



Picadillo UECIDE How to Render a Workshop 4 Image Widget

DOCUMENT DATE: **23th APRIL 2019**
DOCUMENT REVISION: **1.1**



Description

This application note shows how to program the Picadillo-35T using the Universal Embedded Computing IDE (UECIDE) in Windows to make it display an image on the screen. Currently, simple graphics such as basic shapes can be rendered on the Picadillo-35T with the help of the TFT library. The TFT library has functions for drawing lines, rectangles, circles, etc. Image files stored on the uSD card can also be displayed using the BMPFile or the Raw565File libraries. There is also a library for displaying the content of a GCI file. "GCI" stands for "Graphics Composer Image". It is a custom image format used by the Workshop 4 IDE for storing images to be used on 4D display modules. To program the Picadillo-35T to render the contents of a GCI file, we make use of the gciWidget library. Below is an example output.



To achieve the output above, we first make use of the Workshop 4 IDE to generate the GCI file which will contain the 4D logo image. The GCI file is

then copied to the uSD card, which is then mounted to the Picadillo-35T. The next step is to write the UECIDE sketch for making the Picadillo-35T access and render the content of the GCI file.

Before getting started, the following are required:

- The [Picadillo-35T](#)
- The [UECIDE \(Universal Embedded Computing IDE\)](#)
- The [Workshop 4 IDE](#)
- A [uSD card](#)
- A USB-to-mini-USB cable
- 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 discussed in these recommended application notes.
- This application note requires that the reader has a basic knowledge of any programming language such as C and C++.

Content

Description	2
Content	3
Application Overview	4
Setup Procedure	4
<i>The UECIDE Software</i>	4
Install the gciWidget Library	4
<i>The Workshop 4 IDE Software</i>	5
<i>Attached Files</i>	5
The UECIDE Sketch	5
The ViSi-Genie Project	6
Generate the Widgets	6
<i>Launch Workshop 4</i>	6
<i>Create a New Project</i>	6
<i>Select ViSi Genie</i>	8
<i>Add an Image Widget</i>	9
<i>Naming of Widgets</i>	11
<i>Save the Project</i>	11
<i>Insert the μSD Card to the PC</i>	11
<i>Compile and Upload</i>	12
<i>The GCI and DAT Files</i>	13
The DAT File	14
The GCI File	15

The Display-an-Image Program	16
<i>The gciWidget Library</i>	16
Process Diagram	17
Definition of Terms	17
<i>gciWidgetSet Constructor</i>	18
<i>Pointer to a gciWidget Object</i>	18
<i>uSD Card Check Routine</i>	18
<i>Initialize the Widgets</i>	18
<i>“Fetch” the Widgets</i>	18
<i>Group the Widgets by Page</i>	19
<i>Set the Current Page</i>	19
<i>Render the Widgets on the Current Page</i>	19
Compile and Upload	19
<i>Insert the uSD Card</i>	19
<i>Run the Project</i>	20
Proprietary Information	21
Disclaimer of Warranties & Limitation of Liability	21

Application Overview

The objective of this application note is to present to the user the basic procedure for creating widgets in the Workshop 4 IDE and the procedure for programming the Picadillo-35T to display the widgets. After reading this application note, the user shall be able to display an image widget on the Picadillo-35T.

Setup Procedure

The UECIDE Software

Information about the UECIDE can be found here: <http://uecide.org/>. For more information on the following topics, refer to the application note [Picadillo UECIDE Hello World](#).

- How to Download and Install the UECIDE Software
- How to Install the Picadillo-35T Board
- How to Install the ChipKIT Core and the pic32-tools Compiler
- How to Select the Picadillo-35T Board
- How to Select the chipKIT Core
- How to Set the Compiler
- How to Install the TFT Library
- How to Download and install the Picadillo-35T USB Driver

Install the gciWidget Library

The gciWidget library will be used to support the widgets generated in the Workshop 4 IDE. The gciWidget library can be downloaded from [GitHub](#). The procedure described in the application note [Picadillo UECIDE Hello World](#) for installing the TFT library can be used as a guide for installing libraries.

The Workshop 4 IDE Software

Information about the Workshop 4 IDE can be found on its [product page](#). The software can be downloaded from the same page. The [downloads section](#) lists several files that may help the user. For this application note, the following documents are relevant.

- Workshop-4-IDE Installation Guide
- ViSi-Genie User Guide
- ViSi User Guide
- Workshop-4-IDE User Guide

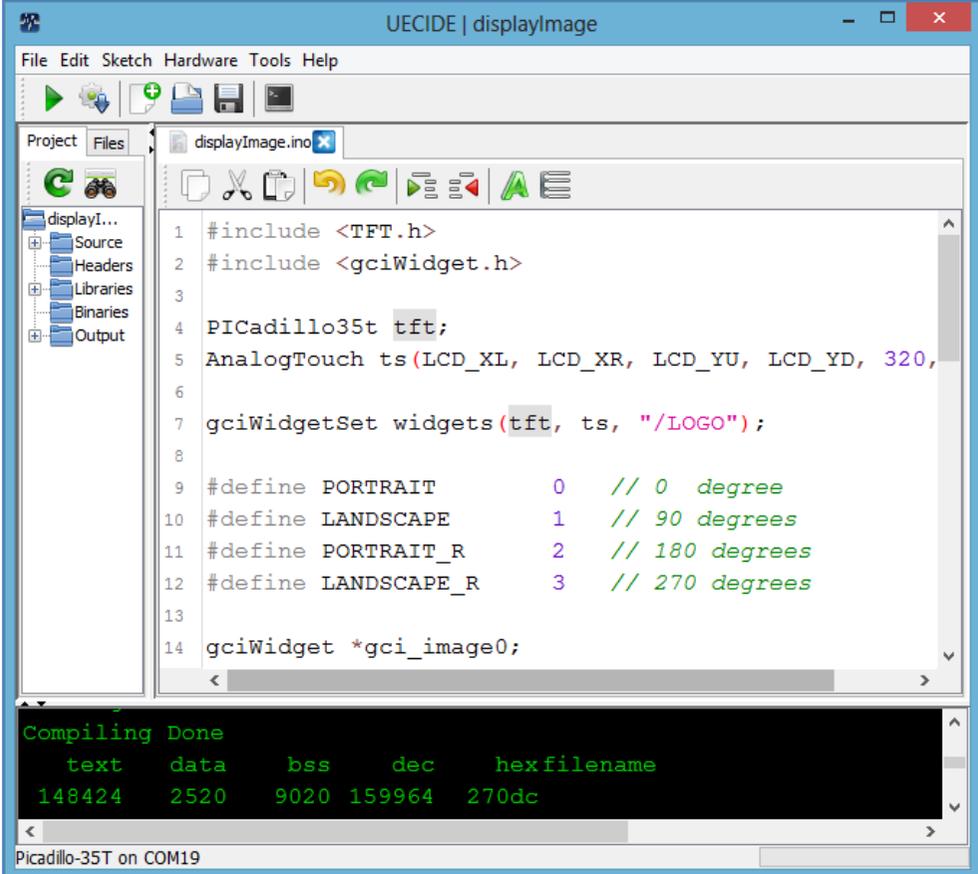
Attached Files

The UECIDE Sketch

This document comes with a demo UECIDE sketch in a zip file.

 displayImage.zip	Compressed (zipped) Folder	1 KB	12
--	----------------------------	------	----

In the file explorer window, extract the content of the zip file to a desired location then open the sketch in the UECIDE.



```
1 #include <TFT.h>
2 #include <gciWidget.h>
3
4 PICadillo35t tft;
5 AnalogTouch ts(LCD_XL, LCD_XR, LCD_YU, LCD_YD, 320,
6
7 gciWidgetSet widgets(tft, ts, "/LOGO");
8
9 #define PORTRAIT      0 // 0 degree
10 #define LANDSCAPE    1 // 90 degrees
11 #define PORTRAIT_R   2 // 180 degrees
12 #define LANDSCAPE_R  3 // 270 degrees
13
14 gciWidget *gci_image0;
```

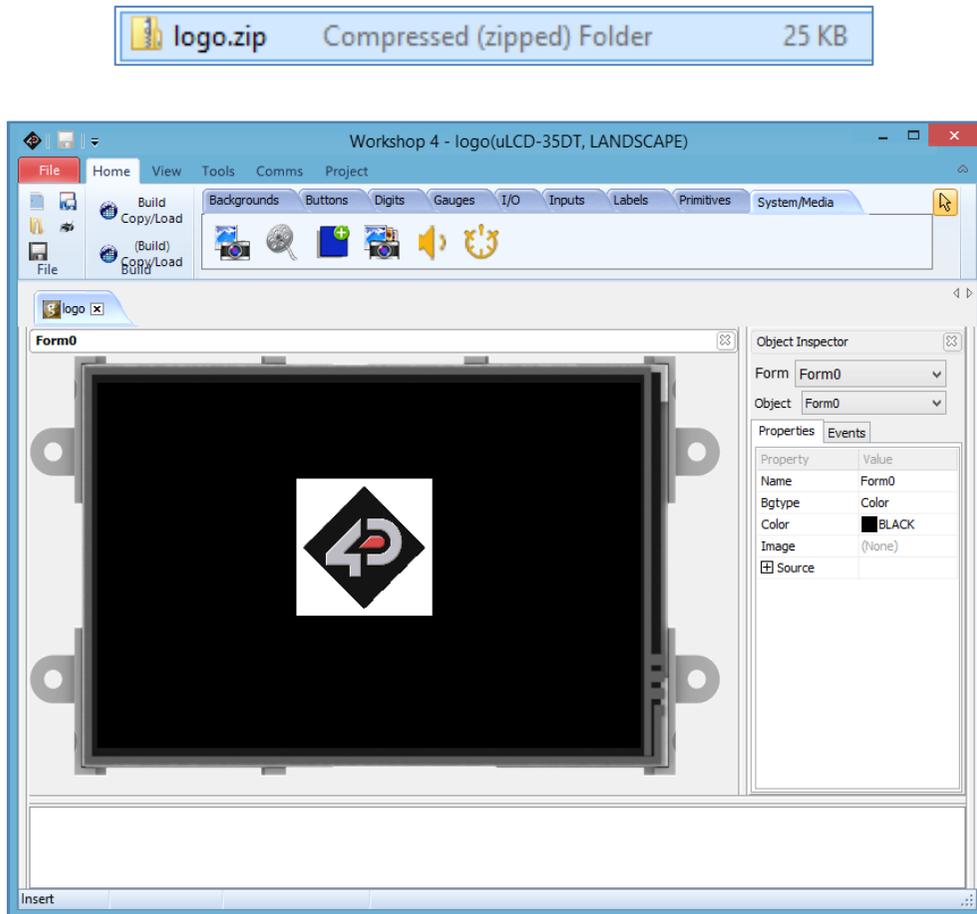
Compiling Done

text	data	bss	dec	hexfilename
148424	2520	9020	159964	270dc

Picadillo-35T on COM19

The ViSi-Genie Project

The Workshop ViSi-Genie project is also contained in a zip file. Double-clicking on the ViSi-Genie project file will open it in the Workshop 4 IDE.



The steps involved in creating this project will now be shown.

Generate the Widgets

Launch Workshop 4

There is a shortcut for Workshop 4 on the desktop.



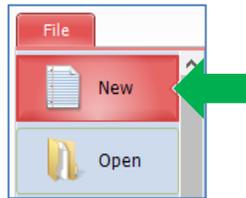
Launch Workshop 4 by double-clicking on the icon.

Create a New Project

Workshop 4 opens and displays the **Recent** page.



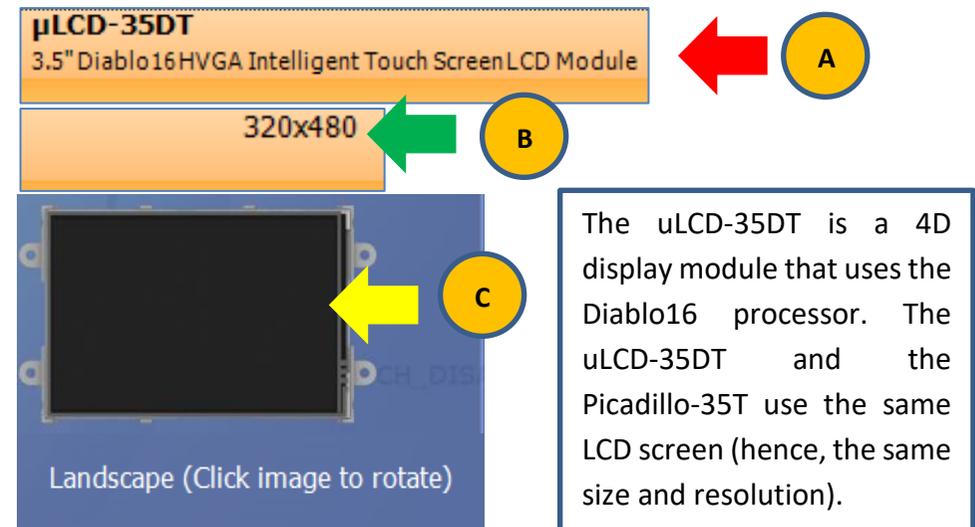
To create a new project, there are two options.
Click on the top left-most icon, New.



Or Click on the icon beside Create a new Project.



The Choose-Your-Product window appears. Select the uLCD-35DT and the landscape orientation. When done, click on the next button.

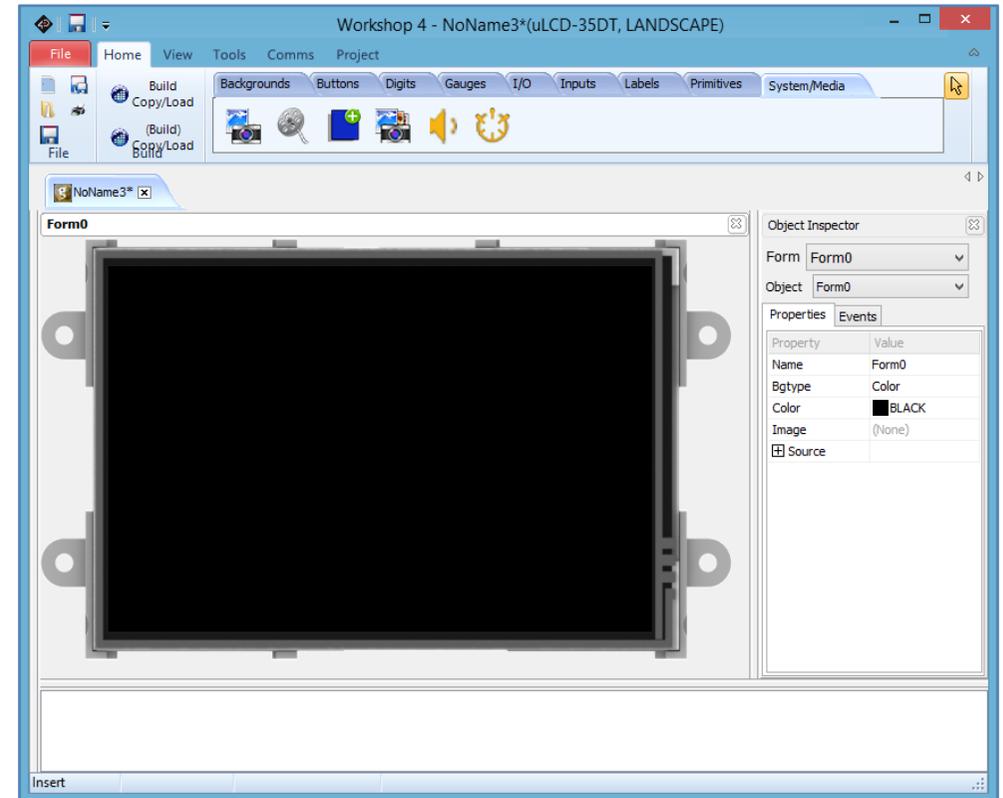


The uLCD-35DT is a 4D display module that uses the Diablo16 processor. The uLCD-35DT and the Picadillo-35T use the same LCD screen (hence, the same size and resolution).

Select ViSi Genie

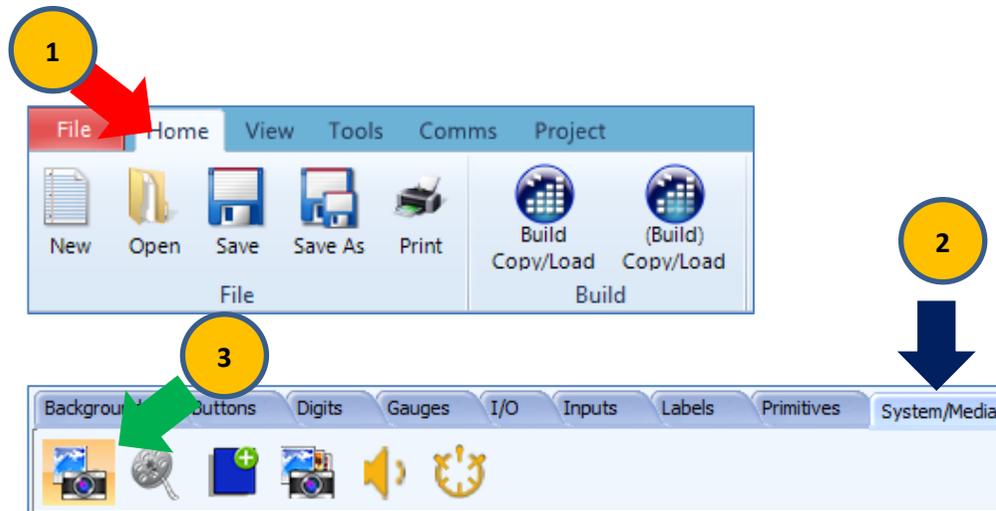


Everything is now ready to start designing the project. **Workshop 4** displays an empty screen, called **Form0**. A **form** is like a page on the screen. The form can contain **widgets** or **objects**, like trackbars, sliders, displays or keyboards. Below is an empty form.

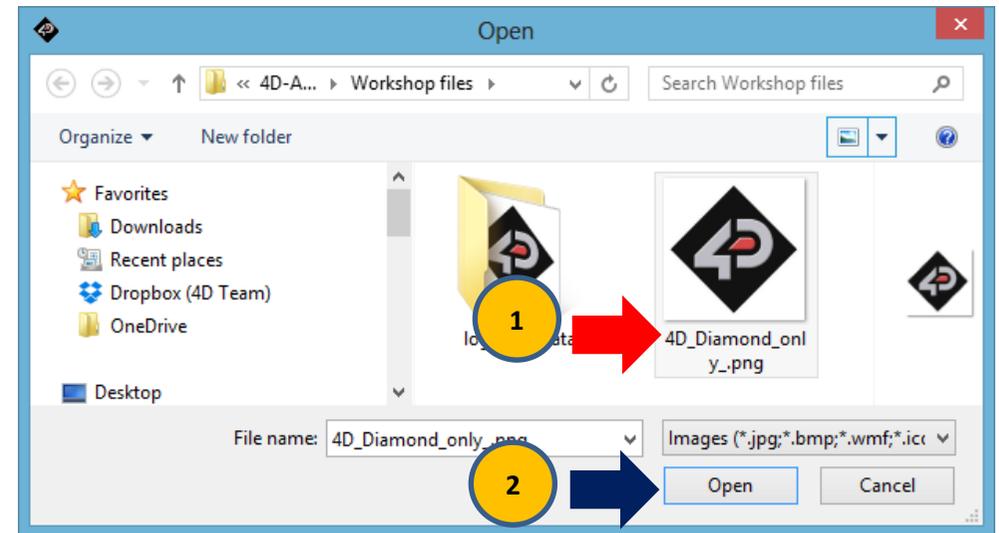


Add an Image Widget

To add an image widget to the **WYSIWYG** (What-You-See-Is-What-You-Get) screen, go to the **home** menu, select the **System/Media** pane, and click on the image widget icon.



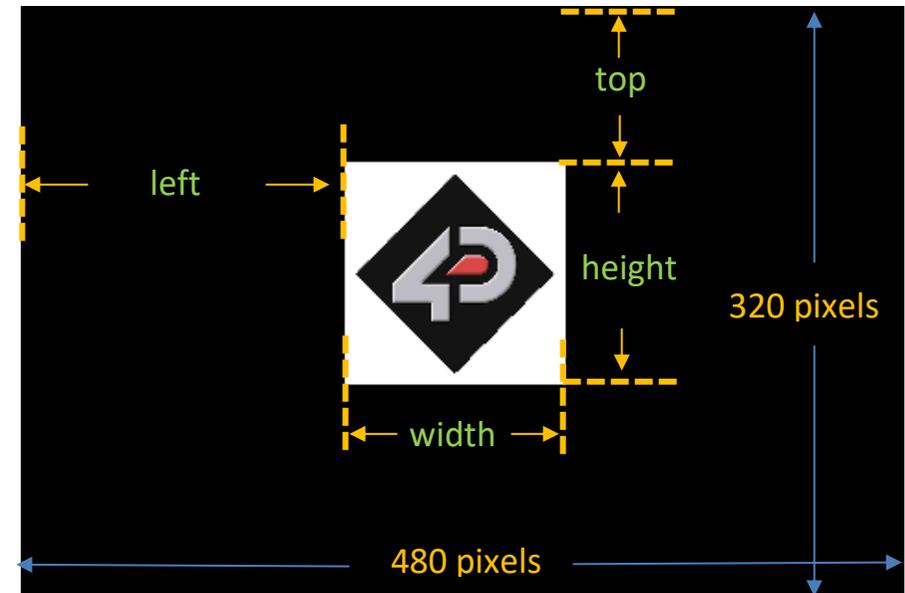
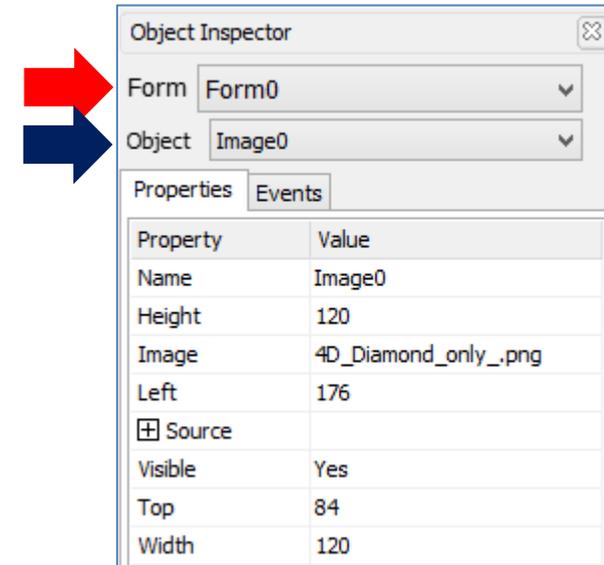
Click on the **WYSIWYG** screen to place the image widget. The WYSIWYG screen simulates the actual appearance of the display module screen. A standard open window appears and asks for an image file.



Here an image file for the 4D Systems logo is chosen. Upon clicking the Open button, the image now appears on the WYSIWYG screen.

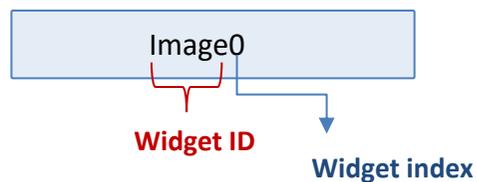


The widget can be dragged to any desired location and resized to the desired dimensions. The **Object Inspector** on the right part of the screen displays all the properties of the newly created image widget or object named **Image0**. The widget is on the first form of the project. The first form is **Form0**. A form can contain one or more widgets. This project, however, will only consist of one form with one widget. The use of multiple forms will be discussed in a separate application note.



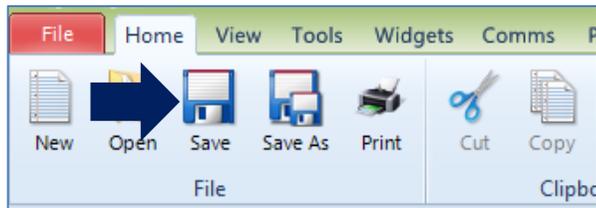
Naming of Widgets

Naming is important to differentiate between widgets or objects of the same kind. For instance, suppose the user adds another image widget to the WYSIWYG screen. This widget will be given the name Image1 – it being the second image widget in the project. The third image will be given the name Image2, and so on. A widget's name therefore identifies its kind and its unique index number. It has an ID (or type) and an index.



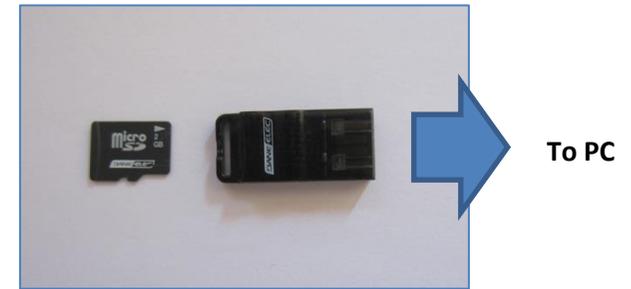
Save the Project

Save the program with the desired file name first.

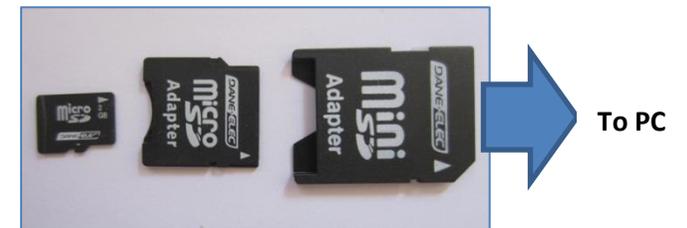


Insert the μ SD Card to the PC

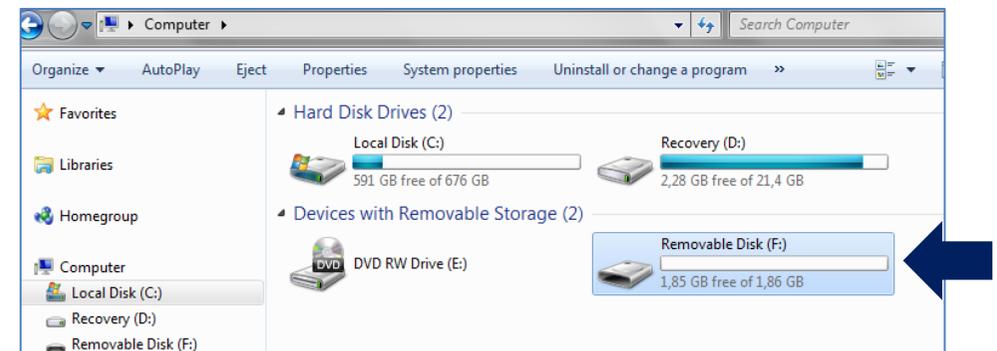
The Workshop 4 IDE will copy files to FAT16- or FAT- formatted μ SD cards only. Format the μ SD card accordingly first before proceeding. The Picadillo-35T, however, supports both the FAT16 and FAT32 file formats. Insert the μ SD card into the USB adaptor and plug the USB adaptor into a USB port of the PC.



OR insert the μ SD card into a μ SD to SD card converter and plug the SD card converter into the SD card slot of the PC.

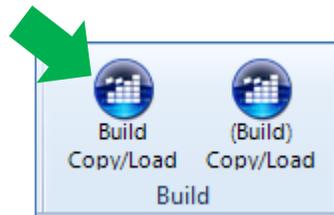


Check if the μ SD card is mounted, here it is mounted as drive E:

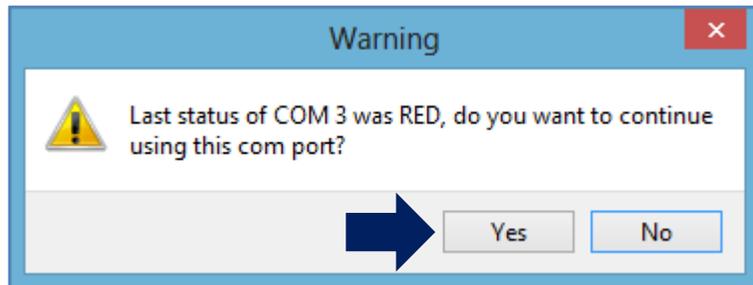


Compile and Upload

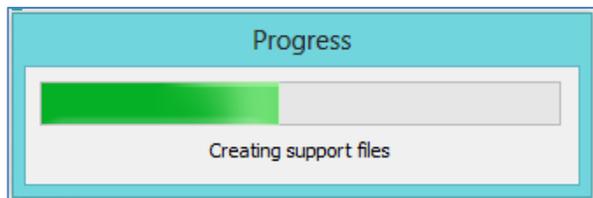
Go to the **Home** menu and click on the **Build Copy/Load** button. Clicking on the left icon always builds and copies the graphics files to the uSD card and downloads the program to the display module. Use the left icon to be sure that the graphics files and program are always up-to-date, i.e., they include the latest changes made.



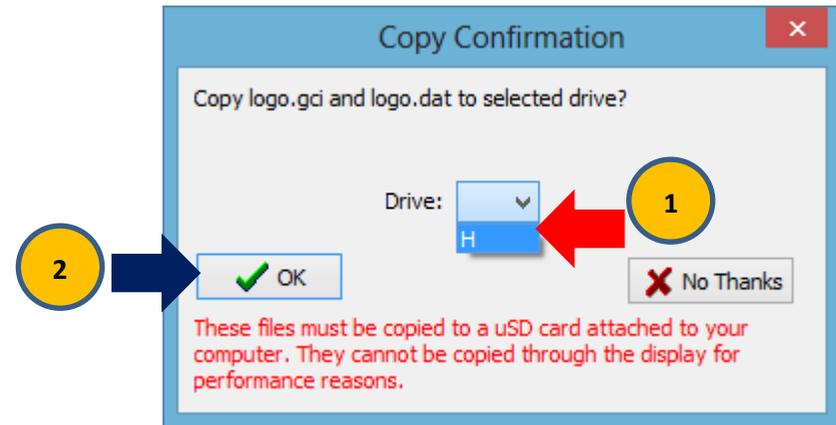
A warning may appear at this point. Workshop 4 is looking for the target display. Ignore the warning by clicking on the yes button.



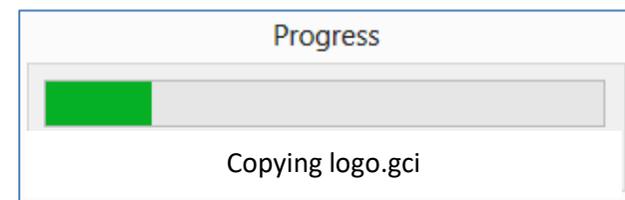
Workshop now generates the graphics files.



Workshop will prompt the user for the μ SD card to which the graphics files will be copied. Select the correct drive in the drop-down list then click on OK.

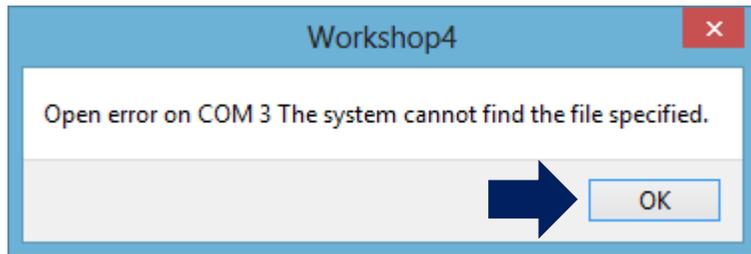


A progress bar is displayed while the files are being copied to the μ SD card. Workshop copies two files to the μ SD card – the GCI and the DAT files. The GCI file contains the graphics and the DAT file contains a list of the widgets or objects inside the GCI file. These files will be accessed by the UECIDE Display-an-Image program when it runs on the Picadillo-35T.



After copying the graphics files to the uSD card, Workshop will now attempt to upload the Genie program to the flash/RAM memory of the display

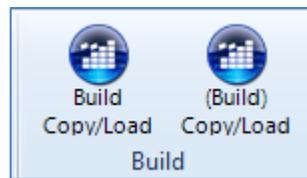
module. However, since we are interested with the uSD card files only and since we don't have a uLCD-35DT, the process can now be stopped.



The uSD card shall now contain the DAT and GCI files.

Name	Date modified	Type	Size
 logo.dat	12/3/2014 7:38 PM	DAT File	1 KB
 logo.gci	12/3/2014 7:38 PM	GCI File	29 KB

Note: There are two Build-Copy/Load buttons.

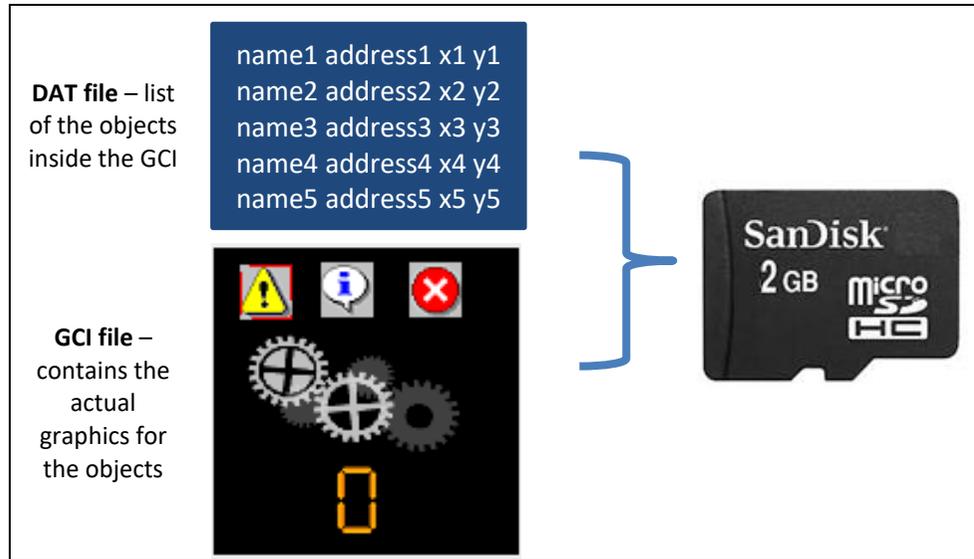


The left button forces a build of the graphics files all the time. The right button builds the graphics files only when Workshop detects any change made on the WYSIWYG screen and/or the Object Inspector. When no changes are detected, clicking on the right button will simply cause Workshop to copy the graphics files to the uSD card and upload the program

to the display. The right button is useful for loading a single Genie application to multiple displays and uSD cards. The left button is the better choice when the user wants to make sure that the graphics files are updated all the time. Note that for larger Genie projects, Workshop may take some time to build the graphics files.

The GCI and DAT Files

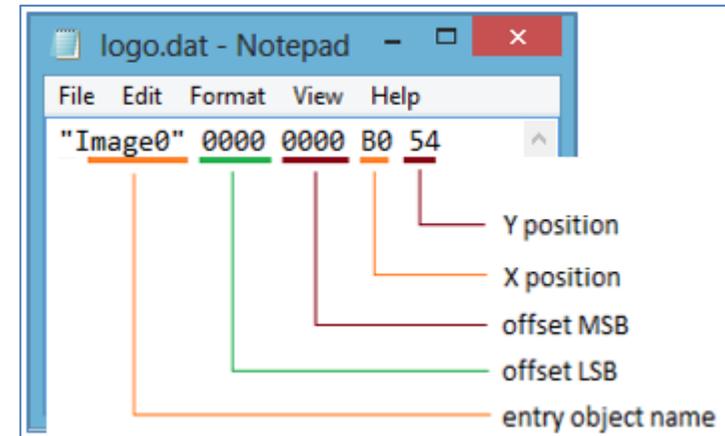
Images, animations, and movie clips copied to the uSD card must be of a format readable by the target processor. These files therefore are not saved on to the memory card in their original formats (common file formats are jpeg, gif, png, mov, flv, mpeg, etc) but are converted and combined by the Workshop 4 IDE into a single graphics file, which is called the "GCI" file. GCI stands for Graphics Composer Image. The Graphics Composer or GC is a program used by Workshop to convert multimedia graphics files into a GCI file. The images or objects inside the GCI file are arranged in a sequential order – that is according to when the user adds them. A list of these objects is created by the Graphics Composer along with the GCI file. This list, which is actually another file – the "DAT" file, contains the names of the object entries, their locations inside the GCI file, and their initial X/Y positions on the screen. To illustrate:



The project created earlier is composed of only one image object, which is the 4D logo.

The DAT File

The DAT file for the 4D logo project, when opened in Notepad, contains the line shown below.



Information	Description
Entry object name	Name of the widget
Offset MSB	Offset of the location of the image in the GCI file, MSB (hexadecimal)
Offset LSB	Offset of the location of the image in the GCI file, LSB (hexadecimal)
X position	X coordinate or the left property in the object inspector (hexadecimal)
Y position	Y coordinate or the top property in the object inspector (hexadecimal)

When the user creates a sketch in the UECIDE using the gciWidget library, the entry object name will be used as a reference to a widget. The gciWidget library will parse the DAT file to extract all of the information above.

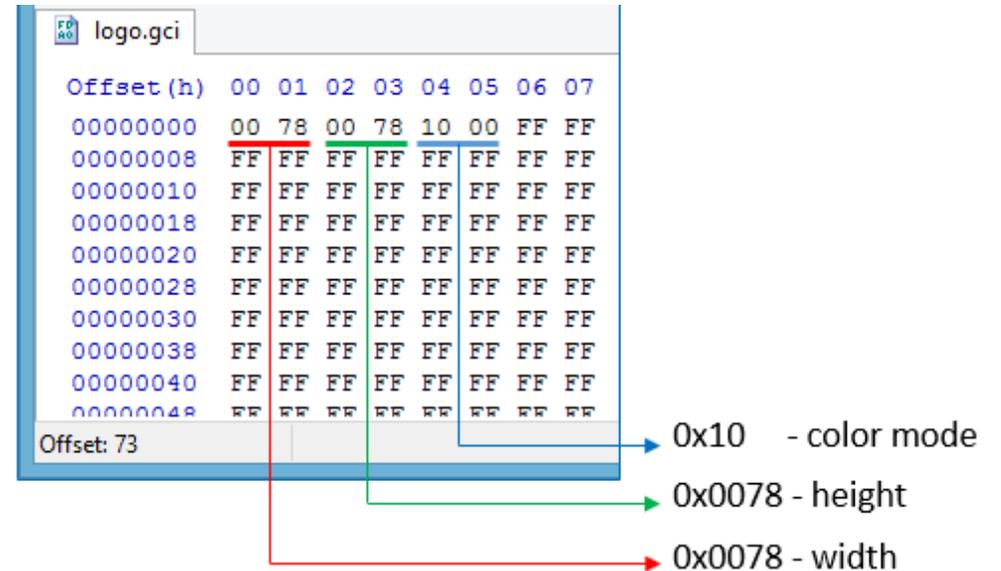
The GCI File

The GCI file for the 4D logo project has a single entry object which is for the 4D logo image. This entry object contains the hexadecimal bytes for each pixel of the 4D logo image (16-bit 565 RGB colour format). For projects with two image widgets, the object entry for the second widget will follow the object entry of the first. Thus, the GCI file may contain several entry objects, depending on the number of widgets in the project.

The entry object for an image widget has a six-byte or a three-word header. The header bytes are then followed by the pixel data (from pixel 1 to pixel N, where N is the total number of pixels in the image).

Header	
Word 1	Horizontal size of the image
Word 2	Vertical size of the image
Word 3	Colour mode (MSB only)

The default colour mode is 65k colour mode or 16-bit 565 RGB format, where each pixel is represented by two bytes.



Shown above are the header bytes of the entry object for the 4D logo image. The rest are the pixel data. The gciWidget library will extract and use the information contained in the header bytes to display the pixel data.

The Display-an-Image Program

Many aspects of the sketch for the Display-an-Image program were already discussed in the application note [Picadillo UEcide Hello World](#). For more information on the topics below, refer to the Hello World application note.

Picadillo UEcide GCI How to Display an Image (this application note) will focus more on the use of the `gciWidget` library.

- The TFT Library
- Backlight Control
- Initialize the Display
- Set the Orientation of the Display
- Initialize the Touch Screen
- Set the Orientation of the Touch Screen
- Set the Touch Screen Area X and Y Scale Parameters
- Fill the Screen with a Colour
- Set the Text Foreground Colour
- Set the Text Font
- Scale the Text Font
- Print a String

The `gciWidget` Library

The `gciWidget` library supports the widgets generated in Workshop.

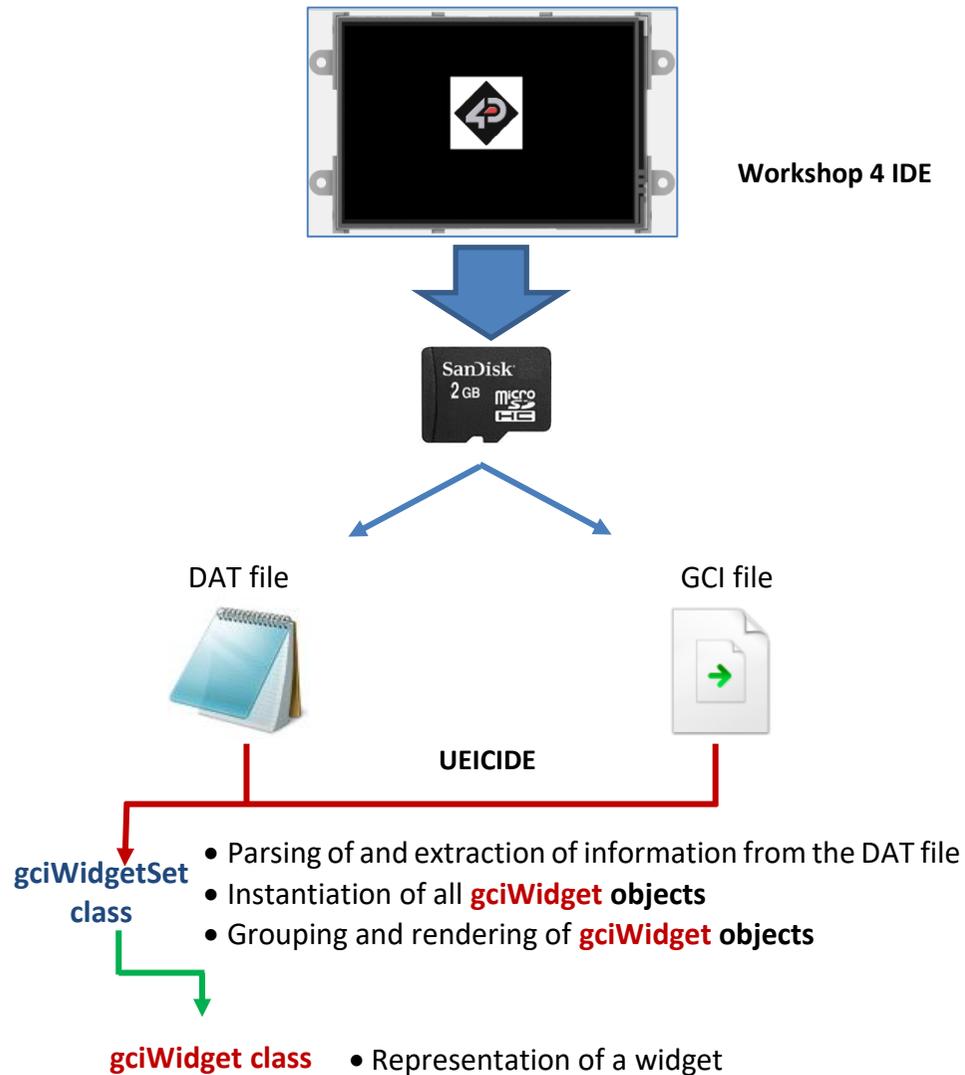
```
2 #include <gciWidget.h>
```

The `gciWidget` library has two classes – the `gciWidget` and the `gciWidgetSet`. To facilitate this discussion, the `gciWidget` class can be described as the class used to represent individual widgets. For each widget, a `gciWidget` object will be instantiated.

The `gciWidgetSet` class, on the other hand, is used to manage all `gciWidget` objects. This class handles the extraction of widget information from the DAT file, the instantiation of all `gciWidget` objects, the grouping and rendering of `gciWidget` objects by page, etc.

Process Diagram

The diagram below attempts to illustrate what the library does in general.



Definition of Terms

To avoid confusion, the terms “**widget**” and “**gciWidget object**” will be differentiated. The term “**widget**” shall refer to the object in the Workshop 4 IDE. A widget can be an image, a slider, a button, etc.



The term “**gciWidget object**” shall refer to the representation of a widget when programming in the UEICIDE.

UEICIDE – “gciWidget object”

```
14 gciWidget *gci_image0;
```

```
37 gci_image0 =
```

```
widgets.getWidgetByName("Image0");
```

More often than not, these two terms have the same general meaning. For example:

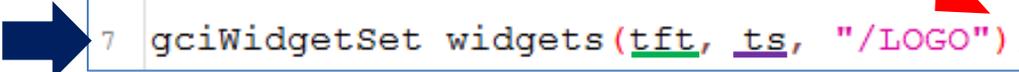
The method **gciWidgetSet::render()** renders the **widgets** on the current page.

The method `gciWidgetSet::render()` renders the **gciWidget** objects for the current page.

gciWidgetSet Constructor

The third argument of the constructor for the `gciWidgetSet` class is the filename of the GCI and DAT files. Here the object `widgets` is instantiated.

```
4 PICadillo35t tft;
5 AnalogTouch ts(LCD_XL, LCD_XR, LCD_YU,
6
7 gciWidgetSet widgets(tft, ts, "/LOGO");
```



Pointer to a gciWidget Object

```
14 gciWidget *gci_image0;
```

A **gciWidget** object for the image widget will be instantiated by `widgets`. Declare a pointer for this.

uSD Card Check Routine

```
32 if (!SD.begin(PIN_SD_SS)) {
33     tft.print("Error opening SD card");
34 }
```

Initialize the Widgets

```
36 widgets.init();
```

This method opens the DAT file and scans it line-by-line. A **gciWidget** object is instantiated for each line. Widget information from each line is also extracted. Remember that each line in the DAT file contains the name of the widget, its x and y coordinates, and the offset of its image in the GCI file. This method also creates a list of all the instantiated **gciWidget** objects. For this project, only one **gciWidget** object will be instantiated since there is only one widget.

“Fetch” the Widgets

```
37 gci_image0 =
    widgets.getWidgetByName("Image0");
```

The instantiation process of **gciWidget** objects is internal to the library. To be able to access a certain **gciWidget** object, use the `gciWidgetSet` class method shown above. This function scans the list of **gciWidget** objects previously created by `widgets.init()` and returns a pointer to the **gciWidget** object with the name “Image0”. If there are other widgets in GCI file of the uSD card, they can be “fetched” using this method as long as their names are known. Remember that widget names are generated in the Workshop 4 IDE.

Group the Widgets by Page

```
38 widgets.setPage(gci_image0, 0);
```

In the Workshop 4 IDE it is possible to create a project with multiple forms. Each form may contain one or more widgets. The GCI and DAT files on the uSD card do not contain information on how the widgets are grouped by forms in the Workshop 4 IDE. The user will have to code this manually in the UECIDE using the method shown above. The gciWidget library refers to “forms” as “pages”.

Set the Current Page

To set the current page number, use the method below.

```
39 widgets.selectPage(0);
```

Render the Widgets on the Current Page

To render the widgets on the current page, write

```
44 widgets.render();
```

Note that this method is now a part of the main loop. The method

```
43 ts.sample();
```

is for touch detection. This line is not relevant for now.

Note: `gciWidgetSet::setPage()` and other methods of the library need to be called for each object. Manually coding these lines can become tedious especially for large projects with dozens of widgets. The solution is to use the “Generated-Files” option in Workshop to generate these lines automatically. This will be discussed in a separate application note.

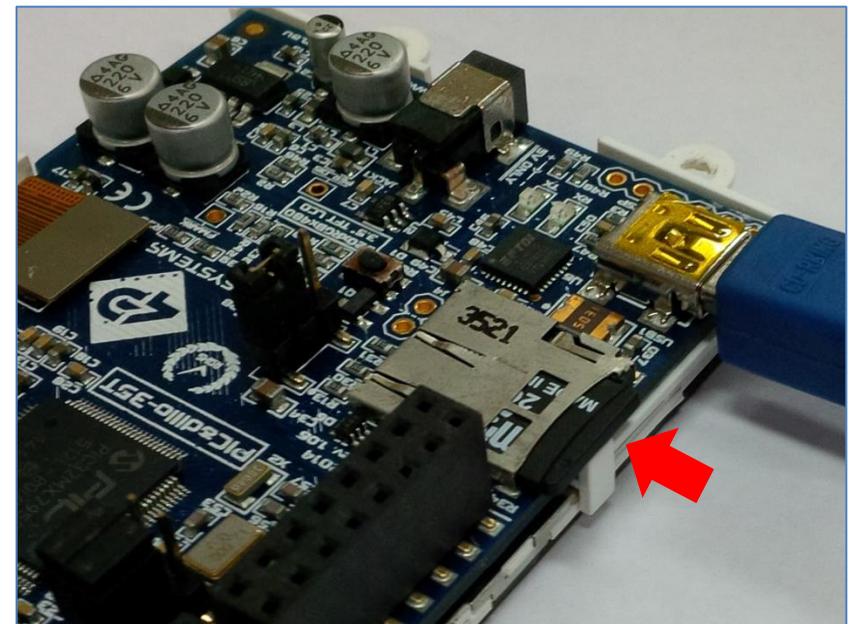
Compile and Upload

Refer to the application note [Picadillo UECIDE Hello World](#) for more information on the following topics:

- How to Set the Serial Port
- How to Upload the Program

Insert the uSD Card

After uploading the program to the Picadillo-35T, insert the uSD card to the uSD card slot.



Run the Project

The program should now run on the Picadillo-35T.



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.