RP2350 Development

Manual

Revision 1.0

Contents

1. Description 4

2. Development Setup 4

3. Application Overview 4

4. Setup Procedure 4

5. Development Roadmap 5

5.1. Launch Workshop5 5

5.2. Create a New Project 6

5.2.1. Display Selection 7

5.3. The Main Screen 10

5.3.1. Area 1: Menus 11

5.3.2. Area 2: Ribbons with Icons 11

5.3.3. Area 3: Code Editor 11

5.3.4. Area 4: Graphics Toolbar 13

5.3.5. Area 5: Visual Editor 13

5.3.6. Area 6: Object Properties 14

5.3.7. Area 7: Message Window 15

5.4. Designing a Graphical Interface 16

5.4.1. Object Selection 16

5.4.2. Object Properties Configuration 20

5.4.2.1. Changing Between Object Properties 20

5.5. Writing the Code 21

5.5.1. Main Tab 21

5.5.2. Generated Tab 22

5.5.3. Generating Widget Code 23

5.6. Programming the Display 26

5.6.1. Connecting the Module 26

5.6.2. Compile and Upload 26

5.6.2.1. MicroSD Card 26

5.6.3. Debugging the Project 28
6. Revision History 30
7. Legal Notice 31
7.1. Proprietary Information 31
7.2. Disclaimer of Warranties & Limitations of Liabilities 31

1. Description

Workshop5 is a comprehensive software from 4D Systems providing a code and graphics editor for 4D Systems' RP2350 display modules.

It can be used to design graphical interfaces for all sorts of applications using the IDE's various widgets. All application code can also be developed within the Workshop5 IDE, easily coupling it with the design, so is a one-stop shop for development with these modules.

The Workshop5 IDE utilises Raspberry Pi's Pico SDK to handle the compiling of RP2350 projects. For more advanced users, the project can be exported for use with common code editors such as Visual Studio Code.

This manual is dedicated to explaining the setup required to run Workshop5 IDE and develop applications for applicable display modules.

2. Development Setup

This section describes how to set up Workshop5 and Pico SDK for developing applications for 4D Systems' RP2350 based display modules.

Both Workshop5 and Pico SDK should be installed in your Windows computer.

- Workshop5 IDE
- Pico Setup Installer

To install Workshop5, please refer to the Installation section of the Workshop5 User Manual.

> **Note**
>
> Workshop5 is a **Windows-only** application.

3. Application Overview

It is often difficult to design a graphical display without being able to see the immediate results of the application code. Pico SDK environment is the perfect software tool that allows the user to see the instant results of their desired graphical layout.

Additionally, there is a selection of inbuilt dials, gauges and meters that can simply be dragged and dropped onto the graphical editor. From here, each can have properties edited and at the click of a button, all relevant code is produced in the user program. Each feature of the Pico SDK environment will be outlined with examples in the next sections.

4. Setup Procedure

To install the **Workshop5 IDE**, follow the instruction on Workshop5 IDE User Manual

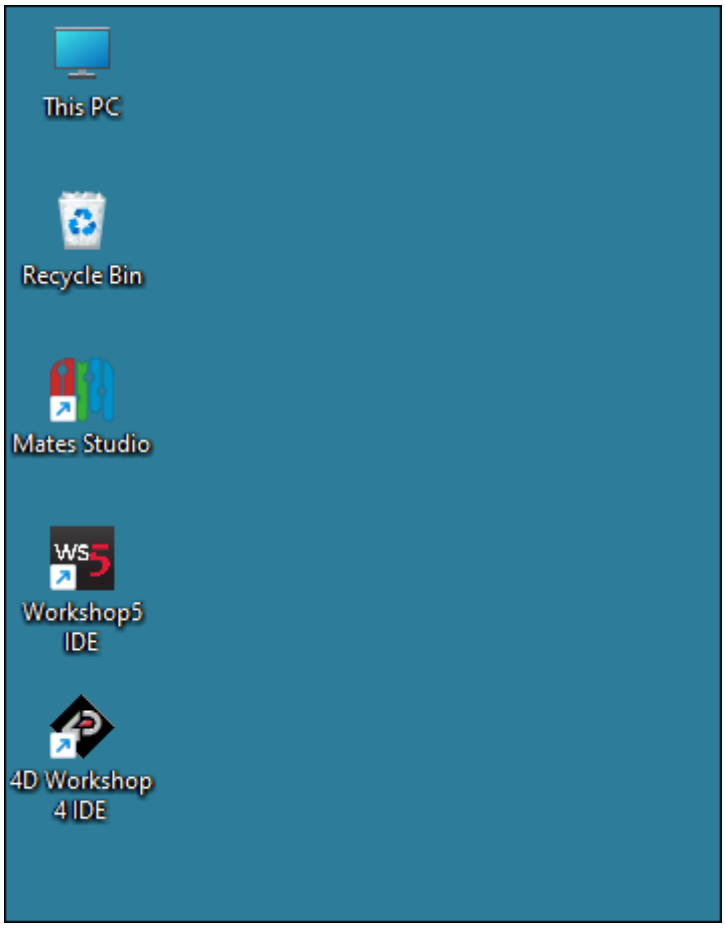
5. Development Roadmap

After having completed the setup procedure, we are now ready to create and develop a project.

This section discusses the overall development process including graphics design, writing code and uploading to the display module.

5.1. Launch Workshop5

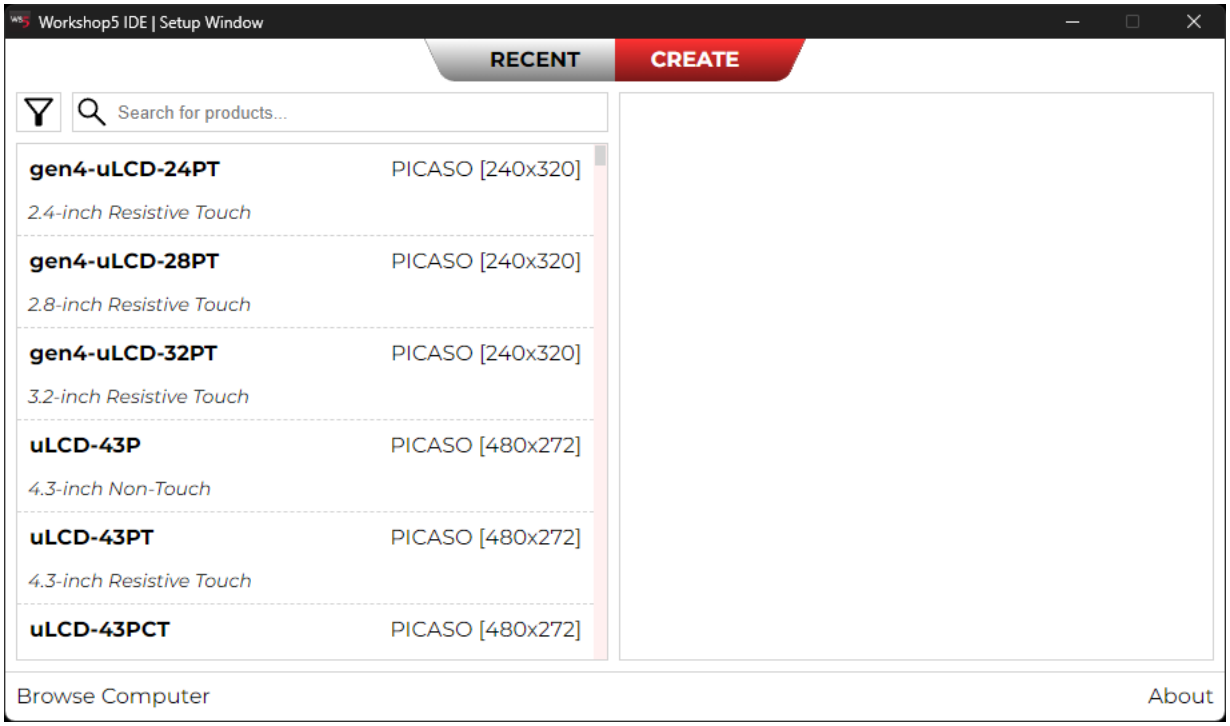
There is an alias for 4D Workshop5 IDE on the desktop.



Launch 4D Workshop5 IDE by double-clicking on the icon.

5.2. Create a New Project

At launch, Workshop5 IDE will display the **Setup Window** which defaults to **Recent** tab when there are recent projects available or to **Create** tab when there are no recent projects.

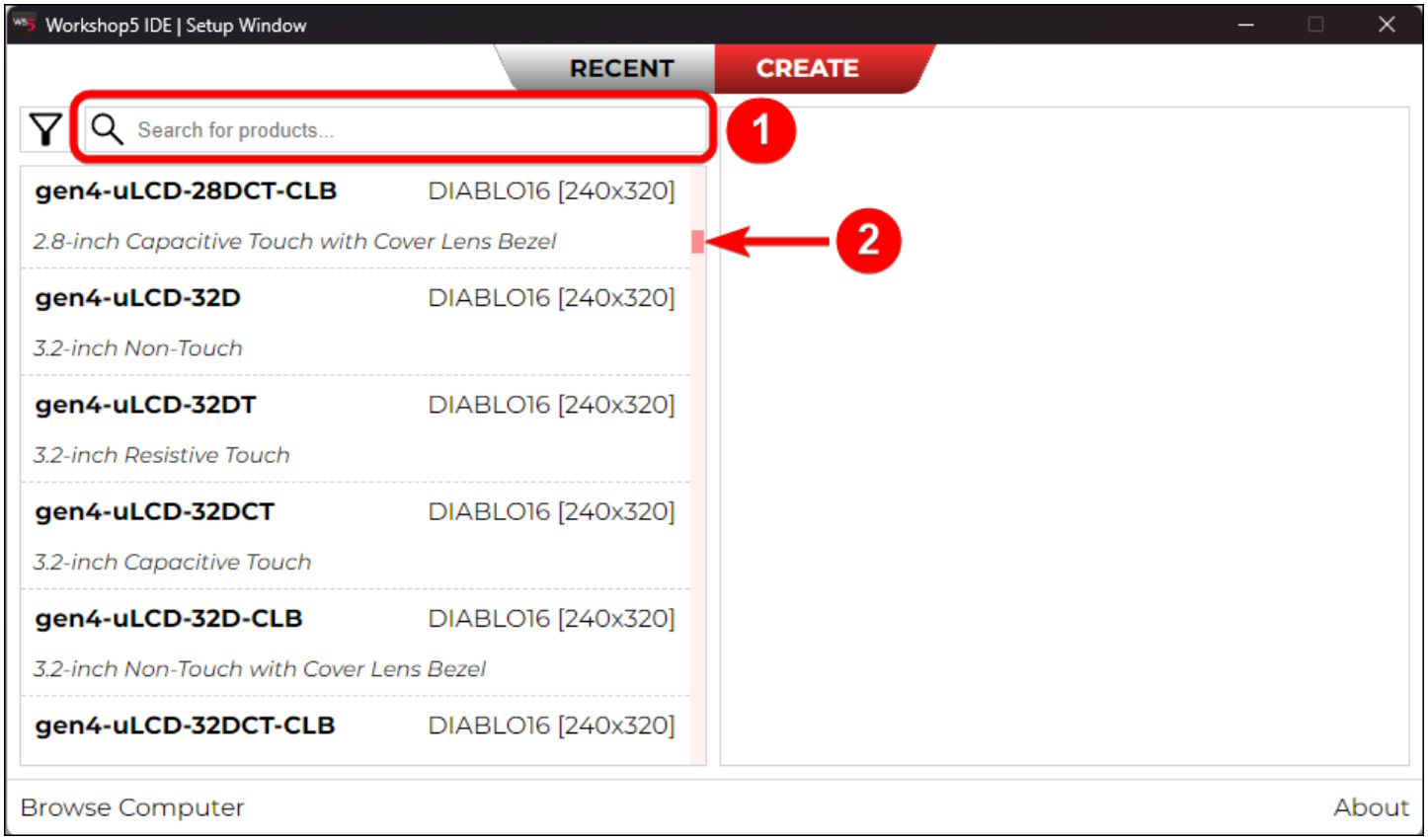


If in the Recents view, click the **CREATE** button.

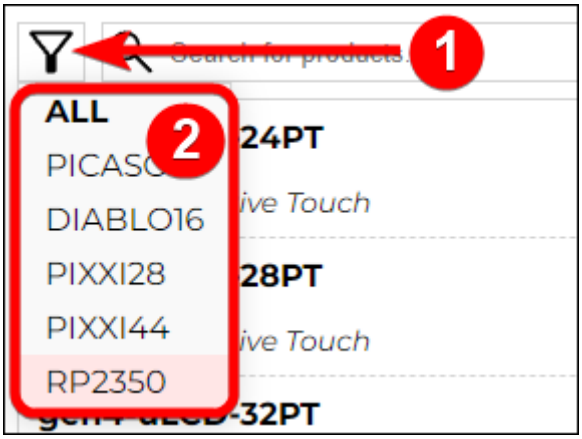
5.2.1. Display Selection

There are two (2) options that can be use to select the target display.

1. Use the Search for products text box.
2. Use the slider or the mouse wheel.

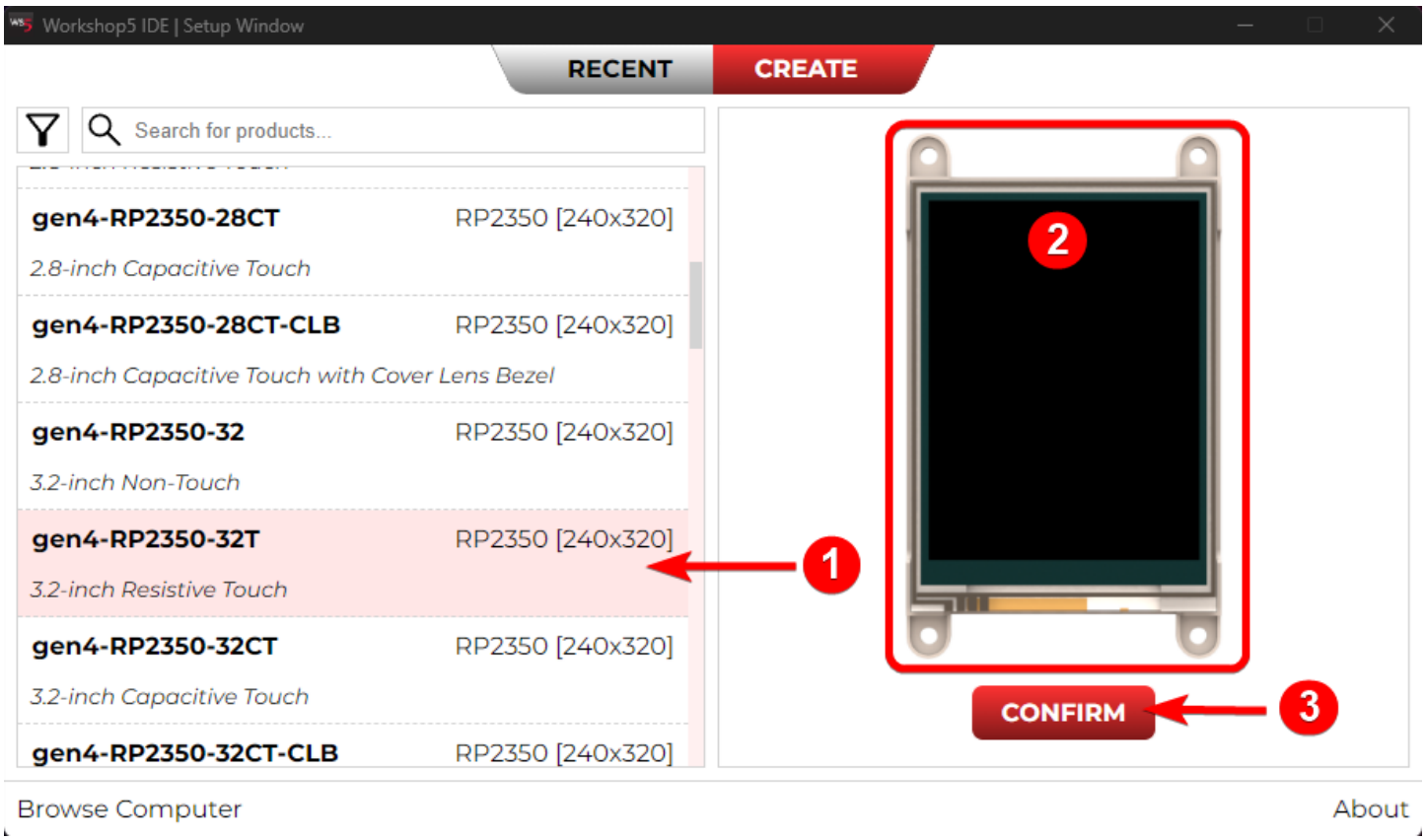


Click the filter button and select the processor of the target display.

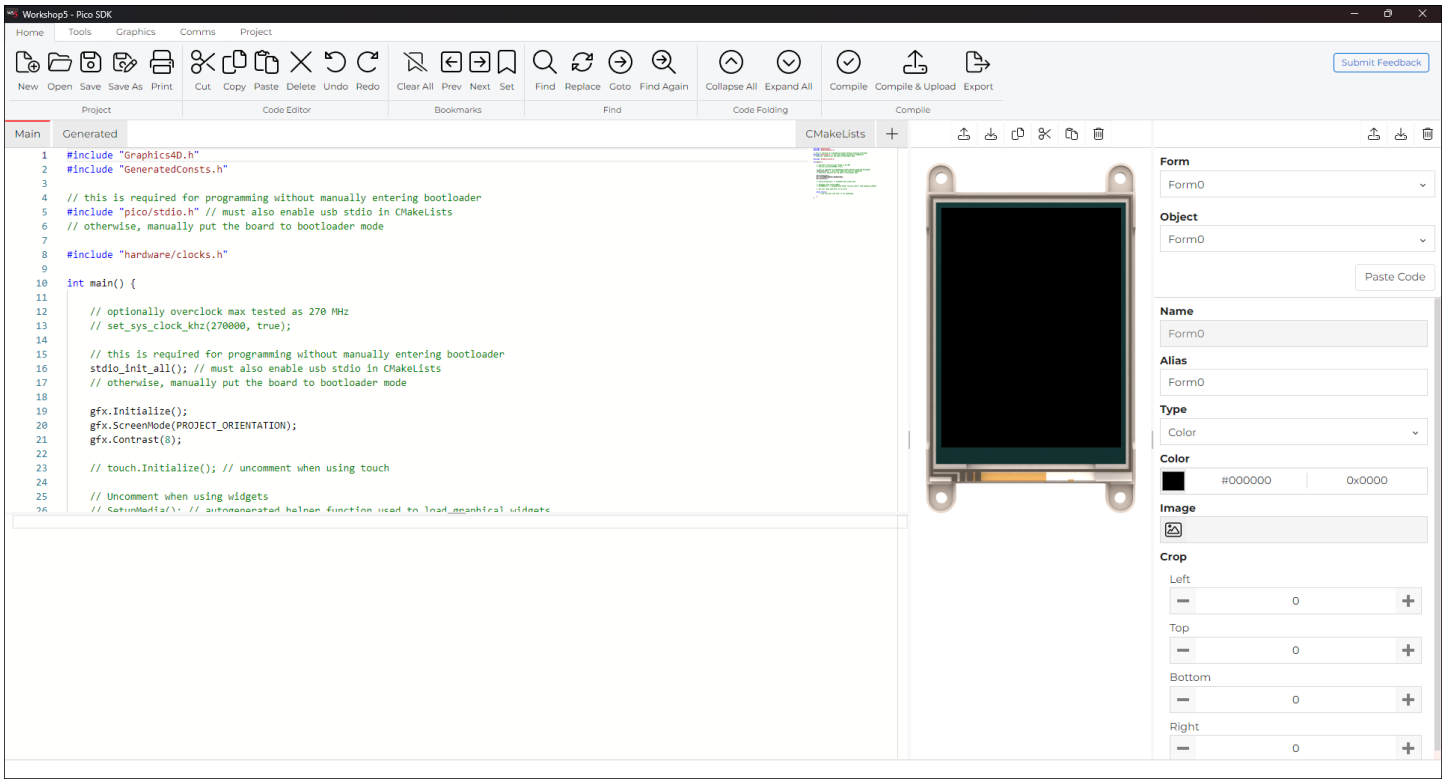


After finding the target display, follow the steps below to proceed.

1. Select target display.
2. Click the display image to select the desired display orientation.
3. Click the Confirm.

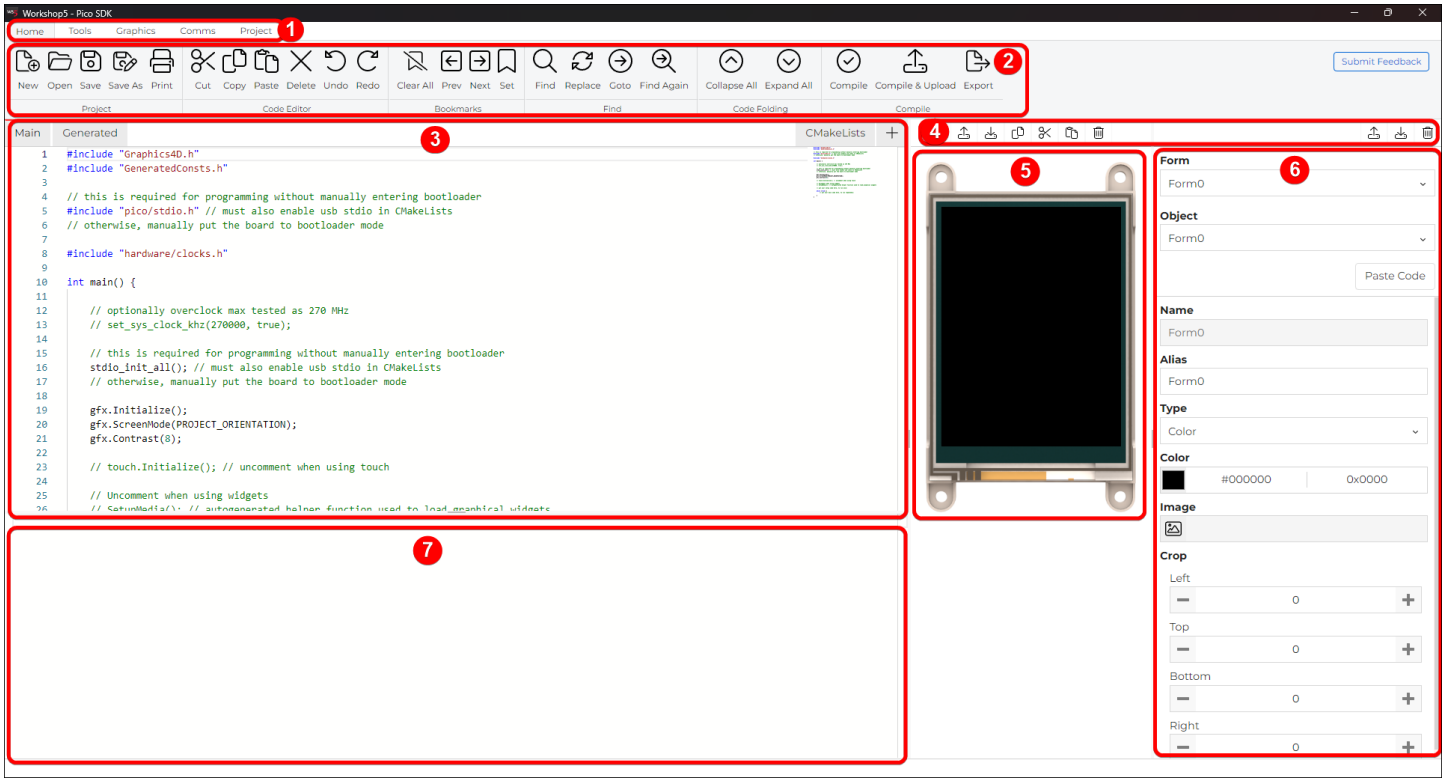


The Pico SDK environment will be display.



5.3. The Main Screen

Let's discuss the different areas of the Pico SDK environment. There are seven (7) different areas, from left to right, from top to bottom.



1. Menus
2. Ribbons with icons
3. Code Editor
4. Graphics Toolbar
5. Visual Editor
6. Object Properties
7. Message Window

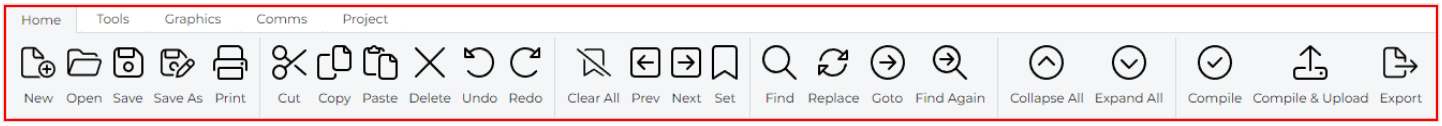
5.3.1. Area 1: Menus

This menu includes standard Windows options. Each menu displays a specific ribbon.

5.3.2. Area 2: Ribbons with Icons

Ribbons with icons are grouped with closely related commands in each menus.

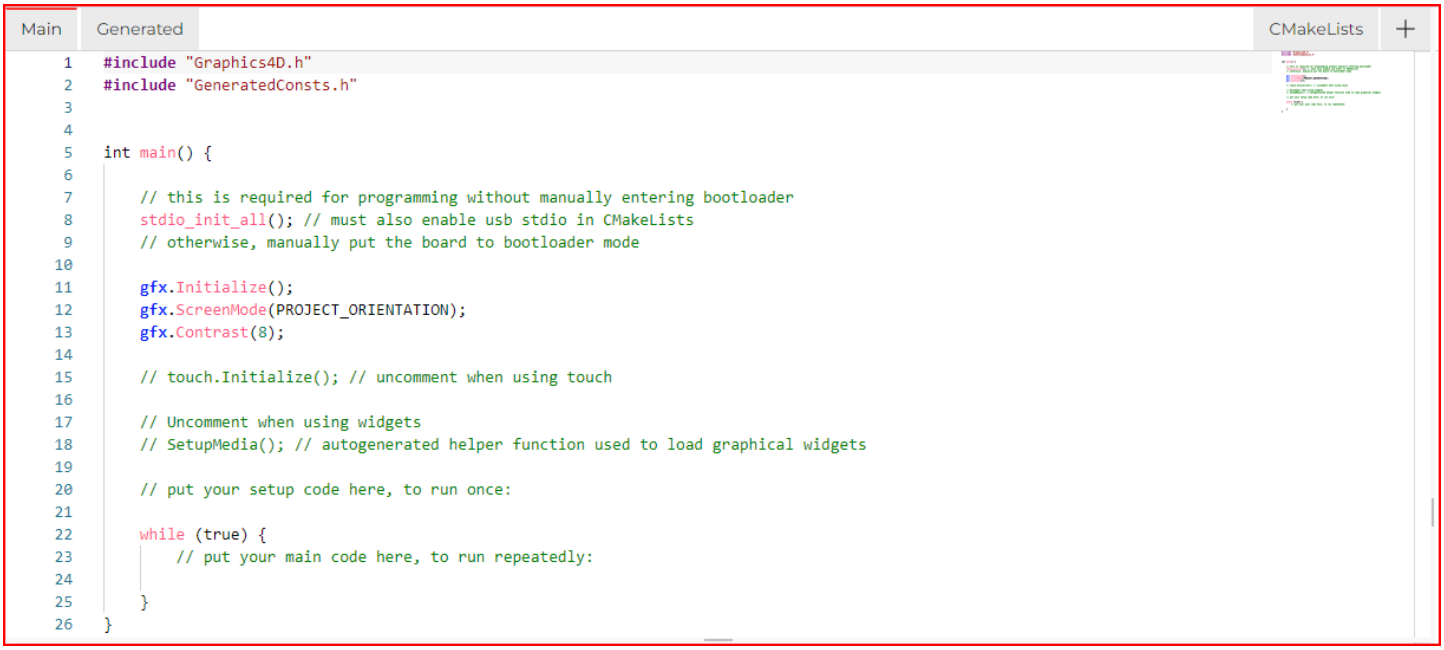
The ribbons on the Home menu are grouped as project, code editor, bookmarks, find and replace, code folding and compile.



5.3.3. Area 3: Code Editor

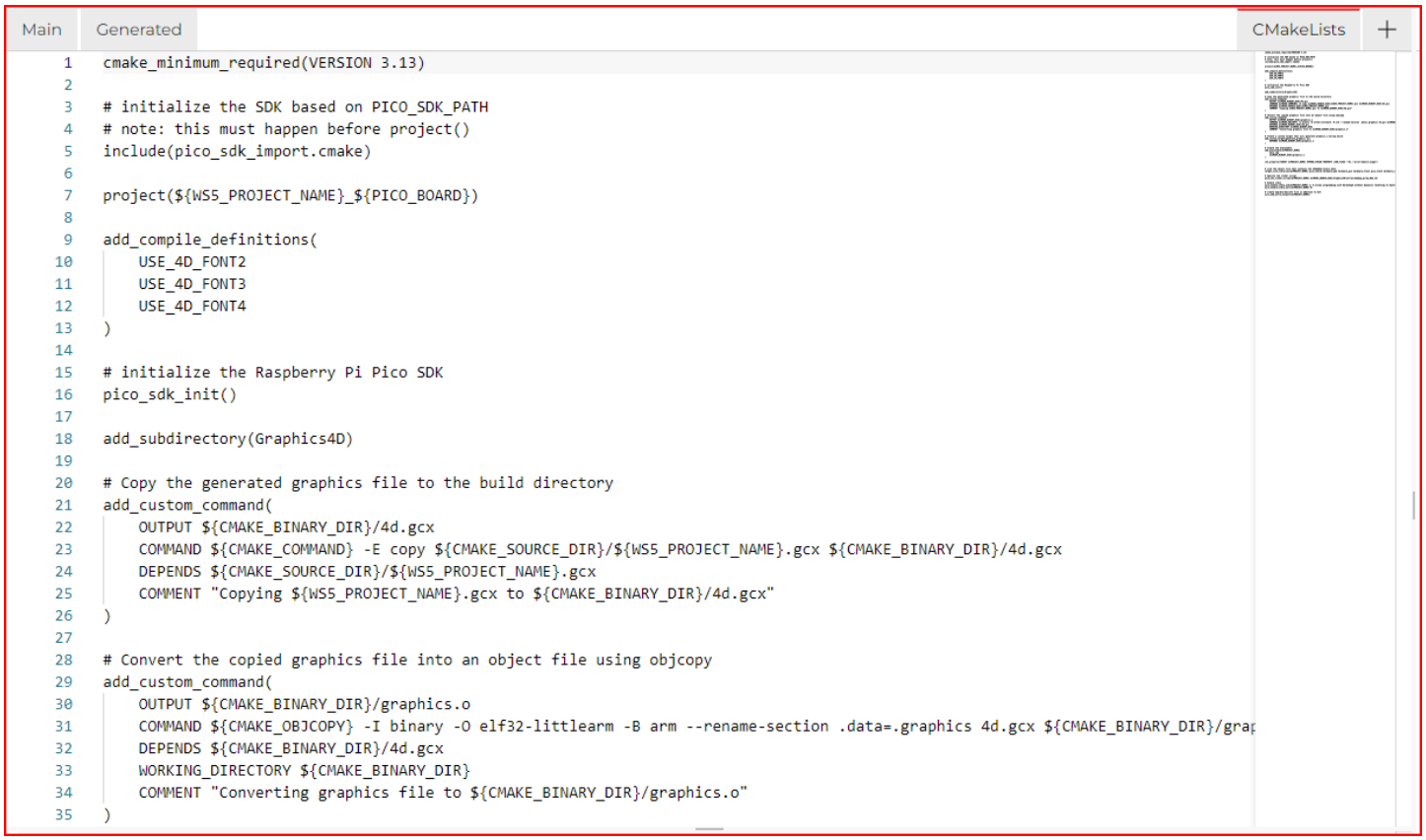
The code editor has three (3) open tabs, Main, Generated and CMakeLists.

The **Main** tab is where to write the C++ code. It has initial lines of codes which are necessary for the project.



The **Generated** tab is a read-only C++ codes that are generated based on project contents and settings.

The **CMakeList** tab is a text file editor that allows custom changes to CMake configuration.



Note

Parts of the CMakeList configuration allows for storing graphical resource to flash and reprogramming the display by resetting to bootloader automatically.

5.3.4. Area 4: Graphics Toolbar

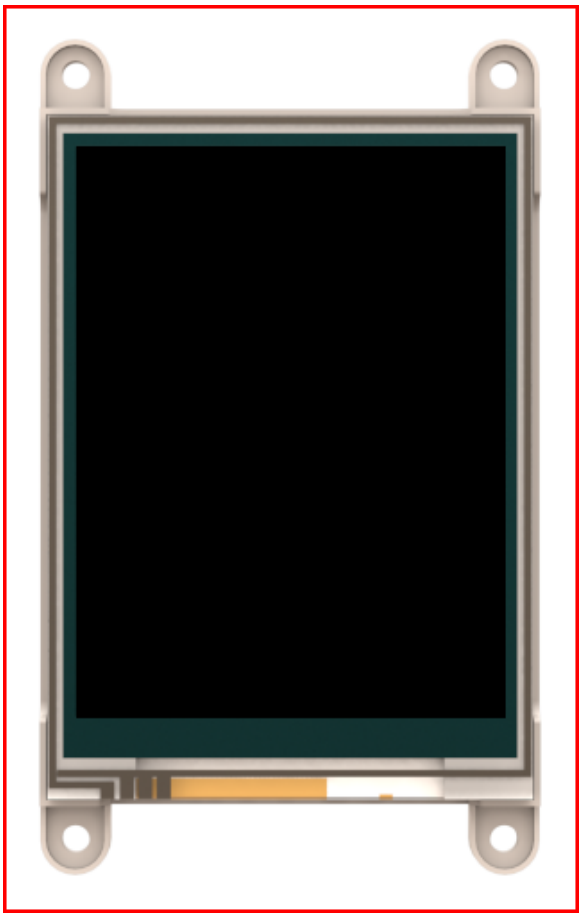
The graphics toolbar provides buttons for managing widgets.

From left to right, the toolbar items are described in the table below.

Item	Description
Load Widget	Opens the Select Widget window.
Save Widget	Opens the Save Widget window.
Copy Widget	Copies the selected widget for pasting.
Cut Widget	Copies the selected widget for moving to another page.
Paste Widget	Pastes the recently copied widget.
Delete Widget	Deletes the selected widget.
Load Form	Opens the Load Form window.
Save Form	Opens the Save Form window.
Delete Form	Deletes the selected form.

5.3.5. Area 5: Visual Editor

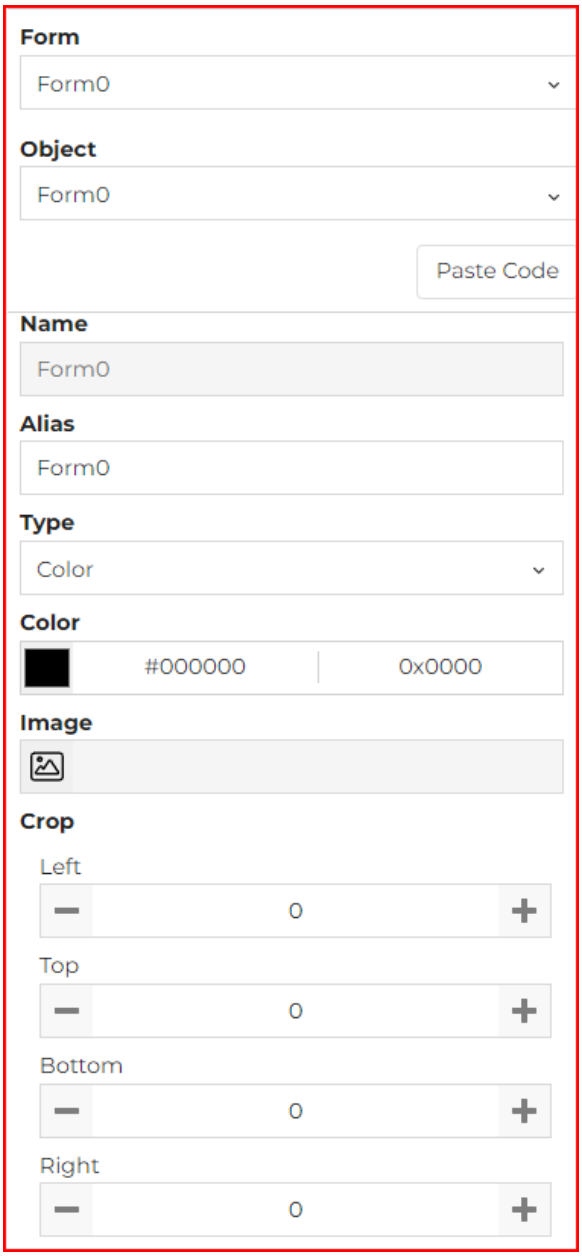
The visual editor represents a What-You-See-Is-What-You-Get (WYSIWYG) area.



The active form is displayed and can be populated by objects from the object selection window.

5.3.6. Area 6: Object Properties

Object properties provides all the information of the selected object. Each object has their own properties that can be set on this area before pasting it on the code editor.



You can use the paste code button to paste the code of the selected object.

5.3.7. Area 7: Message Window

The message window displays errors, warnings and notices after the project was compiled and build.

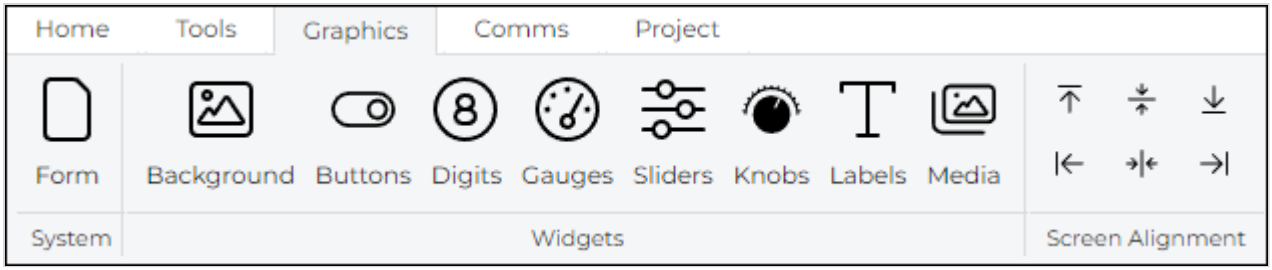
5.4. Designing a Graphical Interface

The Workshop5 IDE provides a simple method for creating graphical user interfaces for 4D Systems display modules. It provides easy-to-use visual editor with support for multiple types of widgets including buttons, sliders, knobs and gauges.

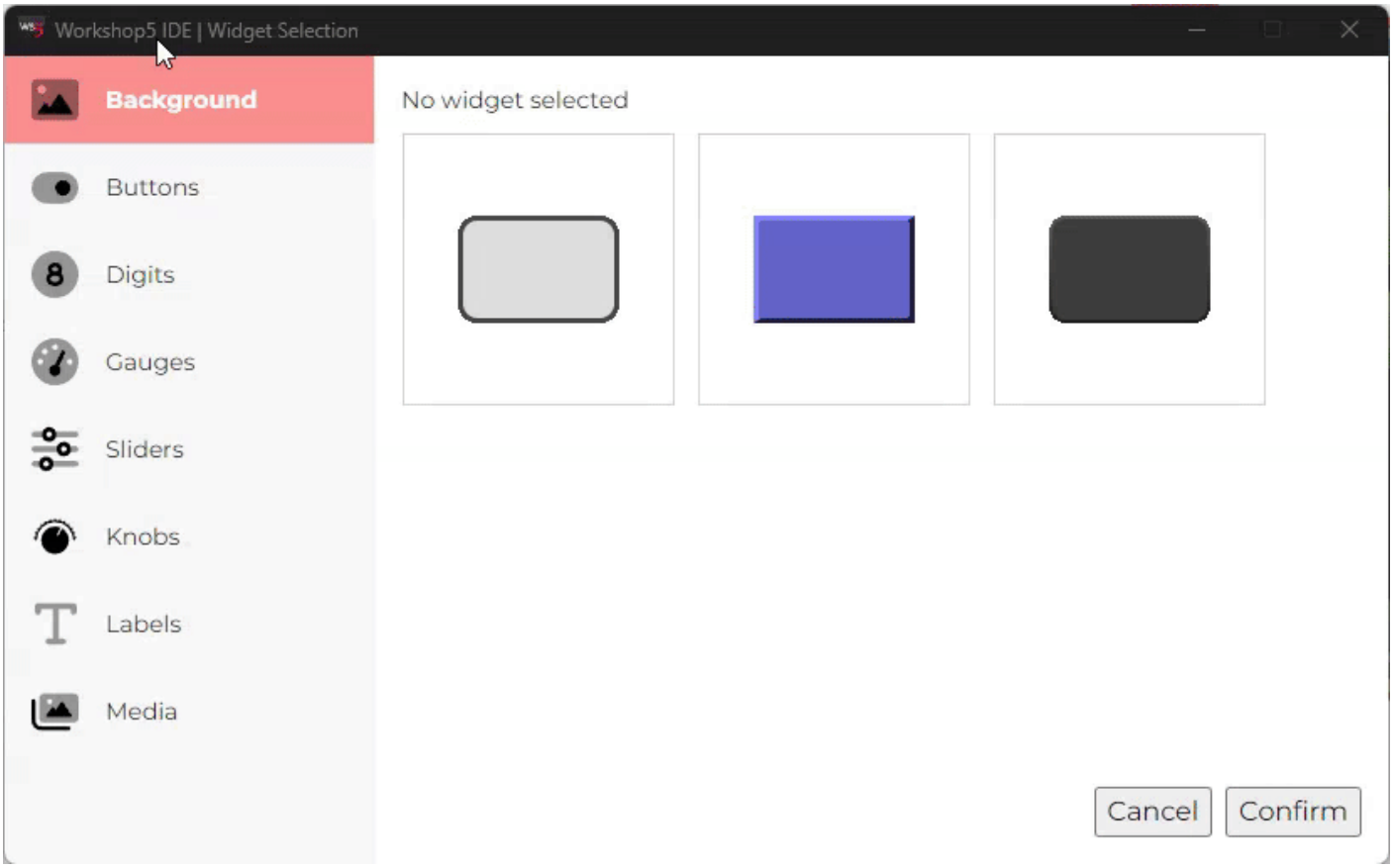
5.4.1. Object Selection

Below is the steps in selecting and putting the objects into the visual editor.

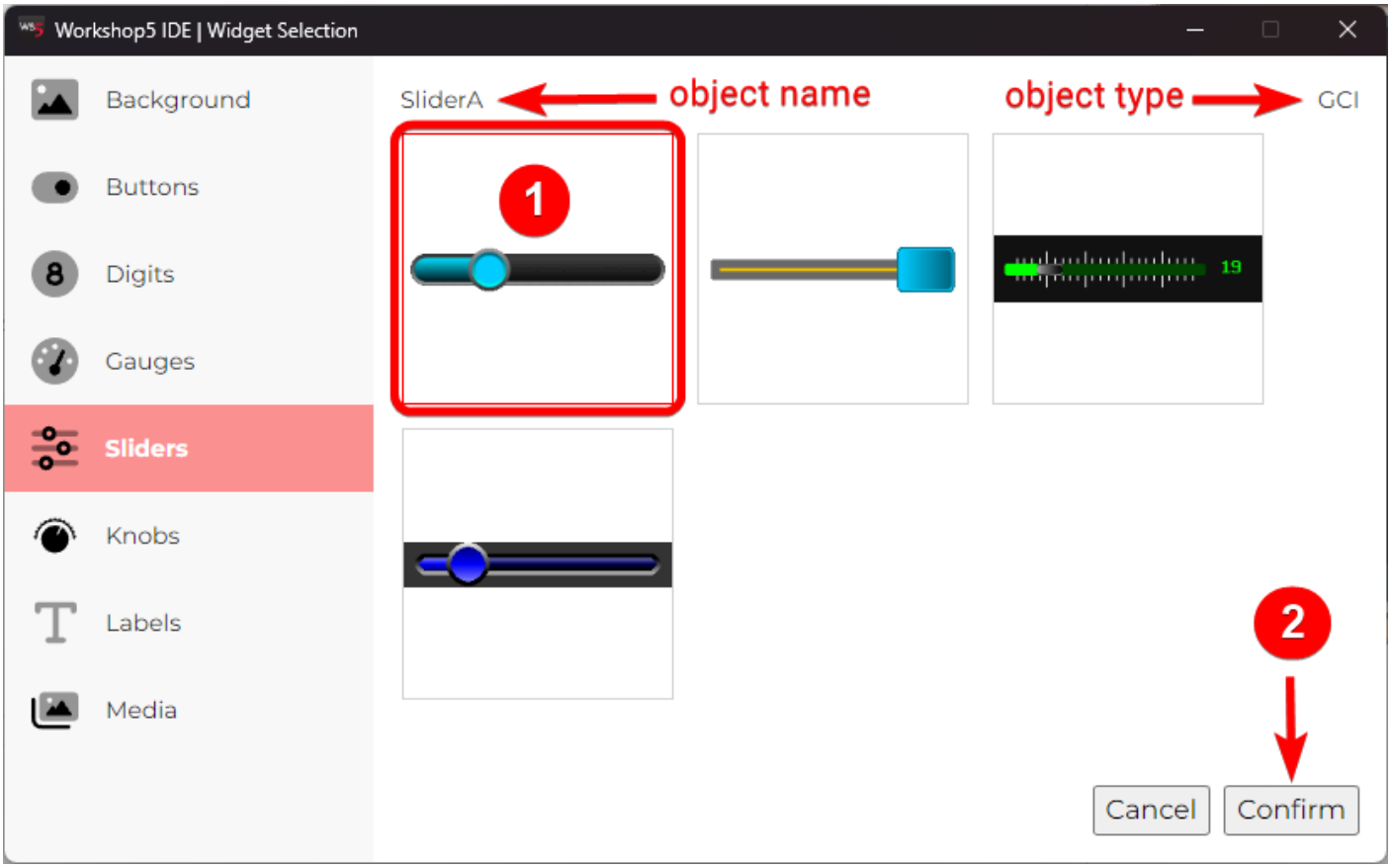
1. To begin, click on the **Widgets** tab and select the object to open the **Widget Selection** window.



2. The widget selection window is open.



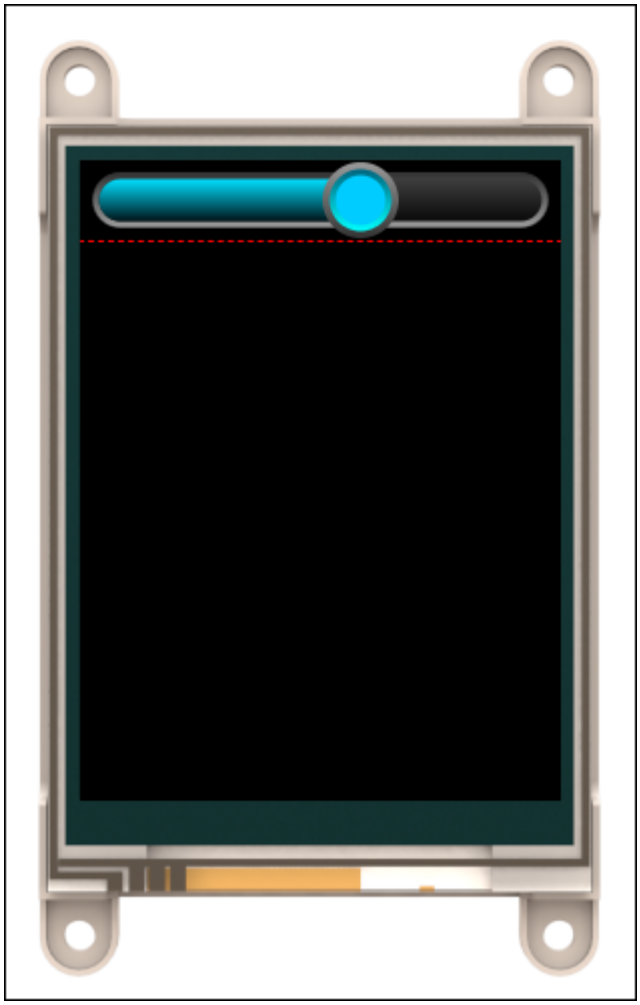
3. Navigate to each widgets pane to select an object. Then click the Confirm button to put the object into the Visual Editor.



Note

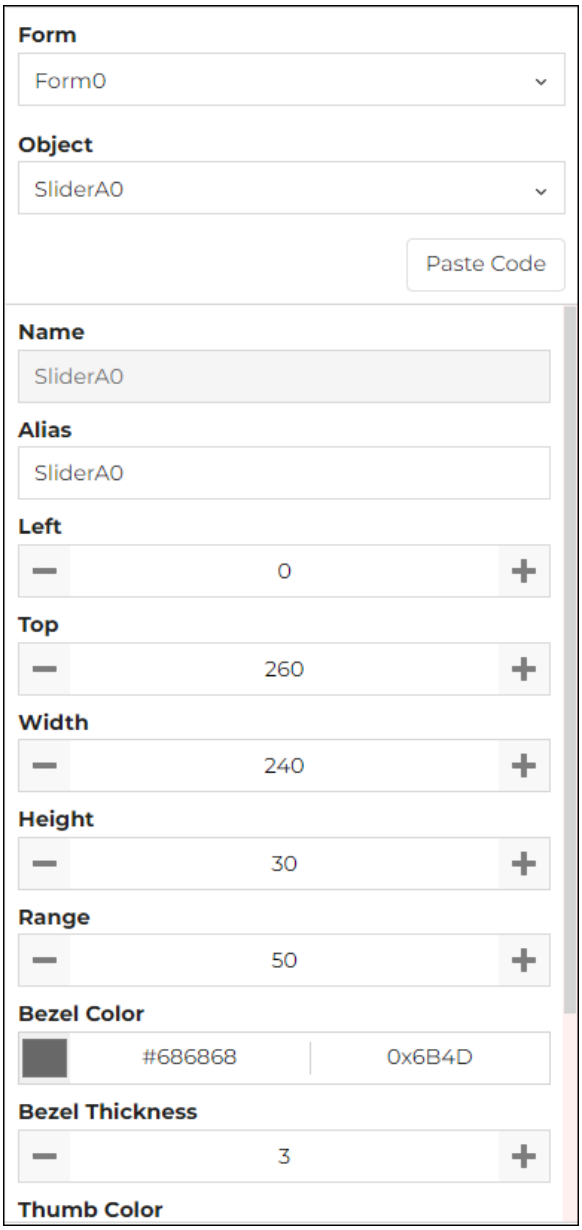
Widgets are filtered according to the selected processor.

4. The object will be shown on the top left of the visual editor.



5.4.2. Object Properties Configuration

Once the object was added to the visual editor, use the object properties to configure the object. Set the position, dimension, appearrance and other properties that neeeded for the project.



To add more objects, just repeat the steps in Object Selection and Object Properties Configuration.

5.4.2.1. Changing Between Object Properties

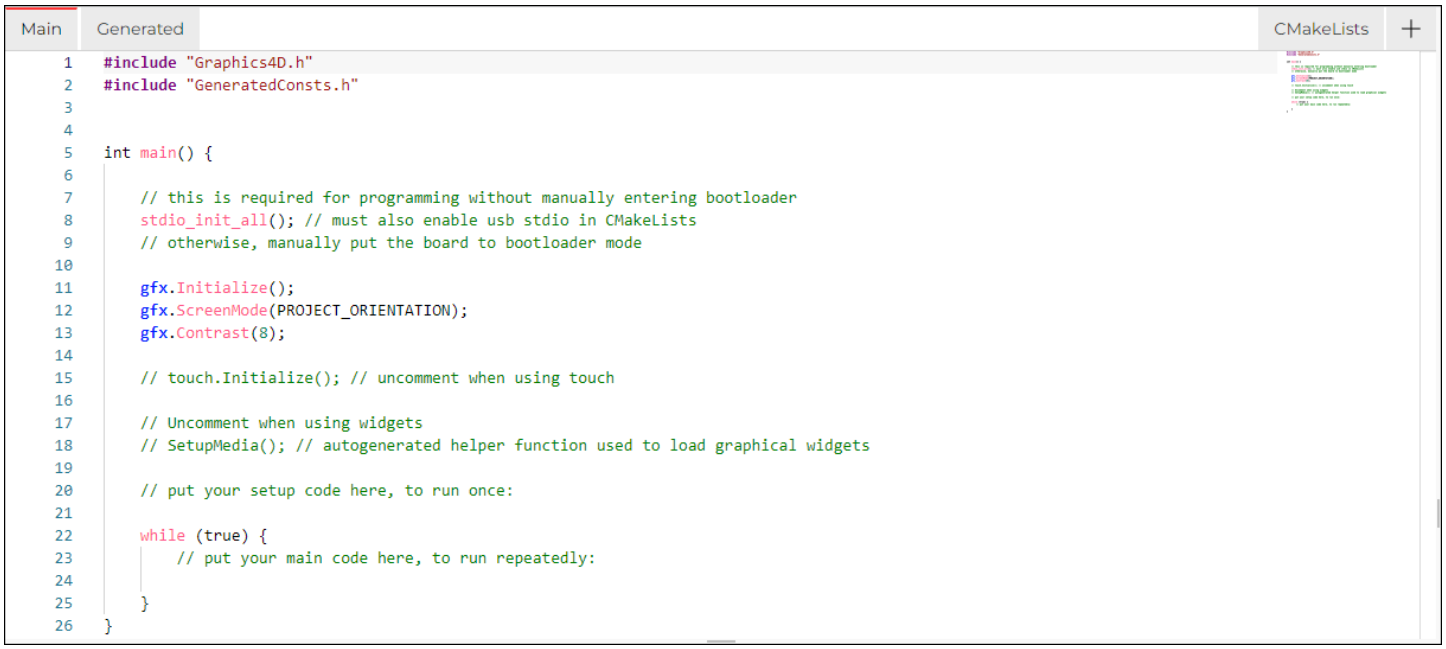
If there are multiple graphical objects on the screen, it may be necessary to change between object properties. Do this by using the dropdown box located at the top of the **Object Inspector**. Alternatively, each object can be edited by clicking on the object in the module display area.

5.5. Writing the Code

Once done in designing the graphical interface, let's now go to the code editor to write the code that we want for our program.

5.5.1. Main Tab

In the code edior **Main** tab contains initial code for including files, libraries and initial setup for the display orientation.



When using the graphical editor, the line `SetupMedia()` function should be uncommented. This function is defined on the **GeneratedConsts.inc** file that is a helper function for the storage media.

5.5.2. Generated Tab

On the **Generated** tab, is a read-only file which is generated automatically during the designing of the user interface. This file defines all the widgets or object that is use on the graphical interface and functions needed on the main file.

This file `GeneratedConsts.inc` is automatically included into the main file.

5.5.3. Generating Widget Code

Workshop5's Pico SDK environment provides the most versatility by allowing users to write their own code while providing a graphics editor. Furthermore, it provides a simple utility that generates code for each widget or object used in the project with a click of a button.

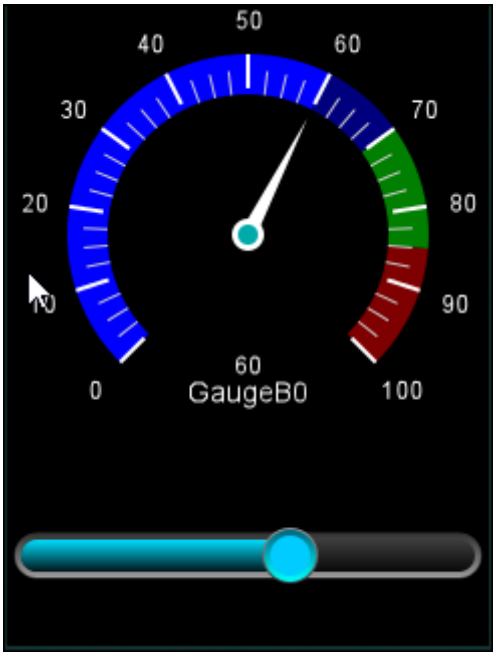
Workshop5 Pico SDK environment provides a **Paste Code** utility that can be used to generate relevant code for the widgets.

This option generates code to update or show widgets at the current cursor position, or more appropriate location or multiple locations in the project.

The following is a list of code snippets Workshop5 generate for a target Workshop5 object.

1. **Set widget value** - the function `img.SetValue()` will set the value of the widget.
2. **Show updated widget** - the function `img.Show()` will be called to show the value of the widget.

Here is an example of a single form project containing a slider and a guage.



The generated code is as shown.

```
#include "Graphics4D.h"
#include "GeneratedConsts.h"


int main() {

    // this is required for programming without manually entering bootloader
    stdio_init_all(); // must also enable usb stdio in CMakeLists
    // otherwise, manually put the board to bootloader mode

    gfx.Initialize();
    gfx.ScreenMode(PROJECT_ORIENTATION);
    gfx.Contrast(8);

    touch.Initialize(); // uncomment when using touch

    // Uncomment when using widgets
    SetupMedia(); // autogenerated helper function used to load graphical widgets

    // put your setup code here, to run once:
    int state, n, x, y;
    int posn;

    img.Show(hndl, iSliderA0); // Show iSliderA0
    img.Show(hndl, iGaugeB0); // Show iGaugeB0

    while (true) {
        // put your main code here, to run repeatedly:
        state = touch.GetStatus();
        n = img.Touched(hndl, -1);

        if (state == TOUCH_PRESSED)
        {
            x = touch.GetX();
            y = touch.GetY();
        }

        if (state  == TOUCH_RELEASED)
        {
            x = touch.GetX();
            y = touch.GetY();
        }

        if (state == TOUCH_MOVING)
        {
            x = touch.GetX();
            y = touch.GetY();

            if (n == iSliderA0)
            {
                posn = y - 8;   // y - top - 8

                if (posn < 0)
                {
                    posn = 0;
```

```
                }
                else if (posn > 223) // width - 17
                {
                    posn = 100;
                }
                else
                {
                    posn = 100 * posn / 223; // max-min * posn /(width - 17)
                }

                img.SetValue(hndl, iSliderA0, posn); // Update SliderA0 value
                img.Show(hndl, iSliderA0); // Show iSliderA0

                img.SetValue(hndl, iGaugeB0, posn); // Update GaugeB0 value
                img.Show(hndl, iGaugeB0); // Show iGaugeB0

            }
        }
    }
}
```

To learn more about the functions use for the above project, please refer to Graphics4D Library Manual.

5.6. Programming the Display

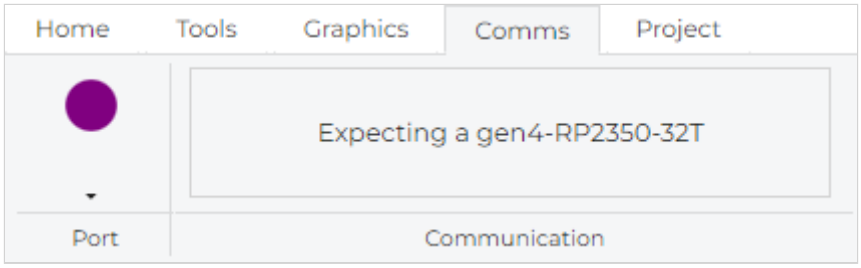
After completing the necessary routine on the project by writing the code. It is now time to upload the project into the display.

5.6.1. Connecting the Module

To connect the target display, simply connect the display module to your computer using a USB type-C cable.

If it's your first time programming the display, or it's been previously configured to not use **USB stdio**, you need to manually put the display into bootloader mode. In this case, there is no need to select a COM port and Workshop5 will automatically attempt to find the RP2350 module.

If the target display is connected with **USB stdio** enabled, the user can now go to the **Comms** tab to select the correct port.



As indicated, Workshop5 expects that the connected device is of the correct display model.

5.6.2. Compile and Upload

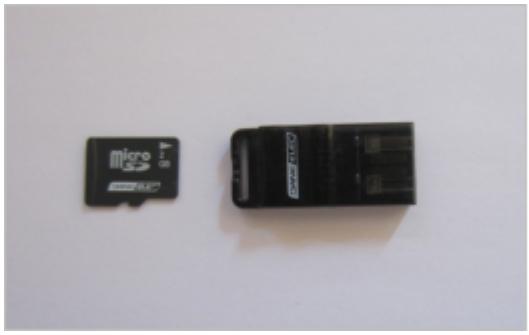
After ensuring that the display is connected and matches the target of the project, go back to **Home** tab and click the **Compile & Upload** button.

5.6.2.1. MicroSD Card

When prompted to copy files to microSD card, proceed with the copy for the first load or when new GCI widgets or fonts are added to the project, or modified in any way.

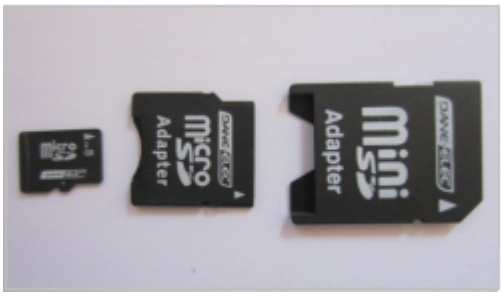
To connect the micro-SD card, either

- Insert the micro-SD card into the USB adaptor and plug the USB adaptor into a USB port of the PC

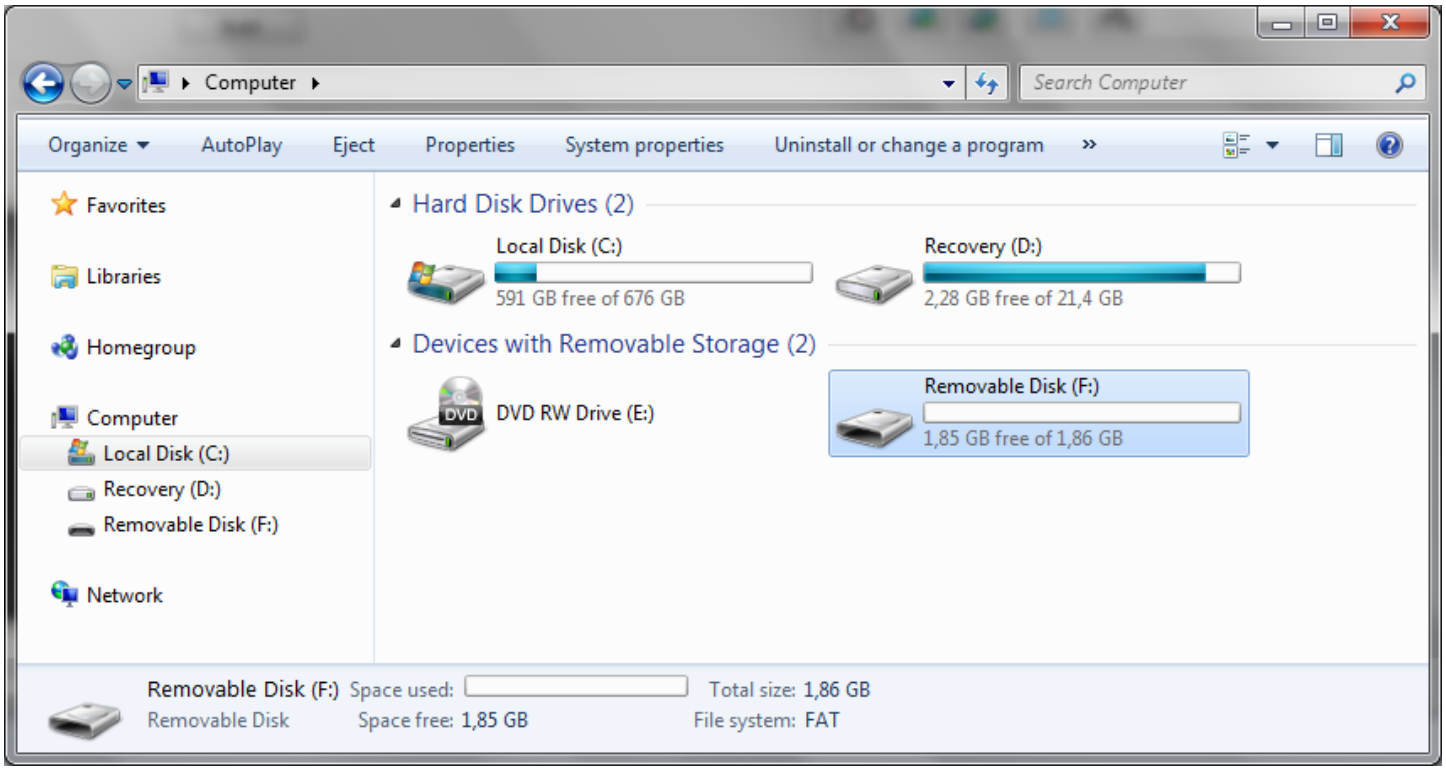


Or

- Insert the micro-SD card into a micro-SD to SD card converter and plug the SD card converter into the SD card slot of the PC.



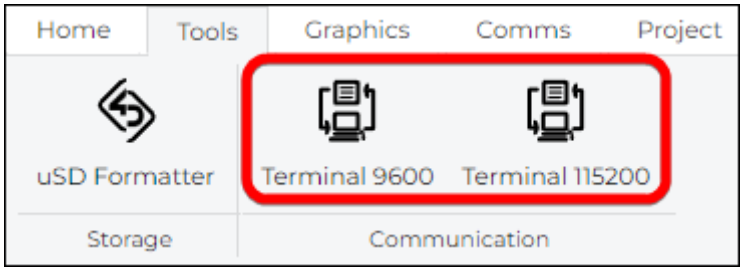
Check the micro-SD card is mounted, here as drive E:.



5.6.3. Debugging the Project

To debug a Pico-SDK project, using **stdio** and **printf** functions to write to USB Serial or UART Serial can be used to send messages to the PC. These messages can then be read by using the Terminal tool.

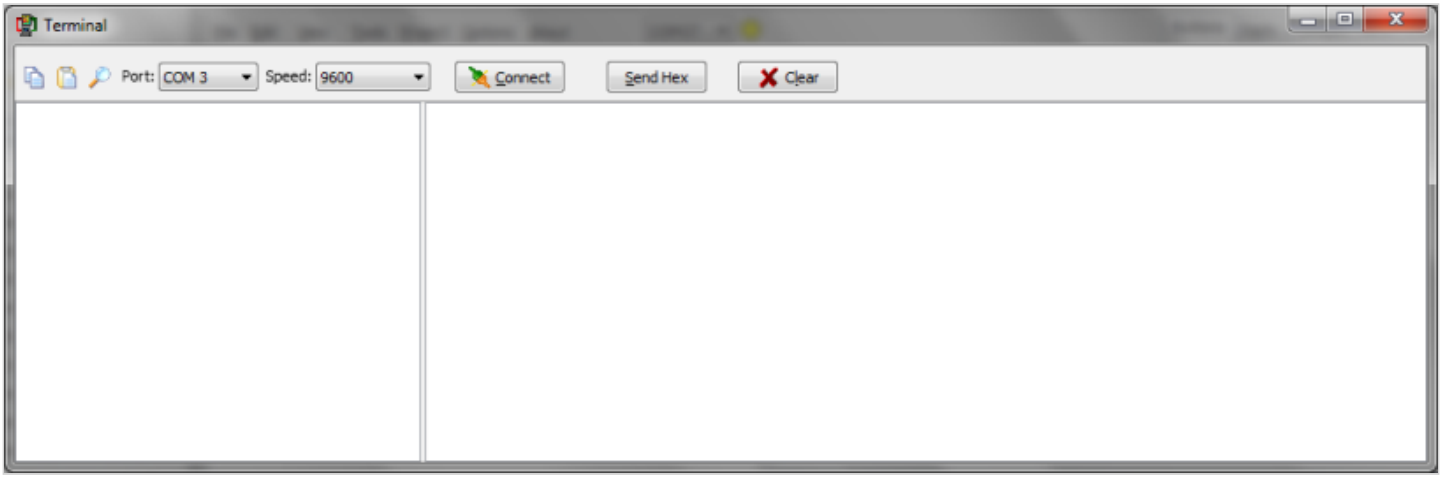
To launch the Terminal, select the **Tools** menu...

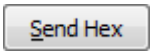


...and

- Click '**Terminal connect 9600**' to open the currently selected com port at 9600 baud in the Terminal program.
- Click '**Terminal connect 115200**' to open the currently selected com port at 115200 baud in the Terminal program.

A new screen appears:

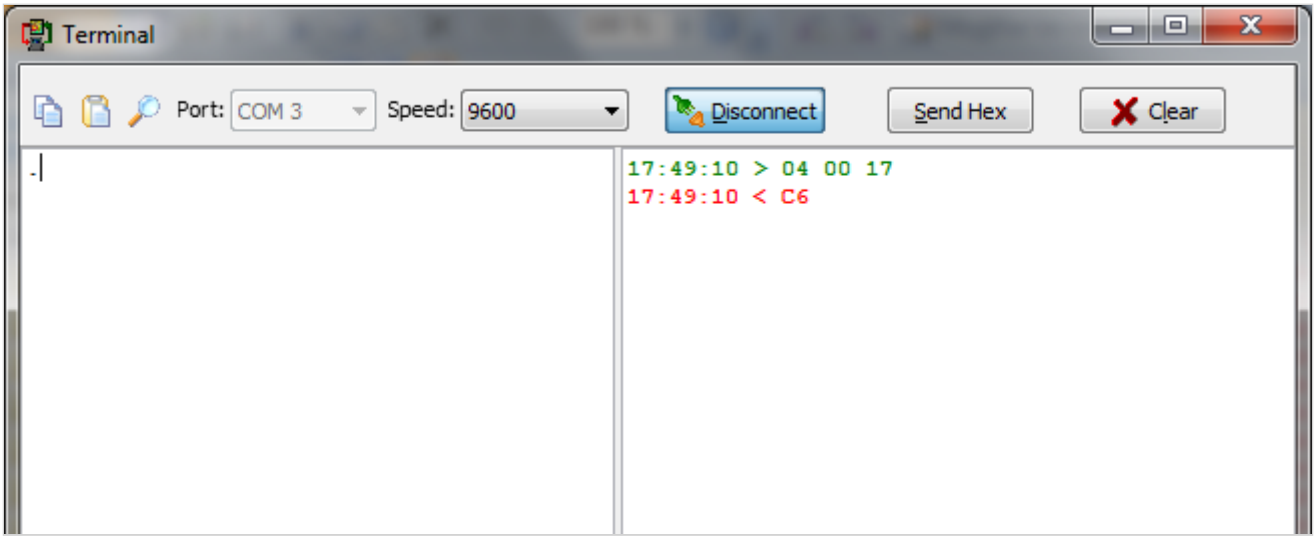


To send the commands on hexadecimal format, press 

The commands sent by the host and the messages sent by the screen are the same as with the **Genie Test Executor** debugger.

The white area on the right displays:

- In **green** the messages sent to the screen;
- And in **red** the messages received from the screen



For more advanced debugging, the project can be exported. Users are able to use their preferred IDEs or code editors with advanced debug tools that can be used in conjunction with the Raspberry Pi Debug Probe.

6. Revision History

Document Revision

Revision Number	Date	Content
1.0	12/11/2024	Initial Release

7. Legal Notice

7.1. 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. 4D Systems reserves the right to modify, update or makes changes to Specifications or written material without prior notice at any time.

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

7.2. Disclaimer of Warranties & Limitations of Liabilities

4D Systems makes no warranty, either expressed 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.

Images and graphics used throughout this document are for illustrative purposes only. All images and graphics used are possible to be displayed on the 4D Systems range of products, however the quality may vary.

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.