Wirebridge Example Code for PowerBasic

There are currently 3 models of Wirebridge available.

Model 1A with 8 I/O pins, 3 channels of PWM output, SPI and I2C connectivity. This is 3.3Volt or 5V switched.

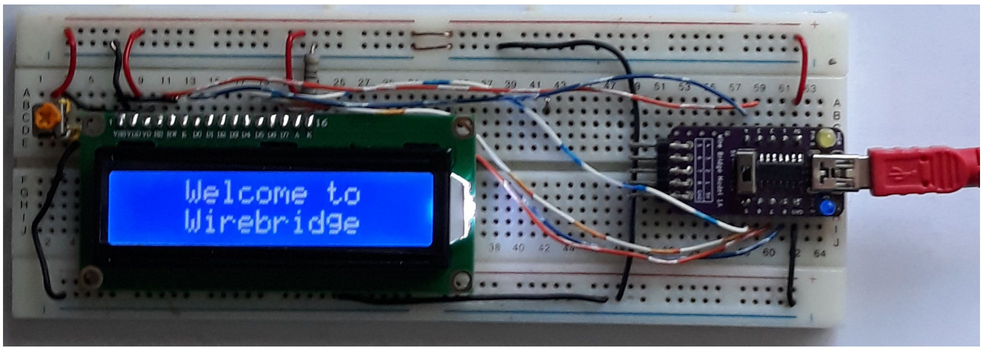
Model 1B with 8 I/O pins, 3 channels of PWM output, SPI and I2C connectivity. This is 5V only.

Model 2 with 6 Output pins, 2 input Pins and 2 PWM channels. This has open drain mosfet outputs which can tolerate up to 24v. The inputs can also handle voltages up to 24V if needed.

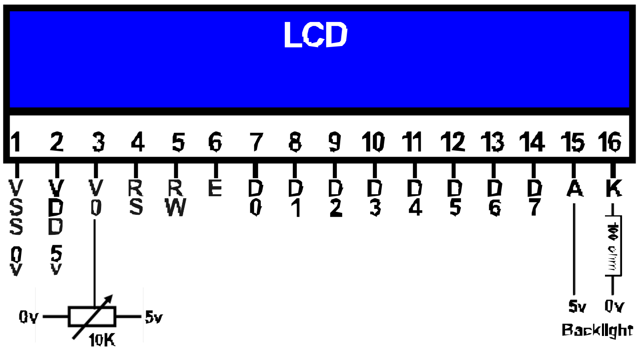
Model 1A

Sample code for driving a 2 line by 16 character LCD using **a** Hitachi HD44780 controller or compatible chipset. This will work with the model 1B. With modification it could be made to work with a Model 2 but would require resistors/ transistors and a powersupply, just not sensible.

Breadboard layout Model 1A



Note with 8 pins the Nibble (2 off 4bit data chunks) method is used to display info on the LCD using a total of 6 pins.



This is wired as follows:
LCD Pins 1(VSS) and 5(R/W) to ground (0V)
LCD Pin 2(VDD) to 5V
LCD Pin 3(V0) to a 10k pot for contrast. *
LCD Pin4 (RS) to Wirebridge pin 3
LCD Pin 6 (E) to Wirebridge pin 4
LCD pin 11 (D4) to Wirebridge pin 5
LCD pin 12 (D5) to Wirebridge pin 6
LCD pin 13 (D6) to Wirebridge pin 7
LCD pin 14 (D7) to Wirebridge pin 8

*Anything between 1 and 10k is OK.

The Wirebridge 5V and Ground pins provide the power, a WB can provide upto 100ma but the LCD only needs 10-20ma depending on the backlight so this is fine.

To drive the LCD a sample PowerBasic Program is provided as source code and compiled.

The program demonstrates the following functions but is not intended as a PowerBasic Tutorial. In the Wirebridge.inc file the function calls are detailed along with the types of variable used. For example the WB_SetMultiplePinsDigital Function is as follows.

```
DECLARE FUNCTION WB_SetMultiplePinsDigital LIB "WireBridge.API.C32.dll" ALIAS
"WB_SetMultiplePinsDigital" _
        (BYVAL WB_Devhandle AS DWORD, _
         BYVAL WB_SMPD_ENTRY AS DWORD, _    'pointer to array of WPSMPD
         BYVAL PCOUNT AS DWORD, _
         BYVAL  WBERR AS WB_ERROR PTR)AS BYTE
```

The return is a byte – called wb_outcome to follow the convention in the API documentation for other languages. The WB_SMPD_ENTRY is a double word pointer (address) of an array of type WB_SMDP which is defined in the Wirebridge.inc as two bytes

```
  TYPE WB_SMPD_ENTRY
         WB_PIN_NUMBER AS BYTE 'pin_number internal of the WB there is an ENUM for this
         WB_SMPD_STATE AS BYTE 'state, this is defined by an ENUM.
  END TYPE
```

```
    ENUM WB_Model1A 'Byte)
      Pin1 = 5
      Pin2 = 4
      Pin3 = 13
      Pin4 = 12
      Pin5 = 11
      Pin6 = 10
      Pin7 = 9
      Pin8 = 8
      PWM_Channel1 = 5
      PWM_Channel2 = 13
      PWM_Channel3 = 11
      SPI_MOSI = 10
      SPI_MISO = 9
      SPI_SCK =  8
      I2C_SDA =  9      'i2c
      I2C_SCL =  8      'i2c
      LED_Pin =  8
    END ENUM
```

```
  ENUM WB_SMPD_STATE ‘Byte
       OutputLow   = &b00  ‘(0)
       OutputHigh  = &b01  ‘(1)
       OutputFloat = &b10  ‘(2)
       InputSet    = &b11  ‘(3)
  END ENUM

Usage
DIM Array(7) AS WB_SMPD_ENTRY
‘to set Array(0)to WB Model 1A Pin 1 and Output
‘high the following is needed.
Array(0).WB_PIN_NUMBER = WB_Model1A.Pin1
Array(0).WB_SMPD_STATE = WB_SMPD_STATE.Outputhigh
```

Other Functions used in this example are.

Result = WB_EnumerateDevicesBegin(WBERR)
wb_outcome = WB_EnumerateDevicesNext(DevHandle, DEVINFO,WBERR)
Result = WB_EnumerateDevicesEnd(DevHandle)
WB_DevHandle = WB_OpenDevice(VARPTR(wb_path) , WBERR)
Result = WB_BuildTimestamp(dtp)
wb_outcome = WB_SetPinDirection (WB_Devhandle, WB_PIN_NUMBER, WB_PIN_DIR,WBERR))
wb_outcome = WB_WritePinDigital(WB_Devhandle ,WB_PIN_NUMBER,WB_DIGITAL_PIN_VALUE, WBERR)
wb_outcome = WB_SetMultiplePinsDigital(WB_Devhandle,WB_SMPD_ENTRY,PCOUNT,WBERR)

The WBERR, which is a pointer to an error structure can be replaced with BYVAL %NULL if not required.

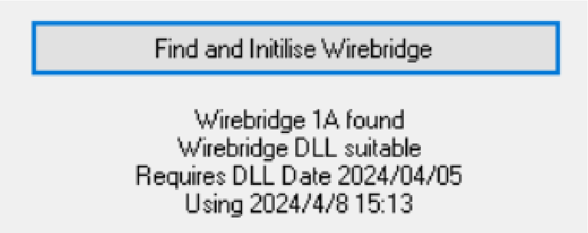
The first subroutine enumerates the Wirebridges to discover what is attached. This is generic and can be used as a template for other Wirebridge programs to list suitable available boards.

Example program subroutines used to communicate with the Wirebridge and on to the LCD.

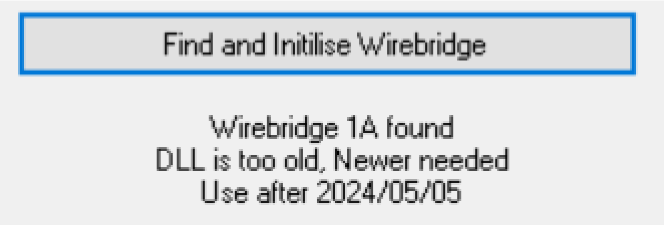
EnumWB (hdlg, WB_DEVICEHANDLE, reply, Myerror)

This Sub returns the WB device handle, as well as information on what is found and any error message. On exit it sets the WB_DEVICEHANDLE as a Global variable for ease of use.

```
SUB EnumWB(hdlg AS LONG,WB_DEVHANDLE AS DWORD, reply AS STRING, MyError AS STRING)

LOCAL WbErr AS WB_ERROR PTR         'Pointer to the error structure
LOCAL WB_ERR AS WB_ERROR            'Variable of type WB_ERROR
LOCAL DT AS WB_Datetime             'Variable of Type WB_DateTime
LOCAL DTP AS WB_DATETIME PTR        'Pointer to the WB_DateTime Structure
LOCAL result, I,J AS LONG
LOCAL WB_DEVENUMERATION AS DWORD    'variable WB_DEVENUMERATION used as handle
LOCAL wb_dev AS  WB_ENUMDEVINFO     'Variable of Type WB_ENUMDEVINFO
LOCAL devinfo AS WB_ENUMDEVINFO PTR 'Pointer to WB_ENUMDEVINFO structure
LOCAL wb_Outcome AS BYTE
LOCAL WB_Path AS WSTRINGZ * %MAX_PATH 'Path to the USB info
LOCAL model AS STRING

WBERR=VARPTR(WB_ERR)               'set the wberr pointer up
wb_path = STRING$$(%MAX_PATH, 32)  'sets up the path variable filled with spaces
wb_dev.USB_path = VARPTR (wb_path) 'Set the pointer to the string for the path
devinfo = VARPTR(wb_dev)           'Set the devinfo pointer up
dtp=VARPTR(DT)                     'Set the datetime pointer up

WB_DEVENUMERATION = WB_EnumerateDevicesBegin ( wberr) 'Call and retrieve the handle

IF WB_DEVENUMERATION=%False THEN             'Test if a handle is returned
  myerror=WB_ERRORTYPE(@wberr.WB_ERRORTYPE)  'extract the readable error
  EXIT SUB                                   'Bail out and return the error (MyError)
END IF
'Handle returned so find the Wirebridge(s) - this is set to exit on first it could be
'modified to give a list of found devices. This is left to the user.
DO
 wb_outcome = WB_EnumerateDevicesNext ( WB_DEVENUMERATION, devinfo, Wberr)
 IF wb_outcome  = 0  THEN              'No error then must have one
    wb_Path=wb_dev.@USB_PATH           'make a copy of the path get Device handle
    model = wb_dev.devinfo.productname ' extract the model name from the pointer
    IF INSTR (model,"1B") THEN
       reply="Wirebridge 1B found"   'comment out if not wanted
       EXIT LOOP                     'comment out if not wanted
    ELSEIF INSTR (model,"1A") THEN
       reply="Wirebridge 1A found"   'comment out if not wanted
       EXIT LOOP                     'comment out if not wanted
    ELSEIF INSTR (model,"2") THEN
     '  reply="Wirebridge 2 found"     'comment out if not wanted
     '  EXIT LOOP                      'comment out if not wanted
    END IF
  ELSE                                        'none attached
    myerror="Device not detected"             ' return a message
    EXIT SUB                                  ' Bail out
  END IF
LOOP
'Arrived with a model 1A or IB these are usable, first job is end the enumeration.
Result = WB_EnumerateDevicesEnd(WB_DEVENUMERATION)     'close Enumeration
'   Now Open the Wirebridge device and make the handle global for ease of use
WB_DevHandle = WB_OpenDevice( VARPTR(wb_path) , WBERR ) 'retrieve the handle
IF result=%False THEN                        'bad handle is an oops!
  myerror=WB_ERRORTYPE(@wberr.WB_ERRORTYPE)  'return error
  EXIT SUB                                   'and bail out
END IF

'getinformation from the DLL on its build to make sure its usable!
 result = WB_BuildTimestamp(dtp)
 ‘reply currently contains the model now check and add the DLL date and the
Wirebridge.in date, test it!
I= VAL(PARSE$($Dlldate,"/",1))*1000+VAL(PARSE$($Dlldate,"/",2))*40 + _
  +VAL(PARSE$($Dlldate,"/",3))
J= dt.year*1000+ DT.MONTH*40+ DT.Day
IF J=>I THEN                                         ' great its later or the same
   Reply = Reply+$CRLF+"Wirebridge DLL suitable"+$CRLF
ELSEIF I>J THEN                                       'problem
   Reply =Reply+$CRLF+"DLL is too old, Newer needed"+$CRLF
   Reply=Reply+"Use after "+$DLLDATE
   CONTROL SET USER hdlg,%IDC_BUT_ENUM,0,1 ‘disable buttons
   MyError="Wirebridge DLL out of date"
   EXIT SUB                                 ‘Bail Out
END IF

‘Update the reply text

 reply = reply +$CRLF+"Requires DLL Date "+$DLLDATE +$CRLF +_
          "Using"+STR$(dt.year)+"/"+TRIM$(STR$(DT.MONTH))+"/"+ _
          TRIM$(STR$(DT.Day))+STR$(DT.Hour)+":"+RIGHT$("0"+TRIM$(STR$(dt.minute)),2)

 GWB_DEVHANDLE = WB_DevHandle  ‘Finally setup the global gWb_devHandle

 END SUB
```



Wirebridge 1A found
Wirebridge DLL suitable
Requires DLL Date 2024/04/05
Using 2024/4/8 15:13



Wirebridge 1A found
DLL is too old, Newer needed
Use after 2024/05/05

Once the Wirebridge is found and is suitable and the DLL is OK then setup the LCD

There are 2 basic subroutines

SendCommand ***‘sends a command***
SendData ***‘send data***
These simply emulate how the LCD is used by setting pins and toggling the RS and E lines.

Everything else including setting up the LCD is done by calling these routines with data/commands

```
SUB sendcommand(cmd AS BYTE)
  LOCAL wb_outcome,LN ,HN, BT AS BYTE
  LOCAL WbErr AS WB_ERROR PTR
  LOCAL WB_ERR AS WB_ERROR
  LOCAL RS,E AS BYTE
  LOCAL P1,P2,P3,P4,P5,P6,p7,P8 AS BYTE
  LOCAL I AS LONG
  DIM pinmask(7) AS WB_SMPD_ENTRY
  LOCAL Pmask,pcount AS LONG'  Used as pointer

   WbErr = VARPTR (WB_ERR)
   'setup the LCD pins
   rs=%WB_Model1A.pin3
   E =%WB_Model1A.pin4
   pinmask(0).wb_pin_number=%WB_Model1A.pin5 ‘setup Pinmask() to actual pin
   pinmask(1).wb_pin_number=%WB_Model1A.pin6 ‘two blocks of 4 in this case
   pinmask(2).wb_pin_number=%WB_Model1A.pin7 ‘ for the two nibbles
   pinmask(3).wb_pin_number=%WB_Model1A.pin8
   pinmask(4).wb_pin_number=%WB_Model1A.pin5
   pinmask(5).wb_pin_number=%WB_Model1A.pin6
   pinmask(6).wb_pin_number=%WB_Model1A.pin7
   pinmask(7).wb_pin_number=%WB_Model1A.pin8
   'makesure E is high and RS low  as this is the command option.
   wb_outcome = WB_WritePinDigital(gWB_Devhandle ,E,%DP_Value.high, WBERR)     'Enable
   wb_outcome = WB_WritePinDigital(gWB_Devhandle ,RS,%DP_Value.low, WBERR)     'Command

  pinmask(0).wb_smpd_state = BIT(cmd,0) ‘this sets the mask to the
  pinmask(1).wb_smpd_state = BIT(cmd,1) ‘bit value in the BYTE
  pinmask(2).wb_smpd_state = BIT(cmd,2)
  pinmask(3).wb_smpd_state = BIT(cmd,3)
  pinmask(4).wb_smpd_state = BIT(cmd,4)
  pinmask(5).wb_smpd_state = BIT(cmd,5)
  pinmask(6).wb_smpd_state = BIT(cmd,6)
  pinmask(7).wb_smpd_state = BIT(cmd,7)
      'send HIGH nibble first that is Pinmask 4/5/6/7 Set the pointer to pinmask(4) and
      tell the WB to use the next 4 as WB_SMPD_ENTRY
    PMask = VARPTR(pinmask(4))  ' point to the High nibble it starts at pinmask(4)
    pcount=4                    '4 pins at once
   wb_outcome = WB_SetMultiplePinsDigital(gWB_Devhandle,pMask,PCOUNT,BYVAL %NULL)
   wb_outcome = WB_WritePinDigital(gWB_Devhandle ,E,%DP_Value.low, WBERR)
   wb_outcome = WB_WritePinDigital(gWB_Devhandle ,E,%DP_Value.high, WBERR)
   'now send LOW Nibble IE PinMask 0,1,2,3. 4 entries from PinMask(0)
   PMask = VARPTR(pinmask(0))    ' point to the Low nibble now
   pcount=4                      ‘ still 4 pins
   wb_outcome =  WB_SetMultiplePinsDigital(gWB_Devhandle,pMask,PCOUNT,BYVAL %NULL)
   wb_outcome = WB_WritePinDigital(gWB_Devhandle ,E,%DP_Value.low, WBERR)
   wb_outcome = WB_WritePinDigital(gWB_Devhandle ,E,%DP_Value.high, WBERR)

END SUB
```

The Sendtext subroutine is the same apart from setting RS to Data not Command, the two could be merged if wanted with a flag variable to switch between data and Command.

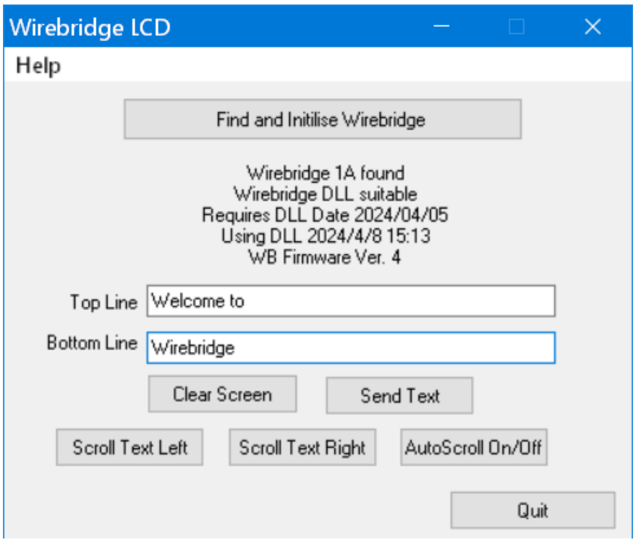
```
SUB  sendtext(cmd AS BYTE)

 ‘this is the change
  wb_outcome = WB_WritePinDigital(gWB_Devhandle ,RS,%DP_Value.high, WBERR)     ' data

END SUB
```

All that remains is to initialise the LCD screen. The Subroutine WB_SETUP does that,

```
SUB WB_Setup(hdlg AS LONG) 'To be here a WB has been found!
  LOCAL wb_outcome AS BYTE
  LOCAL WbErr AS WB_ERROR PTR
  LOCAL WB_ERR AS WB_ERROR
  LOCAL RS,E,D4,D5,D6,D7 AS BYTE
  LOCAL temp AS STRING
  LOCAL I AS LONG

   WbErr = VARPTR (WB_ERR)
   rs=%WB_Model1A.pin3     'setup the LCD pins
   E= %WB_Model1A.pin4
   D4=%WB_Model1A.pin5
   D5=%WB_Model1A.pin6
   D6=%WB_Model1A.pin7
   D7=%WB_Model1A.pin8
‘ Set the pin direction, in this case all output as we’re not reading the LCD
wb_outcome = WB_SetPinDirection ( gWB_Devhandle, %WB_Model1A.pin1 , %PinDir.Dir_out,BYVAL %NULL)
wb_outcome = WB_SetPinDirection ( gWB_Devhandle, %WB_Model1A.pin2 , %PinDir.Dir_out,BYVAL %NULL)
wb_outcome = WB_SetPinDirection ( gWB_Devhandle, %WB_Model1A.pin3 , %PinDir.Dir_out,BYVAL %NULL)
wb_outcome = WB_SetPinDirection ( gWB_Devhandle, %WB_Model1A.pin4 , %PinDir.Dir_out,BYVAL %NULL)
wb_outcome = WB_SetPinDirection ( gWB_Devhandle, %WB_Model1A.pin5 , %PinDir.Dir_out,BYVAL %NULL)
wb_outcome = WB_SetPinDirection ( gWB_Devhandle, %WB_Model1A.pin6 , %PinDir.Dir_out,BYVAL %NULL)
wb_outcome = WB_SetPinDirection ( gWB_Devhandle, %WB_Model1A.pin7 , %PinDir.Dir_out,BYVAL %NULL)
wb_outcome = WB_SetPinDirection ( gWB_Devhandle, %WB_Model1A.pin8 , %PinDir.Dir_out,BYVAL %NULL)

 sendcommand(2)     'this only works on initialization 2 is hi nibble of the command
 sendcommand(&H28) 'then 28 is the low nibble  this sets 2 lines, 4-bit mode
 sendcommand(&H0C) 'Home the cursor
 sendcommand(&H01) 'Clear screen
 sendcommand(&H02) 'home memory

END SUB
```

All the rest of the program is to do with the GUI, entering text, demonstrating sending the scroll command etc.
You are welcome to use any and all parts of the program in your own work.

This is a simple demonstration of writing to a LCD using the normal pin method. The pins on the 1A and 1B can be used for other purposes limited only by imagination.



Wirebridge 1A found
Wirebridge DLL suitable
Requires DLL Date 2024/04/05
Using DLL 2024/4/8 15:13
WB Firmware Ver. 4

Top Line: Welcome to
Bottom Line: Wirebridge

Other Programs demonstrate the I2C and SPI interfaces as well as the PWN functions are planned, The Model2 is interesting here as it has an open output and can be used to drive loads needing higher voltages and power than are available with the 1A and 1B