Contact Information

Sales and Info: 262-522-6500 x35
sales@ccsinfo.com

Technical Support: 262-522-6500 x32
support@ccsinfo.com

EZ App Lynx

Summary

EZ App Lynx is a smartphone/tablet application that pairs with a PIC® microcontroller (MCU) over Bluetooth and allows the MCU to control the appearance and control of the GUI on the smartphone/tablet. This allows developers an easy way to deploy sensors or controllers that can be controlled from a smartphone/tablet over Bluetooth. Since the PIC MCU controls all aspects of the GUI, the user only needs to install one smartphone/tablet application that works across a broad variety of devices.

There are Android and iOS (iPhone, iPad) Apps available. The Android App can be downloaded from Google Play while the iOS App can be downloaded from the Apple App store. These Apps have already been written by CCS and are available on the appropriate app store, meaning utilizing the EZ App Lynx in a project does not require uploading an app to those stores or becoming an Android or iOS developer.

The EZ App Lynx comprises of two main components. First is the 'App', which is the application that runs on smartphone/tablets. Second is the 'Library' which which runs on the PIC MCU and communicates with the App. Bluetooth is used to connect the the App to the Library. The Android and iOS App both support Bluetooth BLE modules with MLDP (such as the Microchip RN4020 module). Android also supports any Bluetooth 'classic' modules with SPP protocol (the iOS app does not support SPP).

The EZ App Lynx Library for the comes with the IDE version of the CCS C Compiler (PCW, PCWH, PCWHD). The EZ App Lynx Library can also be purchased separately for Microchip MPLAB® XC Compiler users. The Library contains an API that controls all aspects of the GUI on the smartphone/tablet (including what to display on the screen) and also controls the data transfer between then smartphone/tablet and the PIC MCU.

Run screen

This is the main screen of the application. The contents of this screen will change depending on the contents received by the PIC MCU.

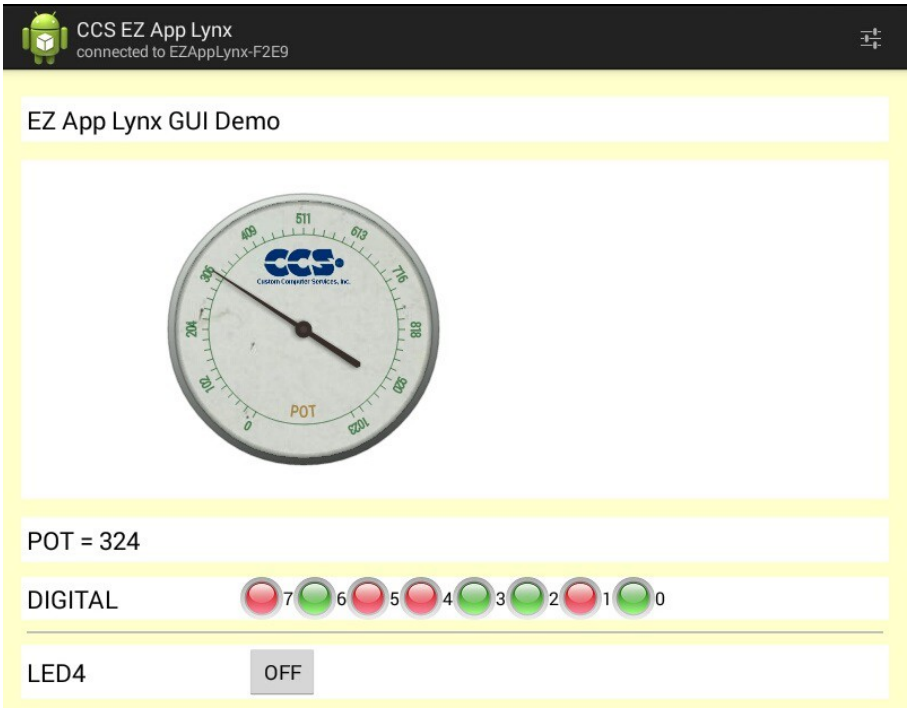


Illustration 1: Example run screen

Settings

Pressing the Settings icon on the toolbar brings up the settings screen.

The settings screen has the following options:

- **Auto Connect**

  If checked, the application will auto connect to the previously connected device when the application is launched.

- **Scan for local devices**

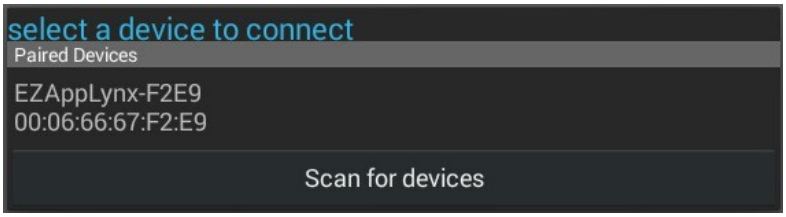


Illustration 2: Scan for local devices

This will show all the available devices the application can connect to. Pressing a device in the list will connect to the device and launch the run screen. If you do not see the device you are looking for, press the 'Scan for devices' button and it will search for new devices.

- **Reset BT adapter on connect**

  This will reset the Bluetooth adapter on the phone/tablet before establishing a connection to the hardware. This generally improves connection throughput and stability over time, but will sever any connections to previously connected Bluetooth devices.

- **Orientation**

  This allows you to lock the application orientation to either Portrait or Landscape.

If the device requires authorization, the application will ask the user for the password before showing the Run screen.

Version 2 Updates

The following changes and additions have been made to version 2 of the App and Library:

- Android supports RN4020 BLE Bluetooth module using it's MLDP mode.
- iOS application (iPhone, iPad, etc). Only the RN4020 module is supported, the iOS application does not support SPP Bluetooth modules like the Android application.
- EZAppAddFieldText() has been added to the API, allows the user to add a left column header to a string or to make a text string editable from the host application. EZAppGetValueString() allows the PIC MCU to get changes from the host. EZAppSetValueStringEE() and EZAppGetValueStringEE() also allow the EZ App Lynx library to read/write strings from an external memory device, like an EEPROM.
- EZAppAddFieldButtons() and EZAppAddFieldButtonsROM() added to the API, that allows for more button types. This allows for a one-state button (value sent by host application is set when button is held down, cleared when released), a series of buttons on one row and the ability to add inset status LEDs inside the buttons. EZAppSetButtonLED() added to control the status of the inset status LED.
- EZAppStylesROM() added to the API that adds global configuration of many style elements in the application, such as colors, spacing, padding, etc.
- EZAppAddFieldAnalogValueScaled() added to the API that allows user to create analog fields that are scaled and/or have a minimum value that isn't 0.
- Added Microchip XC8, XC16 and XC32 compiler support.

Version 3 Updates

The following changes and additions have been made to version 3 of the App and Library:

- EZAppAddFieldGraph() and EZAppAddFieldGraphSeries() added to the API. This allows for adding line and bar graph fields, and setting of colors, line styles, axis and title labels. Multiple Series fields can be applied to one Graph field.
- EZAppAddFieldImage() added to the API. This allows for adding images to a field sourced from URLs or from some built-in images. Image fields can also be set as one or two-state buttons with a different image for each state.
- Added new examples: ex_ezapp_graphing.c, and ex_ezapp_images.c

Changing the contents of the Run screen

The contents of this screen is controlled dynamically by the Library. CCS provides the Library for the CCS C Compilers and Microchip MPLAB® XC Compilers that allows the user to control those contents. Read the EZApp.h file in the compiler's drivers directory for documentation of the API of this library. Also provided are several examples of how to use this library. To find these examples, look for files in the examples directory for files that start with ex_ezapp (like ex_ezapp_pot.c).

Here is an overview of all the GUI elements that can be used/controlled by the library:

Text Field
 	
EZ App Lynx GUI Demo
	Raw text is displayed as-is from the PIC MCU. The application will reload the value when the PIC MCU changes it.
	
	Example code:
	`idx = EZAppFieldString();`
	`EZAppSetValueString(idx,`
	`"EZ App Lynx GUIDemo");`
	
	Relevant API functions:
	`EZAppAddFieldString()`
	`EZAppAddFieldStringDynamic()`
	`EZAppSetValueString()`
	`EZAppSetValueStringROM()`
	`EZAppSetStringStyle()`

Text field, with header column
 	
Status: Good
 	
Added in V2
	Raw text is displayed as-is from the PIC MCU. The application will reload the right value column when the PIC MCU changes it (the left header column cannot be changed).
	
	Example code:
	`idx = EZAppFieldText((rom char*)"Status:", 0);`
	`EZAppSetValueString(idx, "Good");`
	
	Relevant API functions:
	`EZAppAddFieldText()`
	`EZAppSetValueString()`
	`EZAppSetValueStringROM()`
	`EZAppSetStringStyle()`

Editable text
 	
Name: Left Fan
 	
Added in V2
	An editable text box is displayed. When new text is entered into this field it is sent to the PIC MCU.
	
	Example code:
	`char editStr[LEN] = "\0";`
	`idx = EZAppFieldText(`
	`(rom char*)"Name:",`
	`sizeof(editStr));`
	`EZAppSetValueString(idx, editStr);`
	
	Relevant API functions:
	`EZAppAddFieldText()`
	`EZAppSetValueString()`
	`EZAppSetValueStringROM()`
	`EZAppSetStringStyle()`
	`EZAppGetKbhit()`
	`EZAppGetValueString()`

One Button Field	A pressable button. When pressed, the value is toggled and sent to the PIC MCU. The text string on the button represents the current value.

Example code:

```
idx = EZAppAddFieldButtonTwoState(
      (rom char *)"LED4",
      (rom char *)"OFF\tON");
```

Relevant API functions:

```
EZAppAddFieldButtonTwoState()
EZAppGetKbhit()
EZAppGetValue()
EZAppSetValue()
```

Buttons Field

Added in V2

One or a series of buttons on one row. Buttons can be one-state or two-state. Buttons can have an inset status LED (optional, not required).

Example code:

```
ezapp_buttons_t buttonCfg;
buttonCfg.numButtons = 3;
buttonCfg.oneState = TRUE;
idx = EZAppAddFieldButtonsROM(
      buttonCfg,
      (rom char *)"LEDS",
      (rom char*)"LED0\tLED1\tLED2");
```

Relevant API functions:

```
EZAppAddFieldButtons()
EZAppAddFieldButtonsROM()
EZAppGetKbhit()
EZAppGetValue()
EZAppSetValue()
EZAppSetButtonLED()
```

Digital Field

A series of LEDs is displayed to display a digital (on/off) state. The application will reload the value when the PIC MCU changes it.

Example code:

```
idx = EZAppAddFieldDigitalValue(
      (rom char*)"Digital", 8);
EZAppSetValueDigital(idx, 0x4d);
```

Relevant API functions:

```
EZAppAddFieldDigitalValue()
EZAppSetValue()
```

Slider

A slider. Application can control and send new value as the user slides the position.

Example code:

```
idx = EZAppAddFieldAnalogValue(
      (rom char *)"DAC12",
      EZAPP_ANALOG_TYPE_SLIDER_RW, 4095);
EZAppSetValueAnalog(idx, 1000);
```

When creating field with EZAppAddFieldAnalogValue(), use `EZAPP_ANALOG_TYPE_SLIDER_RW` type.

Relevant API functions:

```
EZAppAddFieldAnalogValue()
EZAppSetValue()
EZAppGetValue()
```

	`EZAppGetKbhit()`
Slider (Read only) 	This is the same as the Slider GUI type, but this one is read-only. The application cannot update this value, instead the value is read from the PIC MCU. When creating field with EZAppAddFieldAnalogValue(), use EZAPP_ANALOG_TYPE_SLIDER type.

Pull-down
 
	A pull-down list of select-able items. When user selects new item, application sends it to the PIC MCU.
	
	Example code:
	`idx = EZAppAddFieldPulldownValue(`
	`(rom char*)"LED1", 2,`
	`(rom char*)"Off\tOn");`
	
	Relevant API functions:
	`EZAppAddFieldPulldownValue()`
	`EZAppSetValue()`
	`EZAppGetValue()`
	`EZAppGetKbhit()`

Numeral Field
 
	A numeric value that can be edited with a text-field. When new values are entered by the user in the application it is sent to the PIC MCU.
	
	Example code:
	`idx = EZAppAddFieldAnalogValue(`
	`(rom char *)"DAC15",`
	`EZAPP_ANALOG_TYPE_RW_TEXT_VALUE, 4095);`
	
	`When creating field with EZAppAddFieldAnalogValue(), use EZAPP_ANALOG_TYPE_RW_TEXT_VALUE type.`
	
	Relevant API functions:
	`EZAppAddFieldAnalogValue()`
	`EZAppSetValue()`
	`EZAppGetValue()`
	`EZAppGetKbhit()`

Gas Gauge

A representation of a numeral value with an analog gas gauge type display.

Example code:

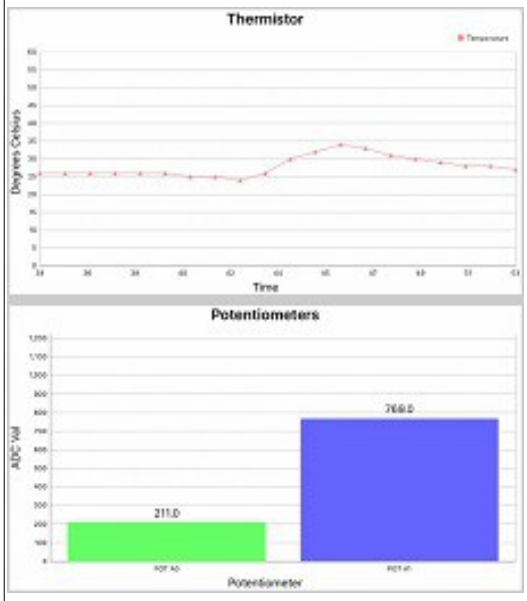
```
idx = EZAppAddFieldAnalogValue(
      (rom char *)"POT",
      EZAPP_ANALOG_TYPE_GAUGE, 1023);

When creating field with
EZAppAddFieldAnalogValue(), use
EZAPP_ANALOG_TYPE_GAUGE type.
```

Relevant API functions:

```
EZAppAddFieldAnalogValue()
EZAppSetValue()
```

Graph and Series Fields

A single graph consists of two field types: Graph and Series. Multiple Series can be added to one Graph.



Added in V3

A Graph field is a container for displaying Graph Series. Graph settings such as type (bar or line), title, axis labels, grid lines, etc. are set through the *ezapp_graph_cfg_t* structure.

A Series field is the data set displayed on the previously added graph. Settings such as line/bar color, line style, and increment can be set. Multiple Series fields can be added to one Graph field to show multiple data sets on one graph.

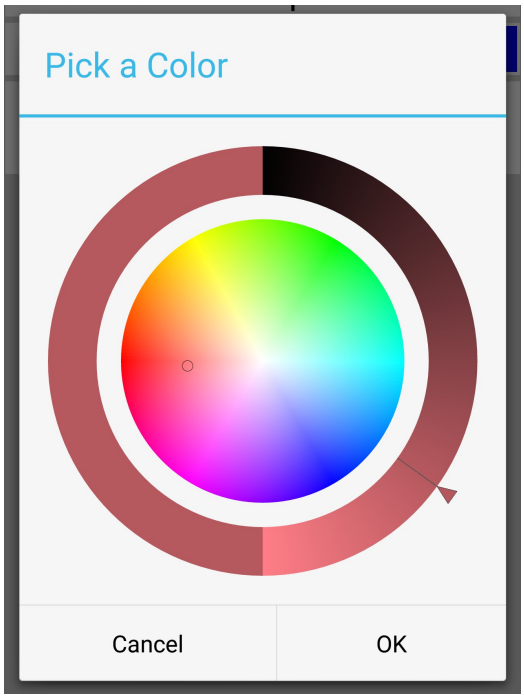
Example Code:

```
ezapp_graph_cfg_t potGraph;

potGraph.decimalX = 0;
potGraph.decimalY = 0;
potGraph.height = 10;
potGraph.maxX = 100;
potGraph.minX = 0;
potGraph.minY = 0;
potGraph.maxY = ADC_MAX+200;
potGraph.scaleX = 1;
potGraph.scaleY = 1;
potGraph.ticksX = 10;
potGraph.ticksY = 10;
potGraph.scrollX = TRUE;
potGraph.scrollY = FALSE;
potGraph.showLegend = FALSE;
potGraph.numPoints = 20;
potGraph.type = EZAPP_GRAPH_BAR;
potGraph.showXGrid = FALSE;
potGraph.showYGrid = TRUE;
potGraph.labels = (rom char *)
                   "Potentiometer\tADC Val";
potGraph.title = (rom char *)"Potentiometers";

EZAppAddFieldGraph(&potGraph);
```

	pot0SeriesIndex = EZAppAddFieldGraphSeries(EZAPP_GRAPH_LINE_SOLID, EZAppCreateRGB(100, 255, 100), 1, (rom char *)"POT A0"); pot1SeriesIndex = EZAppAddFieldGraphSeries(EZAPP_GRAPH_LINE_SOLID, EZAppCreateRGB(100, 100, 255), 1, (rom char *)"POT A1"); **Relevant API functions:** EZAppAddFieldGraph() EZAppAddFieldGraphSeries() EZAppSetValue()

Image Field
 
 	
Added in V3
	Image(s) sourced from URLs or from some built-in images. Image fields can also be set as one or two-state buttons with a different image for each state. Image properties are set through an *ezapp_image_cfg_t* structure.
	
	Example Code:
	
	ezapp_image_cfg_t imgCfg;
	
	imgCfg.scaleType = EZAPP_IMAGE_SCALE_HALF;
	imgCfg.button = TRUE;
	imgCfg.oneState = TRUE;
	imgCfg.preloaded = FALSE;
	imgCfg.useLogo = FALSE;
	
	imgIdx = EZAppAddFieldImage(
	imgCfg,
	(rom char *)
	"http://www.ccsinfo.com/images/content/green-on-
	button.png\thttp://www.ccsinfo.com/images/content/red-off-button.png"
	);
	
	Relevant API functions:
	EZAppAddFieldImage()
	EZAppSetValue()
	EZAppGetValue()
	EZAppGetKbhit()

Color Picker Field  	Color picker with label on the left (optional) and selected color view on the right. Touching the color view will open a dialog allowing the user to pick a new 24-bit color. Requires using *EZAppGetValue32()* and *EZAppSetValue32()* to get/set the full 24-bits. **Example Code:** ezIdxColorPicker = EZAppAddFieldColorPicker((rom char*)"Color:");

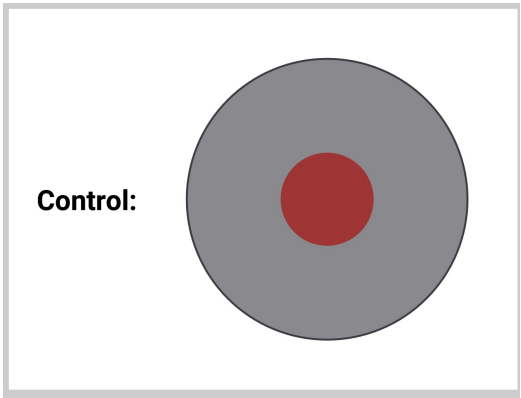


Added in V4

Relevant API functions:

```
EZAppAddFieldColorPicker()
EZAppSetValue32()
EZAppGetValue32()
EZAppGetKbhit()
```

Joystick Field



Added in V4

Joystick with label on the left (optional) and joystick view on the right. The small circle, or "stick," can be moved around the area of the larger base circle, simulating an analog joystick. When moving, this gives "Angle" (0-359 degrees) and "Strength" (0-100%) values. Joystick properties are set through an *ezapp_joystick_cfg_t* structure. Angle and strength can be extracted from the field value using *EZAppGetJoystickAngle()* and *EZAppGetJoystickStrength()*.

Example Code:

```
ezapp_joystick_cfg_t joyCfg;
joyCfg.backgroundColor =
   EZAppCreateRGB(0x8A,0x89,0x8D);
joyCfg.borderColor =
   EZAppCreateRGB(0x3D,0x3B,0x41);
joyCfg.stickColor =
   EZAppCreateRGB(0x9F,0x35,0x34);
joyCfg.borderWidth = 5;
joyCfg.maxHeight = 15;
joyCfg.autoReCenter = true;
ezIdxJoystick = EZAppAddFieldJoystick(&joyCfg,
(rom char*)"Control:");
```

Relevant API functions:

```
EZAppAddFieldJoystick()
EZAppGetValue()
EZAppGetKbhit()
EZAppGetJoystickAngle()
EZAppGetJoystickStrength()
```

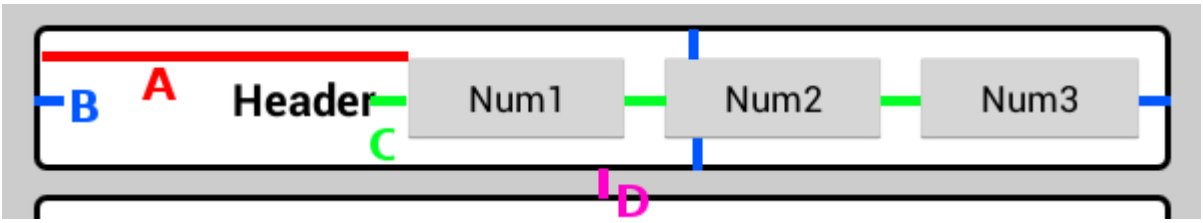
The library allows you the user to develop a multi-screen GUI. This can be achieved with the function EZAppFieldsClearAll(), which clears the screen of all previous GUI elements.

Applying Styles

Added in V2. EZAppStylesROM() can be used in the Library API to apply one or more styles. A style contains information about how things should look, like colors, sizes, margins and spacing, etc.

After EZAppAddStylesROM() is called, all following fields added with EZAppAddField***() will use the previous added styles. Styles can also accumulate, so calling EZAppAddStylesROM() a second time will add new styles on top of the previously added styles.

See the ex_ezapp_style.c and ex_ezapp_backgrounds.c for a demonstration of how to use and apply styles.



A = the length of header column, if it's used. This is set with the 'leftWidth' parameter of the EZAPP_STYLE_ROW_SPACINGS_CREATE() macro.

B = The inner padding. This is set with the 'paddingInner' parameter of the EZAPP_STYLE_ROW_SPACINGS_CREATE() macro.

C = The per column padding. This is set with the 'paddingPerColumn' parameter of the EZAPP_STYLE_ROW_SPACINGS_CREATE() macro.

D = The padding below each row. This is set with the 'paddingBelow' parameter of the EZAPP_STYLE_ROW_SPACINGS_CREATE() macro.

Legal information

EZ App Lynx is developed and maintained by Custom Computer Services, Inc (CCS). For support and more help and documentation regarding EZ App Lynx, go to http://www.ccsinfo.com/ezapp

PIC and MPLAB are registered trademarks of Microchip Technology Inc in the US and other countries.