APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi 1-Wire Demo

DOCUMENT DATE: **13th April 2019**
DOCUMENT REVISION: **1.1**

Description

This application note demonstrates how to connect to one or more 1-wire sensors using a Picaso or Diablo16 Display Module. Before getting started, the following are required:

- Any of the following Picaso display modules:

uLCD-24PTU gen4-uLCD-24PT	uLCD-28PTU gen4-uLCD-28PT	uVGA-III gen4-uLCD-32PT

and other superseded display modules which support the ViSi environment

- The target module can also be a Diablo16 display

gen4-uLCD-24D Series	gen4-uLCD-28D Series	gen4-uLCD-32D Series
gen4-uLCD-35D Series	gen4-uLCD-43D Series	gen4-uLCD-50D Series
gen4-uLCD-70D Series		
uLCD-35DT	uLCD-43D Series	uLCD-70DT

Visit www.4dsystems.com.au/products to see the latest display module products that use the Diablo16 processor.

- 4D Programming Cable / µUSB-PA5/µUSB-PA5-II
  for non-gen4 displays (uLCD-xxx)
- 4D Programming Cable & gen4-IB / gen4-PA / 4D-UPA,
  for gen-4 displays (gen4-uLCD-xxx)
- micro-SD (µSD) memory card
- Workshop 4 IDE (installed according to the installation document)
- When downloading an application note, a list of recommended application notes is shown. It is assumed that the user has read or has a working knowledge of the topics presented in these recommended application notes.

Content

Description **2**
Content **3**
Application Overview **3**
Setup Procedure **4**
Create a New Project **4**
Design the Project **4**
Explanation of Applications in ZIP ***4***
1-Wire Devices ***5***
1-Wire-Discovery.4dViSi explained ***5***
1-Wire-Multiple.4dViSi Explained ***7***
Run the Program **9**
Proprietary Information **10**
Disclaimer of Warranties & Limitation of Liability **10**

Application Overview

This application note does not describe fully how to create the application example which is provided in the ZIP file of this application note. This application note aims to explain how to use the example so it can be modified or integrated into applications by the user.

This application note will not explain every step of the way, so please refer to the application examples in the ZIP file of this application note download, to see the finished applications.

Setup Procedure

For instructions on how to launch Workshop 4, how to open a **ViSi** project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of the application note

ViSi Getting Started - First Project for Picaso and Diablo16

Create a New Project

For instructions on how to create a new **ViSi** project, please refer to the section "**Create a New Project**" of the application note

ViSi Getting Started - First Project for Picaso and Diablo16

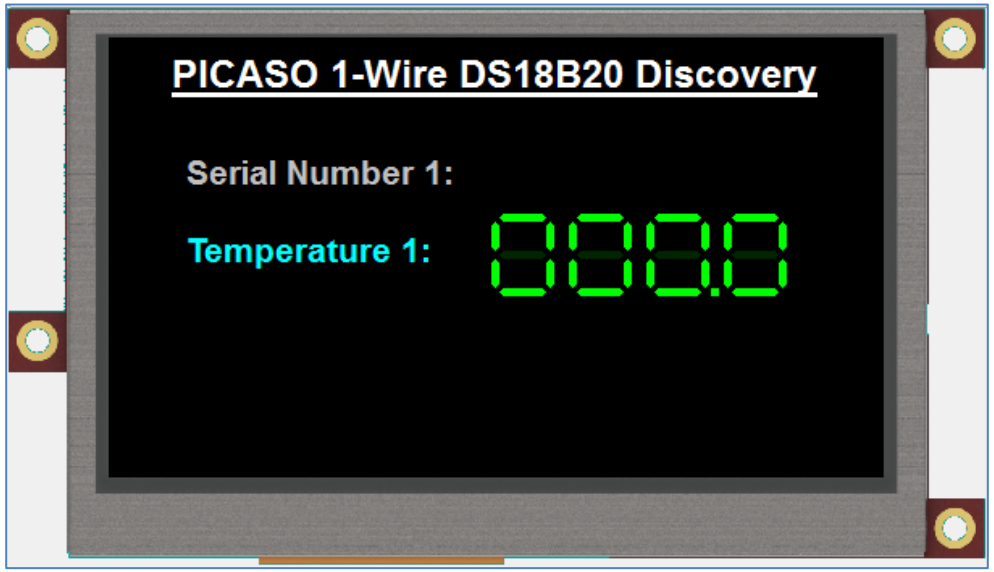
Design the Project

Explanation of Applications in ZIP

This Application Note contains 2 application examples inside the ZIP file. Start with the Application called 1-Wire-Discovery.4dViSi

This application example (1-Wire-Discovery.4dViSi) is designed to scan the 1-wire bus when a single 1-wire device is attached, in order to gain its Serial Number, so the 1-Wire device can be addressed when more than 1 device ends up being connected to the bus. It can also be used if only 1 device is going to be used, as it displays the Temperature also.

The Application example looks like the following:

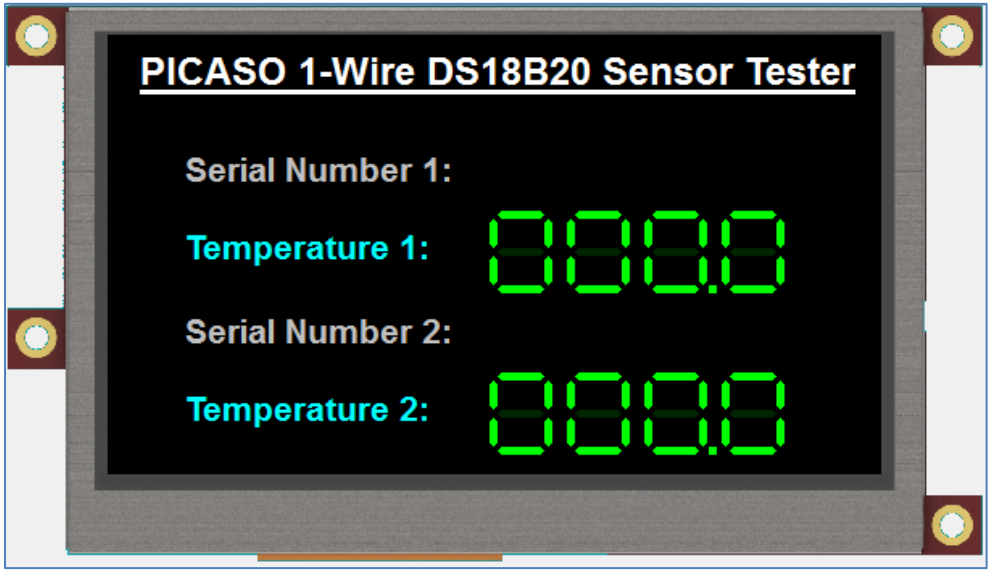


The second application in the ZIP is called 1-Wire-Multiple.4dViSi

This application example (1-Wire-Multiple.4dViSi) is designed to address 2 hard coded DS18B20 sensors, displaying the serial numbers that have been hard coded in, and the live temperatures presented from these sensors.

This application must be edited with the Serial Numbers discovered from the 1-Wire-Discovery.4dViSi application first in order to work.

The Application example looks like the following:



1-Wire Devices

There are numerous 1-wire devices available on the market. In this Application Note, DS18B20 Temperature Sensors have been used.

The 1-wire protocol from 4D Systems allows the bus to be scanned for a device if a single device is present on the bus, and the serial number extracted so the sensor can be addressed. If more than 1 devices are to be addressed on the bus, they initially need to be identified to the user individually so their serial number can be extracted. It is not possible to scan the bus when multiple devices are present.

1-Wire-Discovery.4dViSi explained

In the 1-Wire-Discovery.4dViSi application example, the full application (1-Wire-Multiple.4dViSi) has been cut down to aid with the understanding of what is going on in the application.

Open up this application by double clicking on the 1-Wire-Discovery.4dViSi file from Windows Explorer, or by going File – Open in Workshop 4.

Read the description at the start of the application

```
#platform "uLCD-43PT"

/*****************************************************************************
*                                                                            *
*  4D Visi 1-Wire Discovery - uLCD-43PT (Other Picaso & Diablo16 Processors too)  *
*                                                                            *
*  Date:        13th January 2014                                            *
*                                                                            *
*  Description: This demo provies an example of how to talk to 1 1-wire device  *
*               using a temporary include file created for the Picaso and Diablo16.  *
*               The demo illustrates how 1 DS18B20 Temperature Sensor can be  *
*               connected to a single Picaso or Diablo GPIO, and communicated with.  *
*               This demo is based on common knowledge obtained from the internet, used *
*               on a range of processors.                                    *
*               To use this demo on other Picaso or Diablo16 display modules, the  *
*               rescaling of the graphics may be required in order to fit the target  *
*               resolution.                                                  *
*               This application will present the Serial Number and temperature reading *
*               of a single 1-wire DS18B20 device on the bus.                *
*                                                                            *
*               This example will NOT WORK if more than 1 device is on the bus  *
*                                                                            *
*               A 4.7K Resistor from 5V or 3.3V to the GPIO of the 1-Wire bus is  *
*               required for this application to function, and for the 1-Wire bus to  *
*               function. As per the 1-Wire standard.                        *
*****************************************************************************/
```

The application is set to use pin IO5 on the uLCD-43PT. If you wish to use another pin, simply change the constant on line 38

```
38  #constant OW_PIN IO5_PIN        // define pin for 1-Wire
```

This section of code performs a 0x33 command which is essentially a broadcast the 1-wire bus and all sensors present on the bus will reply. This therefore only works correctly if 1 device is on the bus as they will all try to reply at the same time and the received message will be all gabled and therefore not usable.

```
// Read Serial Number 1 (ONLY POSSIBLE IF 1 DEVICE IS ON THE BUS! If >1 then this will fail)
OW_Write(0x33);                         // Send a command to read a serial number (ONLY IF SINGLE SENSOR PRESENT!)
for(temp := 0; temp < 8; temp++)
    serial_number1[temp] := OW_Read();     // Read 64-bit registration (48-bit serial number) number from 1-wire Slave Device
next

// Serial Number 1
txt_FontID(FONT3) ;
txt_FGcolour(BLUE) ;
txt_BGcolour(BLACK) ;
gfx_MoveTo(236 , 78) ;
for(temp := 0; temp < 8; temp++)
    hex_ascii(serial_number1[temp]);
next
```

This code will then print out the 16 byte serial number that has been discovered, to the display, in Hexadecimal.

For example, the output may be: 281A2AC9040000FB

This is made up of 16 bytes, 0x28, 0x1A, 0x2A etc

This number should be noted down if you are intending to use more the 1 device on the bus. If you only have 1 sensor, then you do not need to use the 1-Wire-Multiple.4DViSi application example.

The next section of code is the main loop, which will loop every 500ms, and communicate with the DS18B20 sensor and return its temperature value, and then display it on the display using the LED Digit object.

This achieves this by doing a 1-Wire bus reset, which is required prior to sending a new command to the bus. It then sends a 0x55, which tells the bus a Serial Number is going to follow, to address a specific sensor. Since the bus has already been scanned and saved above, the serial number has been saved into the serial_number[] array. This is then used when addressing the sensor.

```
99  /*************************************************************************************
100 * Overview:          Main Loop
101 *************************************************************************************/
102
103     repeat
104         // Read from Sensor # 1 using specific ROM Serial Number
105         OW_Reset();      // Reset the 1-wire Bus
106         OW_Write(0x55); // Rom Specific target command
107         for (c := 0; c < 8; c++)
108             OW_Write(serial_number1[c]); //64 bit ROM code to be matched
109         next
110         OW_Write(0x44); // start conversion command
111         OW_Reset();      // Reset the 1-wire Bus
112         OW_Write(0x55); // Rom Specific target command
113         for (c := 0; c < 8; c++)
114             OW_Write(serial_number1[c]); //64 bit ROM code to be matched
115         next
116         OW_Write(0xBE); // Read Scratchpad command
117
118         t1  := OW_Read();     // Read from the Scratchpad
119         t1  := t1 + (OW_Read() << 8);
120         rem := (((t1%16) * 100) / 16) / 10;  // Get remainder as percentage of 16 and divide by 10
121         t1 /= 16;                            // 0.0625 degrees per bit.  Can't multiply by 0.0625, divide by 16 same
122         t1 *= 10;                            // Muliply by 10 to pass to display
123         t1 += rem;                           // Add remainder as the decimal
124
125         if(t1 >= 0 && t1 <= 1000)            // If the value is between 0 and 1000
126             leddigitsDisplay(t1, iLeddigits1+1, 236, 4, 2, 48, 0) ; // Write to the LED Digit1
127         endif
128
129         pause(500);
130     forever
131
132 endfunc
```

The serial number is then sent to the bus using a for loop, writing each of the 8 bytes one at a time. The 0x44 command is then sent telling the DS18B20 to start a temperature conversion.

The bus is then reset again, and a 0x55 is sent once again, followed by the serial number. The 0xBE command is then sent telling the DS18B20 that the temperature is wanted to be read. The scratchpad is then read as 2 bytes, and then converted into a temperature.

The temperature is then sent to the LED Digits object on the display.

This application can be used if there is only 1 DS18B20 sensor on the bus, or to discover the serial number of multiple DS18B20 sensors 1 at a time. The serial number can then be noted down. Repeat process if more than 1 sensor is to be used, noting down the serial numbers. Using the 1-Wire-Multiple.4dViSi application, multiple devices can be then present on the bus at the same time.

1-Wire-Multiple.4dViSi Explained

The 1-Wire-Multiple.4dViSi application is the full application for this Application Note. It features the code from the 1-Wire-Discovery.4dViSi application, along with addition code used to address 2 DS18B20 sensors. The same principle can be used for more than 2 sensors, however this demo only features 1 form showing 2 sensor values.

Please take note of the description at the top of the application

```
1   #platform "uLCD-43PT"
2
3   /*************************************************************************************
4   *                                                                                    *
5   *  4D Visi 1-Wire Demonstation - uLCD-43PT (Other Picaso & Diablo16 Processors too)  *
6   *                                                                                    *
7   *  Date:        13th January 2014                                                    *
8   *                                                                                    *
9   *  Description: This demo provies an example of how to talk to 1 or more 1-wire devices *
10  *               using a temporary include file created for the Picaso and Diablo16.  *
11  *               The demo illustrates how 1 or more DS18B20 Temperature Sensors can be *
12  *               connected to a single Picaso or Diablo GPIO, and communicated with.  *
13  *               This demo is based on common knowledge obtained from the internet, used *
14  *               on a range of processors.                                            *
15  *               To use this demo on other Picaso or Diablo16 display modules, the    *
16  *               rescaling of the graphics may be required in order to fit the target *
17  *               resolution.                                                          *
18  *               This application will present the Serial Numbers and temperature     *
19  *               readings of 2 1-wire DS18B20 device on the bus. The serial number MUST *
20  *               be hard coded after discovering them with the 1-Wire-Discovery       *
21  *               application first.                                                   *
22  *                                                                                    *
23  *               A 4.7K Resistor from 5V or 3.3V to the GPIO of the 1-Wire bus is     *
24  *               required for this application to function, and for the 1-Wire bus to *
25  *               function. As per the 1-Wire standard.                                *
26  *                                                                                    *
27  *************************************************************************************/
```

As per the 1-Wire-Discovery.4dViSi explanation section, if the desired pin used for the 1-wire bus is desired to be anything other than IO5 on the uLCD-43PT, then this can be changed on line 39 of this application.

On line 83 is some code which is found in the 1-Wire-Discovery application, which has been commented out. This has been replaced with code below it, which stores

2 hard coded serial numbers in arrays, used to address 2 sensors on the bus, see lines 101 and 102.

```
91      // Read Serial Number 1 (ONLY POSSIBLE IF 1 DEVICE IS ON THE BUS! If >1 then this will fail)
92      // COMMENT OUT IF MORE THAN 1 DEVICE ON THE BUS. HAVE TO HARD CODE SERIAL NUMBER TO READ IF > 1)
93      /*****************************************************************************************************************
94      OW_Write(0x33);                         // Send a command to read a serial number (ONLY IF SINGLE SENSOR PRESENT!)
95      for(temp := 0; temp < 8; temp++)
96          serial_number1[temp] := OW_Read();     // Read 64-bit registration (48-bit serial number) number from 1-wire Slave Device
97      next
98      *****************************************************************************************************************/
99
100     // HARD CODED serial numbers, must do this if >1 sensor on the bus, else comment out and use the above
101     *serial_number1 := [0x28, 0xCA, 0x49, 0xC9, 0x04, 0x00, 0x00, 0x60];     // MUST Replace this with your Serial Number 1
102     *serial_number2 := [0x28, 0x1A, 0x2A, 0xC9, 0x04, 0x00, 0x00, 0xFB];     // MUST Replace this with your Serail Number 2
```

Code starting on line 105 will print the first serial number out to the display. Code starting on line 114 will print the second serial number to the display.

```
104     // Serial Number 1
105     txt_FontID(FONT3) ;
106     txt_FGcolour(BLUE) ;
107     txt_BGcolour(BLACK) ;
108     gfx_MoveTo(236 , 78) ;
109     for(temp := 0; temp < 8; temp++)
110         hex_ascii(serial_number1[temp]);
111     next
112
113     // Serial Number 2 (COMMENT OUT IF ONLY 1 SERIAL NUMBER)
114     txt_FontID(FONT3) ;
115     txt_FGcolour(BLUE) ;
116     txt_BGcolour(BLACK) ;
117     gfx_MoveTo(236 , 178) ;
118     for(temp := 0; temp < 8; temp++)
119         hex_ascii(serial_number2[temp]);
120     next
```

Inside the main loop starting on line 127, is the code which is used to retrieve the temperature of the sensors. This first section of code reads from the first sensor and then displays it on the first LED Digit on the display. Please refer to the 1-wire-Discovery application explanation section for detail.

```
123 /*****************************************************************************************
124 * Overview:          Main Loop
125 *****************************************************************************************/
126
127     repeat
128         // Read from Sensor # 1 using specific ROM Serial Number
129         OW_Reset();      // Reset the 1-wire Bus
130         OW_Write(0x55); // Rom Specific target command
131         for (c := 0; c < 8; c++)
132             OW_Write(serial_number1[c]); //64 bit ROM code to be matched
133         next
134         OW_Write(0x44); // start conversion command
135         OW_Reset();      // Reset the 1-wire Bus
136         OW_Write(0x55); // Rom Specific target command
137         for (c := 0; c < 8; c++)
138             OW_Write(serial_number1[c]); //64 bit ROM code to be matched
139         next
140         OW_Write(0xBE); // Read Scratchpad command
141
142         t1  := OW_Read();      // Read from the Scratchpad
143         t1  := t1 + (OW_Read() << 8);
144         rem := (((t1%16) * 100) / 16) / 10;  // Get remainder as percentage of 16 and divide by 10
145         t1 /= 16;                            // 0.0625 degrees per bit.  Can't multiply by 0.0625, divide by 16 same
146         t1 *= 10;                            // Muliply by 10 to pass to display
147         t1 += rem;                           // Add remainder as the decimal
148
149         if(t1 >= 0 && t1 <= 1000)            // If the value is between 0 and 1000
150             ledDigitsDisplay(t1, iLeddigits1+1, 236, 4, 2, 48, 0) ; // Write to the LED Digit1
151         endif
```

The code following that starting on line 157 will do the same but for the second sensor, and display it on the second LED digit of the display

```
153         /*****************************************************************************/
154         /*           COMMENT THIS SECTION OUT IF YOU ONLY HAVE 1 SENSOR               */
155
156         // Read from Sensor # 2 using specific ROM Serial Number
157         OW_Reset();      // Reset the 1-wire Bus
158         OW_Write(0x55); // Rom Specific target command
159         for (c := 0; c < 8; c++)
160             OW_Write(serial_number2[c]); //64 bit ROM code to be matched
161         next
162         OW_Write(0x44); // start conversion command
163         OW_Reset();      // Reset the 1-wire Bus
164         OW_Write(0x55); // Rom Specific target command
165         for (c := 0; c < 8; c++)
166             OW_Write(serial_number2[c]); //64 bit ROM code to be matched
167         next
168         OW_Write(0xBE); // Read Scratchpad command
169
170         t2  := OW_Read();                    // Read from the Scratchpad
171         t2  := t2 + (OW_Read() << 8);
172         rem := (((t2%16) * 100) / 16) / 10;  // Get remainder as percentage of 16 and divide by 10
173         t2 /= 16;                            // 0.0625 degrees per bit.  Can't multiply by 0.0625, divide by 16 same
174         t2 *= 10;                            // Muliply by 10 to pass to display
175         t2 += rem;                           // Add remainder as the decimal
176
177         if(t2 >= 0 && t2 <= 1000)            // If the value is between 0 and 1000
178             ledDigitsDisplay(t2, iLeddigits2+1, 236, 4, 2, 48, 0) ;  // Write to the LED Digit2
179         endif
180
181         /*****************************************************************************/
```

The program can be compiled and downloaded to the display.

It should also be noted that the 1-Wire-TEMP.inc file contains the code which defines the 1-wire Protocol, and is included in the Application example at line 37.

Run the Program

For instructions on how to save a **ViSi** project, how to connect the target display to the PC, how to select the program destination (this option is not available for Goldelox displays), and how to compile and download a program, please refer to the section "**Run the Program**" of the application note

ViSi Getting Started - First Project for Picaso and Diablo16

The uLCD-32PTU and uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other displays.

Proprietary Information

Disclaimer of Warranties & Limitation of Liability