



## Designer or ViSi: MOTG-BLUETOOTH

DOCUMENT DATE: **14<sup>th</sup> DECEMBER 2020**  
DOCUMENT REVISION: **1.0**



## Description

This application note provides instructions on how to setup and connect the MOTG- BLUETOOTH with other Bluetooth Low Energy (BLE) devices. Before getting started, the following are required:

### Hardware

- Any gen4-series 4D Systems display module powered by any of the following processors:
  - Diablo16
  - Picaso
- MOTG-BLUETOOTH
- Programming Adaptor for target display module
- One of the following Interface Boards:
  - gen4-MOTG-AC1 (For 2.4" - 3.5" Diablo16-based display)
  - gen4-MOTG-AC2 (For 3.2" - 3.5" Diablo16-based display)
  - gen4-MOTG-AC3 (For Picaso-based display)
  - gen4-MOTG-AC4 (For 4.3" Diablo16-based display)

### Software

- Workshop4
- **Serial Bluetooth Terminal (Android/PC)**
- **SmartData(iPhone)**

**Note:** Using a non-4D programming interface could damage the processor and void the warranty.

## Content

<b>Description .....</b>	<b>2</b>
<b>Content .....</b>	<b>2</b>
<b>Application Overview.....</b>	<b>3</b>
<b>Setup Procedure .....</b>	<b>3</b>
<b>Create a New Project .....</b>	<b>3</b>
<b>Design the Project.....</b>	<b>4</b>
<i>Initialization.....</i>	<i>4</i>
<i>Repeat-Forever Loop .....</i>	<i>4</i>
<b>The Hardware Connection.....</b>	<b>5</b>
<i>Configure the MOTG-BLUETOOTH .....</i>	<i>5</i>
<i>Hardware Connection .....</i>	<i>6</i>
<b>Build and Upload the Project.....</b>	<b>6</b>
<b>Demonstration.....</b>	<b>6</b>
<b>Proprietary Information .....</b>	<b>8</b>
<b>Disclaimer of Warranties &amp; Limitation of Liability .....</b>	<b>8</b>

## Application Overview

The MOTG-Bluetooth is an accessory module that can be used to add Bluetooth connectivity to gen4 display modules. MOTG-BLUETOOTH is powered by Microchip RN4870 which is a BLE module.

This application notes demonstrate to how to setup the MOTG- BLUETOOTH where another BLE device can pair with. When another device is connected, it can then send data and receive an echo from MOTG-BLUETOOTH.

The display in this application note uses a gen4-uLCD-32DCT-CLB which is a Diablo16 display module. To interface with the MOTG-Bluetooth, the gen4-MOTG-AC1 is used. If you are using a Picaso display module, you will need to use a gen4-MOTG-AC3.

Please refer to the **MOTG datasheet** to know more about the pin diagram.

**Note:** If you are want to use a Pixxi display module, you can use a MOTG-Breadtooth. You should also note that the Pixxi will need to utilize its COM0 which is also used when programming the display.

## Setup Procedure

For instructions on how to launch Workshop4, how to open a **Designer or ViSi** project, and how to change the target display, kindly refer to the section “**Setup Procedure**” of any of the following application notes:

- **Designer Getting Started - First Project**
- **ViSi Getting Started – First Project for Picaso and Diablo16**

## Create a New Project

For instructions on how to create a new **Designer or ViSi** project, please refer to the section “**Create a New Project**” of any of the following application notes:

- **Designer Getting Started - First Project**
- **ViSi Getting Started – First Project for Picaso and Diablo16**

## Design the Project

### Initialization

The MOTG-BLUETOOTH device is first reset before setting up serial communication. The hardware pin is set as an output using the function `pin_Set`. Then, the MOTG-BLUETOOTH is reset by pulling the reset pin low then high through the functions `pin_LO()` and `pin_HI()`. The preprocessor selection for Diablo and Picaso will automatically choose which code to use during compile and upload.

```

16 // Reset MOTG Bluetooth Module
17 #IF EXISTS DIABLO
18 pin_Set(PIN_OUT, PA4);
19 pin_LO(PA4);
20 pause(100);
21 pin_HI(PA4);
22 pause(1000);
23 #ENDIF // EXIST DIABLO
24
25 #IF EXISTS PICASO
26 pin_Set(OUTPUT, IO3_PIN);
27 pin_LO(IO3_PIN);
28 pause(100);
29 pin_HI(IO3_PIN);
30 pause(1000);
31 #ENDIF // EXIST PICASO

```

After the MOTG-BLUETOOTH is reset, the serial communication is then set. The Transmit(TX) pin and Receive(RX) pins need to be selected for the Diablo based display modules, which can be set using the `comX_RX_Pin()` and `comX_TX_Pin()` functions, this however is not needed for the Picaso based display modules since it is fixed.

```

33 // Setup serial
34 #IF EXISTS DIABLO // Setup hardware pin only for Diablo, Picaso is fixed on RX1 and TX1
35 COM1_RX_pin(PA2); // select the hardware pin for the COM1 receive line
36 COM1_TX_pin(PA3); // select the hardware pin for COM1 transmit line
37 #ENDIF

```

The `com_SetBaud` function then sets the speed of the communication port. Then the `comX_Init` function initializes the serial buffer.

```

38 com1_Init(serialBuffer, 50, 0);
39 com_SetBaud(COM1, 11520);
40

```

### Repeat-Forever Loop

By default, the MOTG-BLUETOOTH has transparent service enabled which is now ready for pairing. If another device is paired and connected it will then display the status on the screen.

```

41 repeat
42   if (com1_Count() > 0) // Check if buffer receive data from MOTG-BLUETOOTH
43     c := serin1();
44     if (c == '%') // Check for start and end of Status message
45       statusMsg++;
46       if (statusMsg == 2)
47         // End of status message
48         putch(c);
49         putch('\n');
50         statusMsg := 0;
51         continue;
52       endif
53     endif
54     if (statusMsg)
55       // Status message
56       // Do something to assemble status message
57       putch(c); // Print Status Message on Screen
58     else
59       // Actual UART message
60       serout1(c); // Echo character
61       // Do something to assemble actual message
62     endif
63   endif
64   forever

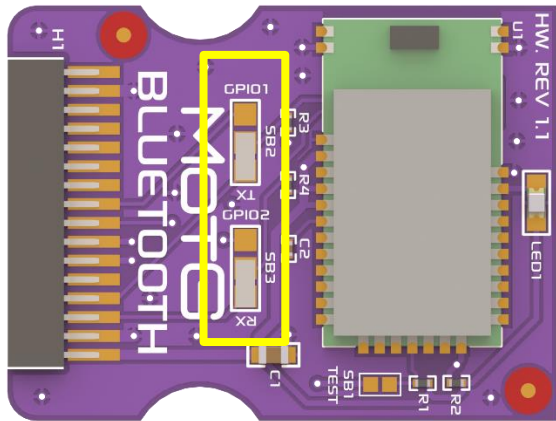
```

The serial data coming from the MOTG-BLUETOOTH could be a command or the actual data from the paired device. This could be detected by checking the received characters from the MOTG-BLUETOOTH which is enclosed by the % symbol. In this example, the status is directly printed on the screen as follows while the actual data is sent directly back to the paired device.

# The Hardware Connection

## Configure the MOTG-BLUETOOTH

The two UART pins (RX and TX) of the MOTG-BLUETOOTH is not connected to the UMI of the MOTG by default. In order to use the RX and TX pins of the module, a solder blob is added on the pads of SB2 and SB3 as shown below.



The user also has the option of using the GPIO1 and GPIO2 as the UART pin (RX and TX). For Picaso based displays, use the TX and RX pins. In this example, the TX and RX pins are used for the Bluetooth chip to connect to UART.

In this application note, a gen4-MOTG-AC1 adaptor board is used to interface MOTG-BLUETOOTH to the Diablo16 based gen4-display modules.

Alternative option is to use the gen4-MOTG-AC2 board to provide two MOTG slots, for larger displays like the 4.3”, the user can opt to use the gen4-MOTG-AC4.

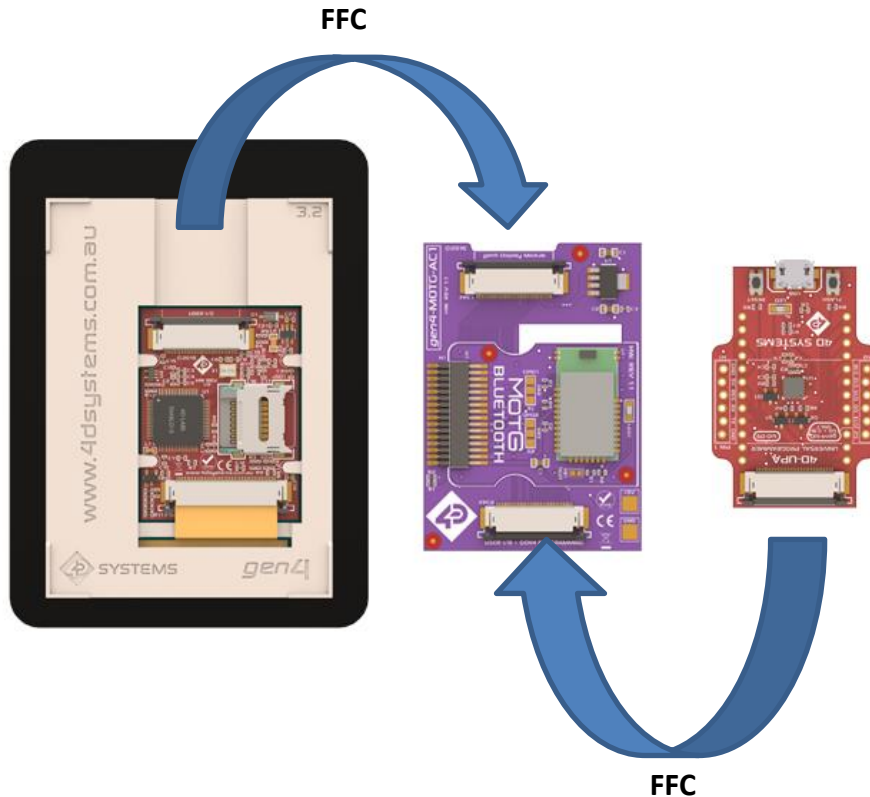
For Picaso based gen4-display modules, a gen4-MOTG-AC3 adaptor board will be required to interface with the MOTG-BLUETOOTH.

For setting up the display module reset and serial pins, the user can refer to the table below.

Variant	MOTG Solder Bridge Configuration	Pins		
		RX	TX	RST
gen4-Picaso + MOTG	TX + RX	TX1	RX1	IO3
gen4-Diablo + MOTG MOTG Adaptor Slot No. 1	TX + RX	PA3	PA2	PA4
gen4-Diablo + MOTG MOTG Adaptor Slot No. 2	TX + RX	PA3	PA2	PA7
gen4-Diablo + MOTG MOTG Adaptor Slot No. 1	GPIO1 + GPIO2	PA1	PA0	PA4
gen4-Diablo + MOTG MOTG Adaptor Slot No. 2	GPIO1 + GPIO2	PA1	PA0	PA7

## Hardware Connection

The hardware configuration is shown below.



## Build and Upload the Project

For instructions on how to build and upload a **Designer/ViSi** project to the target display, please refer to the section “**Build and Upload the Project**” of the application note

- **Designer Getting Started - First Project**
- **ViSi Getting Started – First Project for Picaso and Diablo16**

## Demonstration

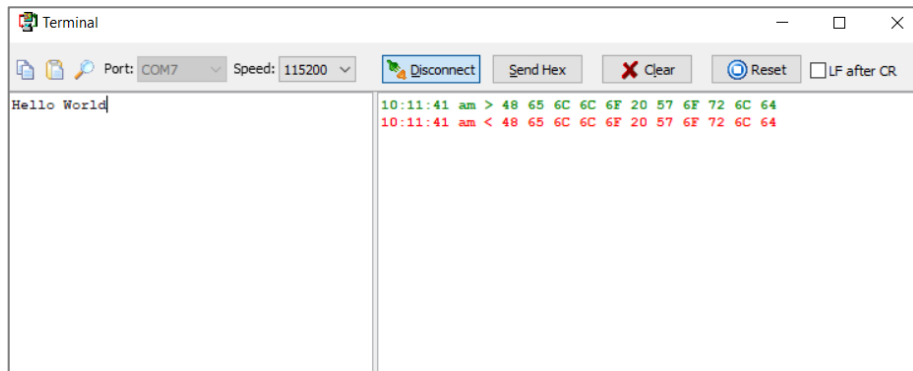
The following image shows what is printed on the screen once the MOTG-BLUETOOTH is initialized.

A screenshot of a screen displaying the text 'MOTG-BLUETOOTH ECHO' in green on a black background.

The MOTG-BLUETOOTH can then be paired with a BLE capable device. Once paired, the status message will be printed on the screen as shown below.

A screenshot of a screen displaying the text 'MOTG-BLUETOOTH ECHO' in green on a black background. Below the title, the following status messages are printed in green: '%CONNECT, 1, 4AB3BB895FF%', '%SECURED%', '%STREAM\_OPEN%', and '%CONN\_PARAM,0012,0000,0200%'.

The paired device can then send any message and receive them back through the MOTG-BLUETOOTH.



## Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

## Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement, and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.