUTS,"LCD6402B",0
dIntro4     data 32,gCURSOR,8,42, gFONT,1
            data gPUTS,"You pressed key #:",0
dTextCur    data 3,gCURSOR,56,52

'////////////////////////////////////////////////////////
'//                                                    //
'//                  Start                             //
'//                                                    //
'////////////////////////////////////////////////////////

' Initialise
'
gosub LCD64_BusyWait
byCMD = cmdCLEAR_SCREEN : gosub LCD64_CMD ' clear screen
gosub LCD64_BusyWait
byCMD = cmdBACKLIGHT_ON : gosub LCD64_CMD ' backlight on


' Display introduction
'
pData = dIntro1: gosub LCD_PutData
pData = dIntro2: gosub LCD_PutData
pData = dIntro3: gosub LCD_PutData
pData = dIntro4: gosub LCD_PutData


' Main program loop
'
MainLoop:
      ' Wait for key press
      gosub LCD64_KEY
      if byResult = 0 then MainLoop

      ' Display key no.
      pData = dTextCur: gosub LCD_PutData
      byPar1 = byResult dig 1 +"0": gosub LCD_Putc
      byPar1 = byResult dig 0 +"0": gosub LCD_Putc


goto MainLoop


'////////////////////////////////////////////////////////
'//                                                    //
'//                LCD WRAPPER FUNCTIONS               //
'//                                                    //
```



```
'////////////////////////////////////////////////////////

'
' Print character in byPar1
'
LCD_Putc
      ' Wait for busy to clear
      i2creg = rSTATUS
      gosub I2cByteRead
      if (i2cdata & $80) <> 0 then LCD_Putc:

      i2creg = rINS
      i2cdata = gPUTC
      gosub i2cbytewrite
      i2cdata = byPar1
      gosub i2cbytewrite
return

'
' Put data pointed at by pData
'
LCD_PutData:

      ' Send graphic instructions
      read pData, byLoop
LCD_PD1:
      ' Wait for busy to clear
      i2creg = rSTATUS
      gosub I2cByteRead
      if (i2cdata & $80) <> 0 then LCD_PD1:


      pData = pData + 1
      read pData, i2cdata
      i2creg = rINS
      gosub i2cbytewrite
      byLoop = byLoop - 1     ' dec loop
      if byLoop <> 0 then LCD_PD1
return



'////////////////////////////////////////////////////////
'//                                                    //
'//                LCD64XX INTERFACE FUNCTIONS         //
'//                                                    //
'////////////////////////////////////////////////////////

'
' Read and return NEWKEY register
'
LCD64_KEY:
    i2creg = rNEWKEY          ' Read new key register
    gosub i2cbyteread
    byResult = i2cdata        ' return result
return

'
```



```
' Write command byCMD to LCD64 CMD register
'
LCD64_CMD:
    i2cdata = byCMD
    i2creg = rCMD
    gosub I2cByteWrite
return

'
' Write graphic instruction byINS to LCD64 INS register
'
LCD64_INS:
    i2cdata = byINS
    i2creg = rINS
    gosub I2cByteWrite
return

'
' Wait for LCD64 busy flag to clear
'
LCD64_BusyWait:
    ' Wait for busy to clear
    i2creg = rSTATUS
    gosub I2cByteRead
    if (i2cdata & $80) <> 0 then LCD64_BusyWait
return

'////////////////////////////////////////////////////////
'//                                                    //
'//                I2C FUNCTIONS                       //
'//                                                    //
'////////////////////////////////////////////////////////

'
' Writes I2cData to I2cReg at I2cAddr
'
I2cByteWrite:
    gosub I2cStart
    I2cBuf = I2cAddr
    gosub I2cOutByte                  ' device address
    I2cBuf = I2cReg
    gosub I2cOutByte                  ' register number
    I2cBuf = I2cData
    gosub I2cOutByte                  ' data
    gosub I2cStop
return

'
' Read I2cData from I2cReg
'
I2CByteRead:
    gosub I2cStart
    I2cBuf = I2cAddr
    gosub I2cOutByte                  ' device address
    I2cBuf = I2cReg
    gosub I2cOutByte                  ' register number
    gosub I2cStart                    ' start
    I2cBuf = I2cAddr | 1
    gosub I2cOutByte                  ' device address
```



```
    I2cAck = 0                          '
    gosub I2cInByte
    I2cData = I2cBuf                    ' read data
    gosub I2cStop
return

'
' Send I2C byte
'
I2cOutByte:
    shiftout SDA, SCL, MSBFIRST, [I2cBuf]
    input SDA
    high SCL                            ' clock in ack
    low SCL
return

'
' Get I2C byte
'
I2cInByte:
    shiftin SDA, SCL, MSBPRE, [I2cBuf]
    SDAout = 0
    SDAdir = I2cAck
    high SCL                            ' clock out ack
    low SCL
    input SDA
return

'
' Set I2C start condition
'
I2cStart
    high SDA
    high SCL
    low SDA
    low SCL
return

'
' Set I2C stop condition
'
I2cStop:
    low SDA
    high SCL
    high SDA
return
```

