



# **Board Support Packages Help**

MPLAB Harmony Integrated Software Framework

## Volume II: Supported Hardware

---

This volume provides information on hardware supported by MPLAB Harmony, including information on devices, development boards, and board support packages.

### Description



MPLAB Harmony Board Support Packages provide software support for PIC32 device families and related demonstration development hardware.

A Board Support Package (BSP) provides support for board-specific hardware, which is provided in two forms:

- **BSP Library:** Provides board-specific initialization code and an interface to switches, LEDs, and GPIO pins
- **Initial Configuration:** Provides a starting point for configuration settings for clocks, ports, and selected libraries when creating a new project (or project configuration)



## Board Support Packages Help

This section describes the Board Support Packages (BSP) that are available in MPLAB Harmony.

### Introduction

This topic provides information for the Board Support Package (BSP) capabilities MPLAB Harmony.

### Description

A MPLAB Harmony Board Support Package (BSP) provides support for board-specific hardware, which is provided in two forms:

- **BSP Library:** Provides board-specific initialization code and an interface to switches, LEDs, and GPIO pins
- **Initial Configuration:** Provides a starting point for configuration settings for clocks, ports, and selected libraries when creating a new project (or project configuration)

### BSP Library

A MPLAB Harmony BSP library (consisting of a `bsp.h` and `bsp.c` file) is configured specifically for the selected board and generated into the project's configuration folder at the following location:

```
<project>/firmware/src/system_config/<configuration>/bsp
```

Where:

`<project>` - Is the root folder of the project

`<configuration>` - Is the project's configuration folder when the BSP is generated

A BSP library has a set of system interface functions and a set of client interface functions. However, unlike most MPLAB Harmony library modules, BSPs do not often have a state machine of their own, and therefore, they usually do not have a `BSP_Tasks` function or other system interface functions. In a project configuration, the BSP's initialize function (`BSP_Initialize`) is normally called from the system's `SYS_Initialize` function before any other initialization function (except those that initialize any required core processor capabilities). This allows the BSP to initialize any necessary board-specific hardware (like memory and GPIO controllers) before other libraries are initialized. Refer to *Volume IV: MPLAB Harmony Development > MPLAB Harmony Driver Development Guide > System Interface* for additional information on system interface functions.)

A BSP also provides a simple and extensible client interface for controlling switches, LEDs, and other GPIO pins. This interface includes functions to turn LEDs and pins on-or-off and to toggle, set, or get their current state, where so configured. The names of these functions all begin with "`BSP_LED`" and are followed by the operation verb (On, Off, Toggle, StateSet, or StateGet). There is also a BSP function to get the current state of switches on the board called `BSP_SwitchStateGet`. These functions accept values from a custom-generated enumeration as a parameter to identify the specific switches, LEDs, and pins on which to perform the operation. The names given to the switches, LEDs, and other pins used by the BSP interface functions are defined in the MPLAB Harmony Configurator's Pin Settings window. These names are used to define the labels in the custom-generated enumerations used by these BSP functions. Refer to *Volume III: MPLAB Harmony Configurator (MHC) > MPLAB Harmony Configurator User's Guide > MPLAB Harmony Graphical Pin Manager* and to the following topics in this section for additional information on how to define custom pin names.

Additionally, the BSP client interface is extensible. The names given to the switches, LEDs, and pins in the Graphical Pin Manager are also prepended to the operation verbs (On, Off, Toggle, StateSet, and StateGet) to generate a completely custom set of control functions for each pin. These functions are described in the Library Interface section of this document using monikers prefixed with "custom\_gpio\_name", "custom\_switch\_name", and "custom\_gpio\_name". No actual functions exist with these names. Instead, these labels are replaced with the custom name given to each pin and function-like macros are generated (defined in `system_config.h`) to provide functions that directly access these IO pins. These macros can be used by the application, but be aware that the common BSP functions may be more portable as these custom macros must be generated by the configuration. For more information, refer to [Extending the BSP Library Interface](#).

### Initial Configuration

An initial BSP configuration is basically another type of MHC configuration file. (Refer to *Volume III: MPLAB Harmony Configurator (MHC) > MPLAB Harmony Configurator Developer's Guide > MHC Files* for details of the MHC file.) The initial configuration can contain settings selected using the MHC (see *Volume III: MPLAB Harmony Configurator (MHC)* for information on using the MHC). When a new project (or a new configuration of an existing project) is created (or any time a BSP configuration is imported), the MHC allows the user the option to select an initial board configuration (MHC) file. If one is selected, the MHC initializes the project configuration with the settings in the selected file (overwriting the default or existing settings). This gives the user a known starting point from which to configure the project, based upon the board selected. A single BSP can have multiple initial board configuration files with different configuration settings, allowing the user to select the initial configuration that most closely represents the desired project configuration.

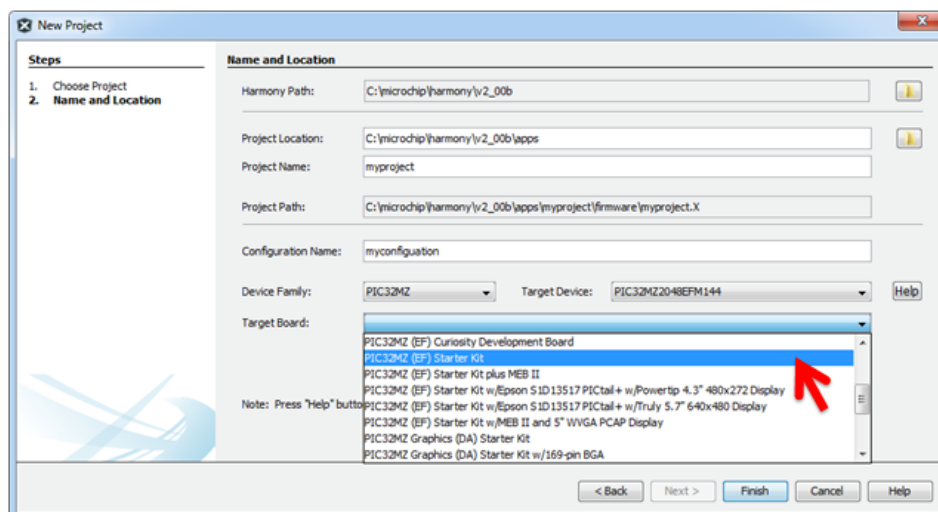
### Using a BSP

This section provides information on using a MPLAB Harmony BSP.

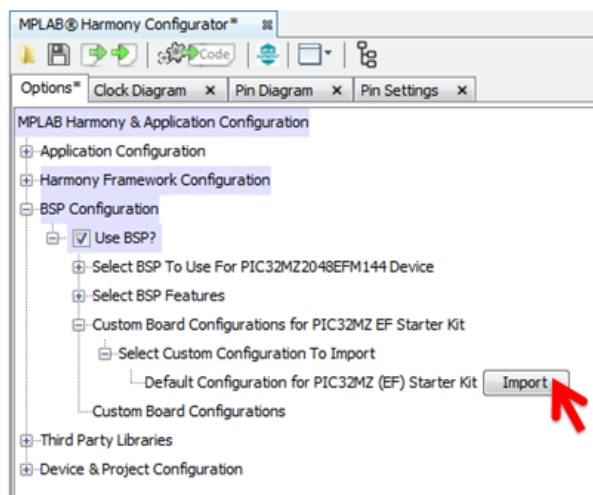
### Description

To use a BSP, select the desired BSP from the list of available BSPs in the MPLAB X IDE New Project Wizard when creating a new project, as

shown in the following figure, and complete the New Project Wizard.



The default board configuration settings will have been applied when the project is created, even before clicking the Generate Code button. You may then make modifications to the current configuration as desired, or load a custom board configuration, or reload the default BSP configuration by expanding *BSP Configuration > USB BSP > Custom Board Configurations for [BSP Name] > Select Custom Configuration to Import*, and then clicking the **Import** button as shown in the following figure. The imported settings will overwrite any current settings, and then be saved in the configuration's MHC file (not the BSP's board configuration file) when you click either the **Generate Code** or **Save Configuration** button.



Refer to *Volume III: MPLAB Harmony Configurator (MHC) > MPLAB Harmony Configurator User's Guide* for information on using the MHC. To utilize the BSP library functions, simply call the desired function from your application as shown in the example section within the documentation for each function in the Library Interface section of this document.

## Customizing a BSP

Describes how to customize a BSP library and create custom initial board configurations.

### Description

Customizations can be made by extending or modifying the BSP library source code or by creating custom initial board configurations, as described in the following topics.

## Extending the BSP Library Interface

Describes how to extend the BSP library interface.

### Description

The MHC will generate custom code that extends the BSP API when any of the following pin functions are selected in the Pin Settings table.

## BSP-Related Pin Functions

- LED\_AH Selects active high LED functionality that will generate custom named macros to control an LED that turns on when the pin is driven

high

- LED\_AL Selects active low LED functionality that will generate custom named macros to control an LED that turns on when the pin is driven low
- GPIO Selects general purpose IO pin functionality that will generate custom named macro constants for use with the Ports System Service functions
- GPIO\_IN Selects general purpose input pin that will generate a custom named macro to obtain the state of a general purpose input pin
- GPIO\_OUT Selects general purpose output pin functionality that will generate custom named macros to control a general purpose output pin

Select one of the BSP-related pin functions in the MHC's Pin Setting window and provide a custom name for the pin, as shown in the following figure.

Pin Number	Pin ID	Voltage Tolerance	Name	Function	Direction (TRIS)	Latch (LAT)	Open Drain (ODC)	Mode (ANSEL)	Change Notification (CNEN)
1	RG15		CUSTOM_LED	LED_AH	Out	Low	<input type="checkbox"/>	Digital	<input type="checkbox"/>
2	RA5		CUSTOM_SWITCH	SWITCH	In	Low	<input type="checkbox"/>	Digital	<input checked="" type="checkbox"/>
3	RE5		CUSTOM_GPIO	GPIO	In	Low	<input type="checkbox"/>	Digital	<input type="checkbox"/>
4	RE6		CUSTOM_GPIO_IN	GPIO_IN	In	Low	<input type="checkbox"/>	Digital	<input type="checkbox"/>
5	RE7		CUSTOM_GPIO_OUT	GPIO_OUT	Out	Low	<input type="checkbox"/>	Digital	<input type="checkbox"/>



**Note:**

Be sure to correctly configure the pin direction, initial latch value, digital-or-analog mode, and other settings.

After clicking the **Generate Code** button, the `bsp.h` file will be updated to contain macro definitions for any custom-named pins that are configured for BSP-related pin functions, as shown in the following examples.

### Example Macros for LED\_AH or LED\_AL Pin Named CUSTOM\_LED

```
#ifndef CUSTOM_LEDToggle
#define CUSTOM_LEDToggle() PLIB_PORTS_PinToggle \
    (PORTS_ID_0, PORT_CHANNEL_G, PORTS_BIT_POS_15)
#endif
#ifndef CUSTOM_LEDon
#define CUSTOM_LEDon() PLIB_PORTS_PinSet \
    (PORTS_ID_0, PORT_CHANNEL_G, PORTS_BIT_POS_15)
#endif
#ifndef CUSTOM_LEDOff
#define CUSTOM_LEDOff() PLIB_PORTS_PinClear \
    (PORTS_ID_0, PORT_CHANNEL_G, PORTS_BIT_POS_15)
#endif
#ifndef CUSTOM_LEDStateGet
#define CUSTOM_LEDStateGet() PLIB_PORTS_PinGetLatched \
    (PORTS_ID_0, PORT_CHANNEL_G, PORTS_BIT_POS_15)
#endif
#endif
```



**Note:**

The macros will be implemented to correctly set or clear the pin based on the selection of active-high or active-low LED function.

### Example Macro for SWITCH Pin Named CUSTOM\_SWITCH

```
#ifndef CUSTOM_SWITCHStateGet
#define CUSTOM_SWITCHStateGet() PLIB_PORTS_PinGet \
    (PORTS_ID_0, PORT_CHANNEL_A, PORTS_BIT_POS_5)
#endif
```

### Example Macros for GPIO\_OUT Pin Named CUSTOM\_GPIO\_OUT

```
#ifndef CUSTOM_GPIO_OUTToggle
#define CUSTOM_GPIO_OUTToggle() PLIB_PORTS_PinToggle \
    (PORTS_ID_0, PORT_CHANNEL_E, PORTS_BIT_POS_7)
#endif
#ifndef CUSTOM_GPIO_OUTOn
#define CUSTOM_GPIO_OUTOn() PLIB_PORTS_PinSet \
    (PORTS_ID_0, PORT_CHANNEL_E, PORTS_BIT_POS_7)
#endif
#ifndef CUSTOM_GPIO_OUTOff
#define CUSTOM_GPIO_OUTOff() PLIB_PORTS_PinClear \
    (PORTS_ID_0, PORT_CHANNEL_E, PORTS_BIT_POS_7)
#endif
#endif
```

```
#ifndef CUSTOM_GPIO_OUTStateGet
#define CUSTOM_GPIO_OUTStateGet() PLIB_PORTS_PinGetLatched \
(PORTS_ID_0, PORT_CHANNEL_E, PORTS_BIT_POS_7)
#endif
#ifndef CUSTOM_GPIO_OUTStateSet
#define CUSTOM_GPIO_OUTStateSet(Value) PLIB_PORTS_PinWrite \
(PORTS_ID_0, PORT_CHANNEL_E, PORTS_BIT_POS_7, Value)
#endif
```

### Example Macro for GPIO\_IN Pin Named CUSTOM\_GPIO\_IN

```
#ifndef CUSTOM_GPIO_INStateGet
#define CUSTOM_GPIO_INStateGet() PLIB_PORTS_PinGet \
(PORTS_ID_0, PORT_CHANNEL_E, PORTS_BIT_POS_6)
#endif
```

### Example Macros for GPIO Pin Named CUSTOM\_GPIO

```
#define CUSTOM_GPIO_PORT PORT_CHANNEL_E
#define CUSTOM_GPIO_PIN PORTS_BIT_POS_5
#define CUSTOM_GPIO_PIN_MASK (0x1 << 5)
```



**Note:**

Custom GPIO pin name macros are intended for use with the PORTS PLIB or System Service library interface functions. Refer to *Volume IV: MPLAB Harmony Framework Reference > Peripheral Libraries Help > Ports Peripheral Library* or *Volume IV: MPLAB Harmony Framework Reference > System Service Libraries Help > Ports System Service Library* for information on these functions.

The extended BSP interface macros can be overridden by defining them in the project's `system_config.h` header, if desired.



**Note:**

Current versions of MPLAB Harmony only define these macros in the `system_config.h` file, so they may be overridden by redefining them there.

## Modifying the BSP Library Source Code

Describes how to modify the BSP library source code.

### Description

The `bsp.h` and `bsp.c` source code files are generated by the MHC directly into the project's current configuration folder as described in the [Introduction](#). Therefore, these files can be directly edited as required to support the configuration. The MHC will show a “diff” tool window when generating updates to the configuration, allowing you to merge the newly generated code with the existing code.

## Creating Custom Initial Board Configurations

Describes how to create custom initial board configurations.

### Description

An initial board configuration is a MHC (\*.mhc) file that has the same format and contains the same sort of information (about configuration selections made using the MHC graphical user interface) as the MHC file in which a project configuration is stored. The difference is that it has been added to the MHC's list of initial board configurations for an associated BSP library within an installation of MPLAB Harmony.

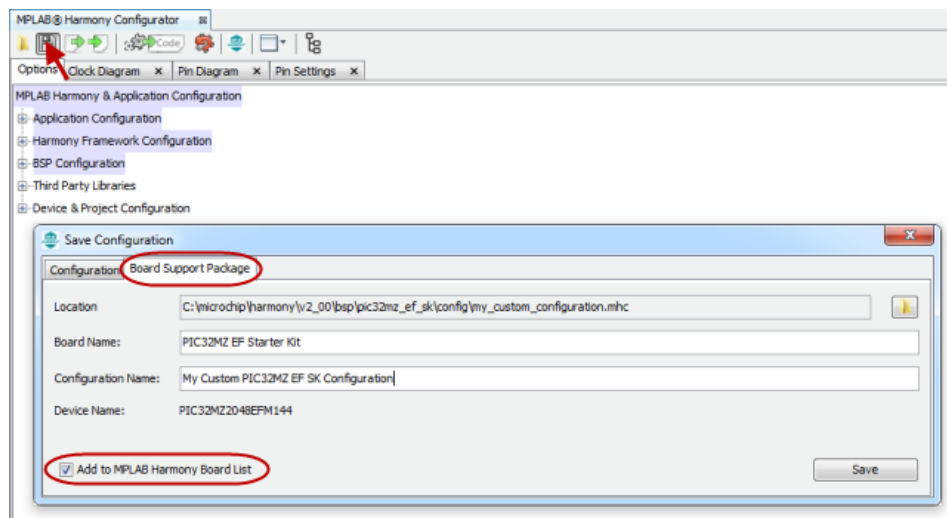
The easiest way to create a custom initial board configuration is to create a temporary MPLAB Harmony project based upon an existing BSP, and import and customize any desired configuration settings from existing project configurations. Refer to *Volume III: MPLAB Harmony Configurator (MHC) > MPLAB Harmony Configurator User's Guide* for information on how to create new projects using the MHC. Refer to *Volume III: MPLAB Harmony Configurator (MHC) > MPLAB Harmony Configurator User's Guide > Importing and Exporting Data* for information on how to import settings from other configurations.



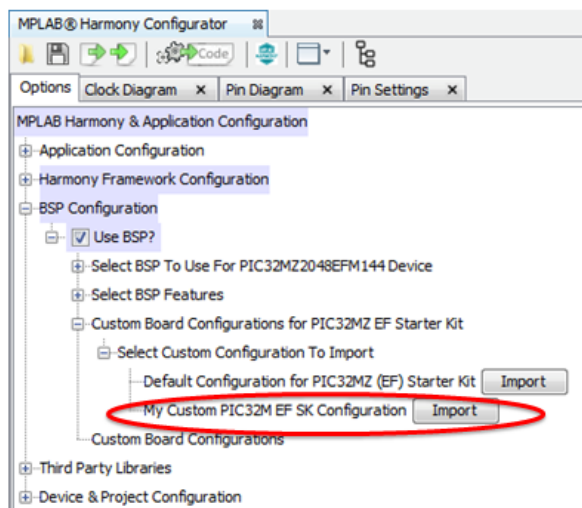
**Note:**

While it is possible to import or select application settings in an initial board configuration, it is not usually recommended, as the file may be used to initialize configurations for various different applications.

Once you have imported and/or selected all of the desired settings, save the configuration as an initial board configuration by choosing the Board Support Package tab, instead of the default Configurations tab, as shown in the following figure.



To associate the initial board configuration with a specific BSP library, be sure to save it in the `config` folder of the BSP for which it was created. For example, `<install-dir>/bsp/pic32mz_ef_sk/config/`, as shown in the previous example, and be sure to select **Add to MPLAB Harmony Board List** so that the MHC will display it as a Custom Board Configuration for the associated BSP library. Give it a name that is descriptive of how it configures the board, so that you can easily understand why you might choose it over the default (or other) initial board configuration. Once saved, close and reopen the MHC and the new initial board Configuration Name will appear in the list of Custom Board Configurations for the associated BSP, as shown in the following image.



## Board Support Packages

Provides information on the individual board support packages included in your installation of MPLAB Harmony.

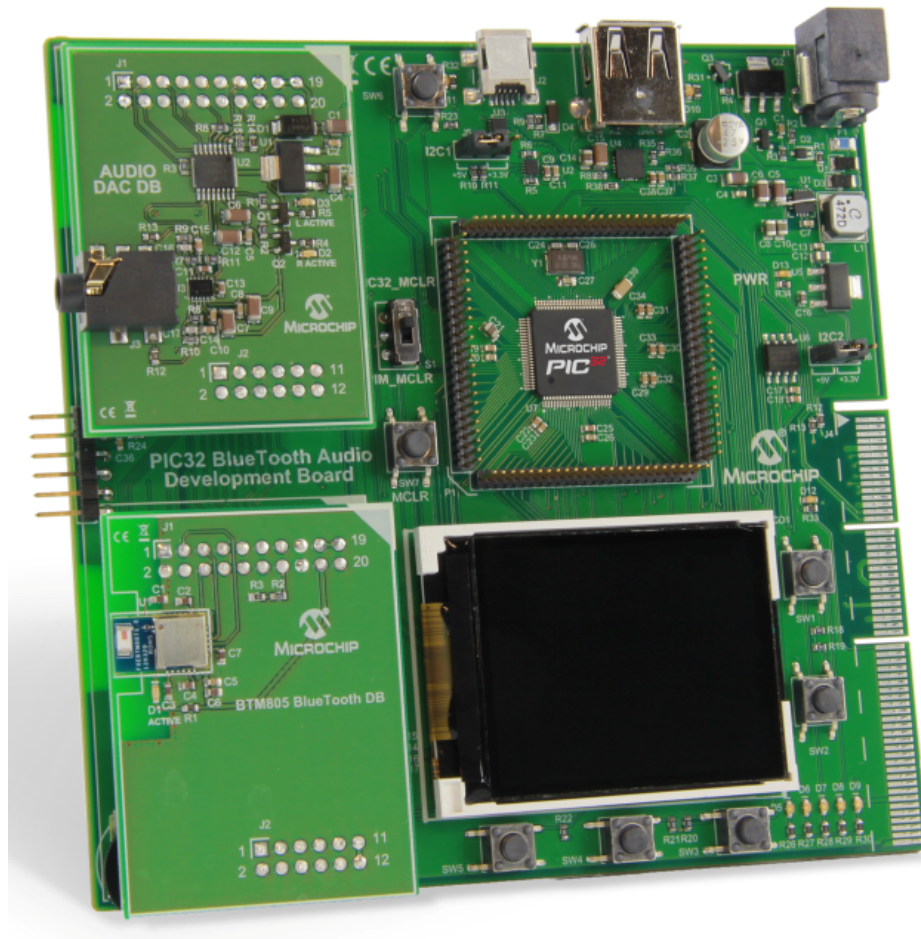
### ***bt\_audio\_dk***

PIC32 Bluetooth Audio Development Kit BSP.

#### **Description**

This BSP is intended for the PIC32 Bluetooth Audio Development Kit.

The following figure illustrates the hardware configuration.



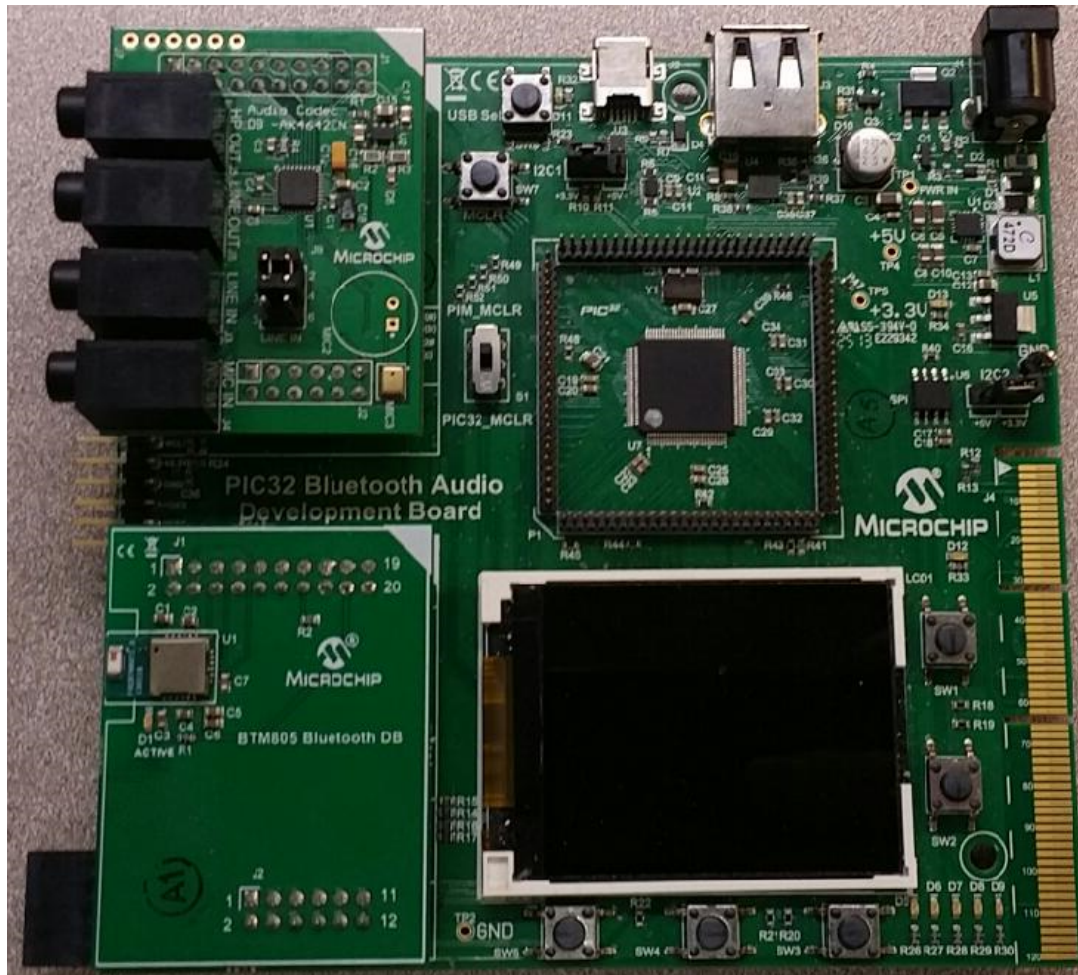
### ***bt\_audio\_dk+4642***

PIC32 Bluetooth Audio Development Kit with the AK4642 Audio Codec BSP.

#### **Description**

This BSP is intended for the PIC32 Bluetooth Audio Development Kit with the AK4642 Audio Codec. The following figure illustrates the hardware configuration.



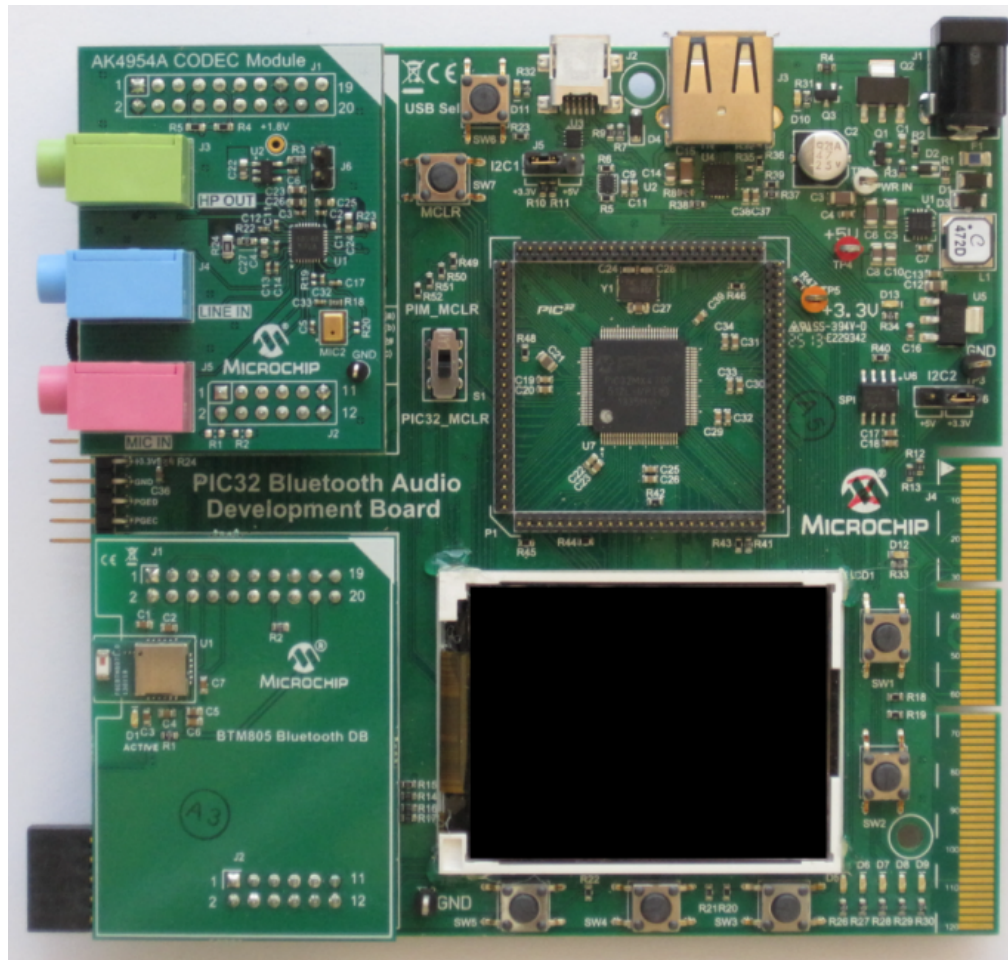


### **bt\_audio\_dk+ak4954**

PIC32 Bluetooth Audio Development Kit with the AK4954 Audio Codec BSP.

### **Description**

This BSP is intended for the PIC32 Bluetooth Audio Development Kit with the AK4954 Audio Codec. The following figure illustrates the hardware configuration.



### **bt\_audio\_dk+ak7755**

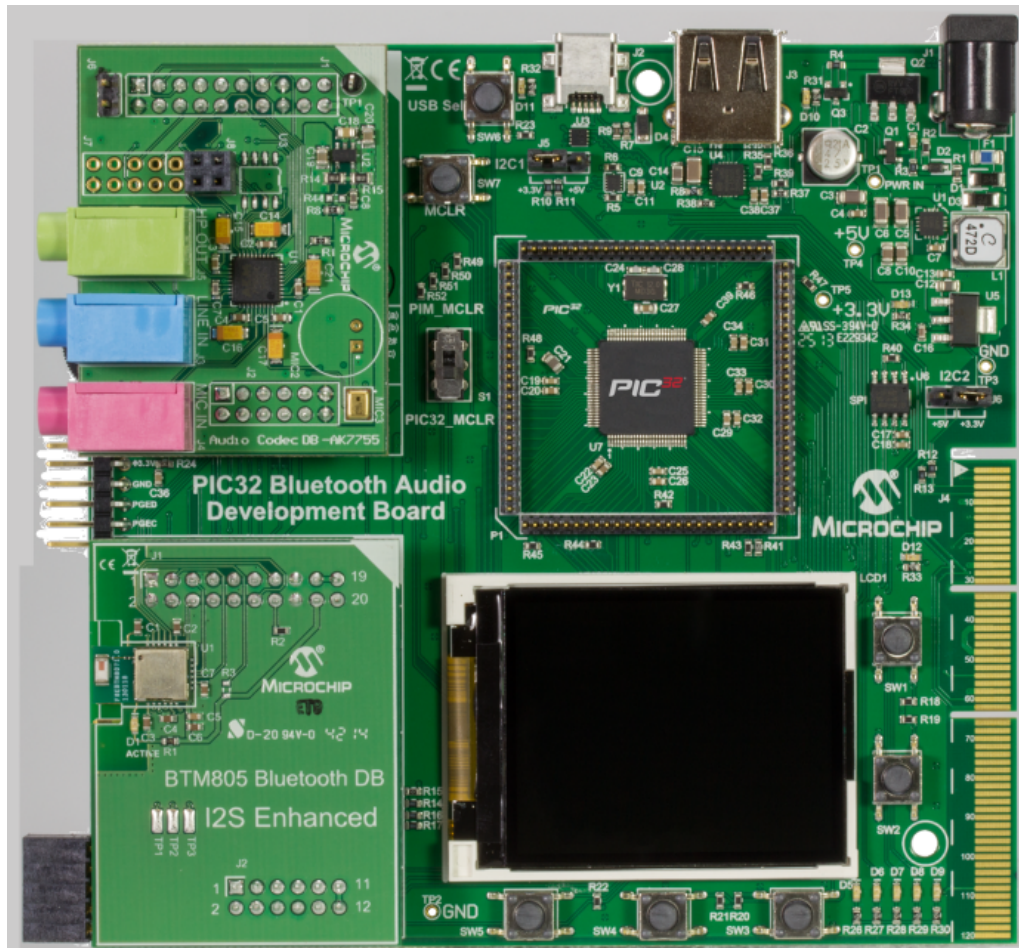
PIC32 Bluetooth Audio Development Kit with the AK7755 Audio Codec BSP.

### **Description**

This BSP is intended for the PIC32 Bluetooth Audio Development Kit with the AK7755 Audio Codec.

The following figure illustrates the hardware configuration.





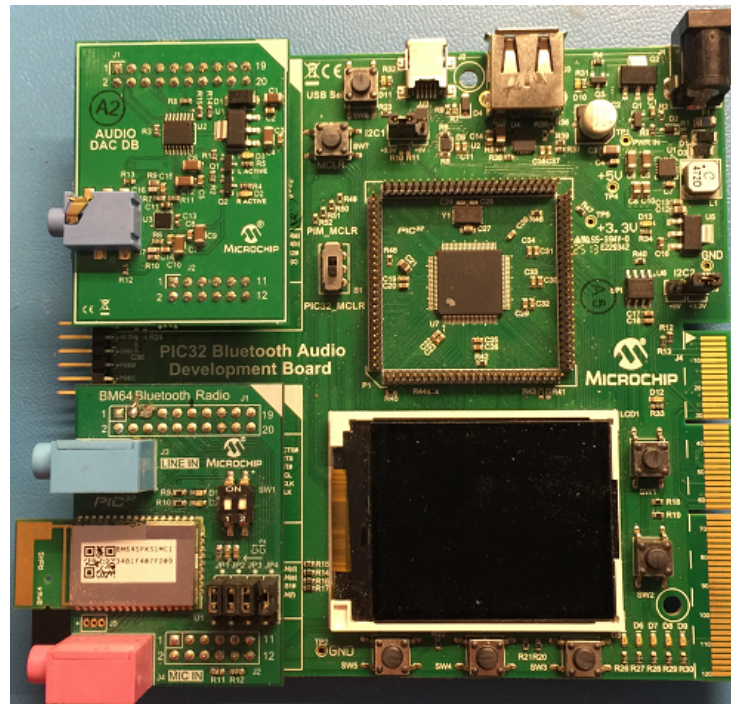
### ***bt\_audio\_dk+bm64***

PIC32 Bluetooth Audio Development Kit with the BM64 Bluetooth Module BSP.

### **Description**

This BSP is intended for the [PIC32 Bluetooth Audio Development Kit](#) with the BM64 Bluetooth Module Daughter Board and the default PIC32 Audio DAC Daughter Board.

The following figure illustrates the hardware configuration.



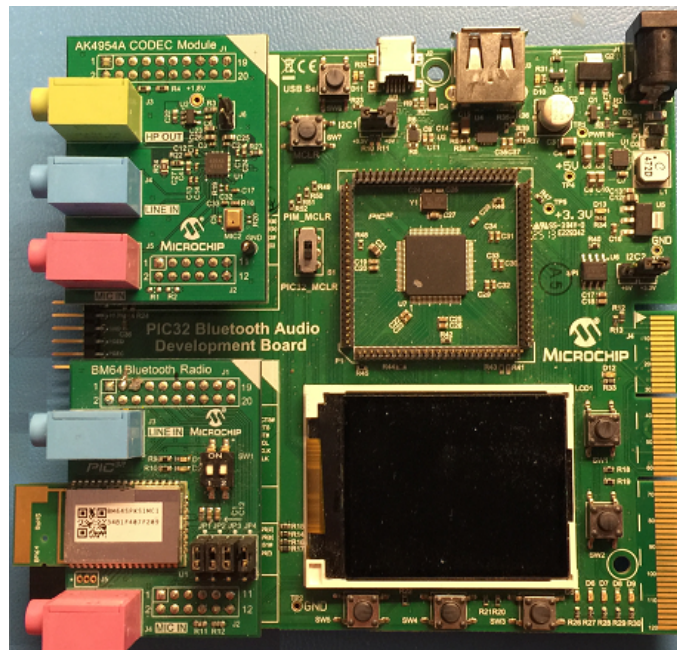
### **bt\_audio\_dk+bm64+ak4954**

PIC32 Bluetooth Audio Development Kit with the BM64 Bluetooth Module and the AK4954 Audio Codec BSP.

#### **Description**

This BSP is intended for the [PIC32 Bluetooth Audio Development Kit](#) with the BM64 Bluetooth Module Daughter Board and the PIC32 Audio Codec Daughter Board - AK4954A.

The following figure illustrates the hardware configuration.

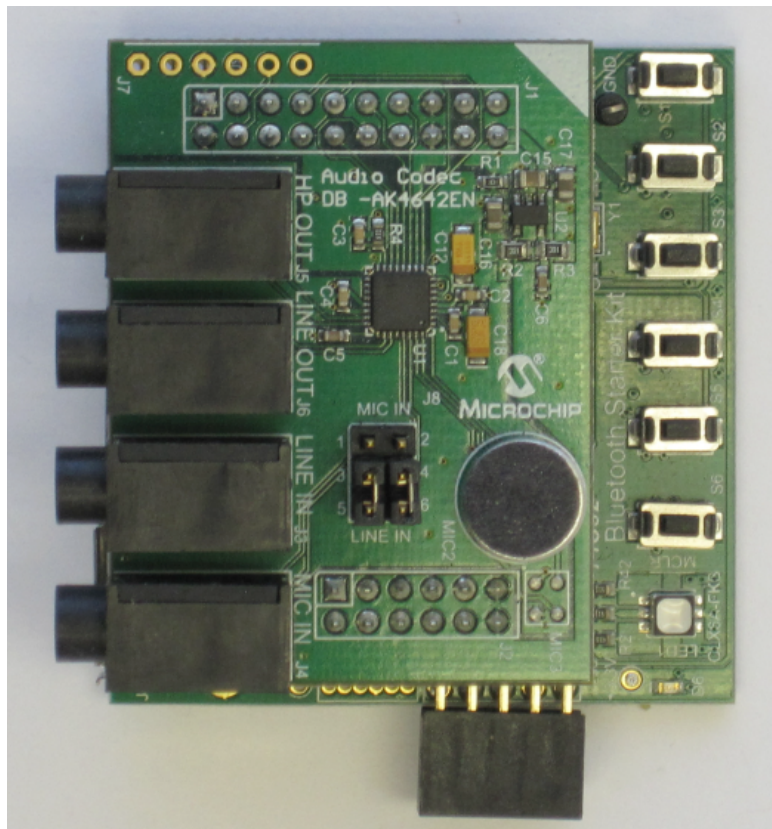


### **bt\_sk+4642**

PIC32 Bluetooth Starter Kit with the AK4642 Audio Codec BSP.

## Description

This BSP is intended for the PIC32 Bluetooth Audio Development Kit with the AK4642 Audio Codec. The following figure illustrates the hardware configuration.



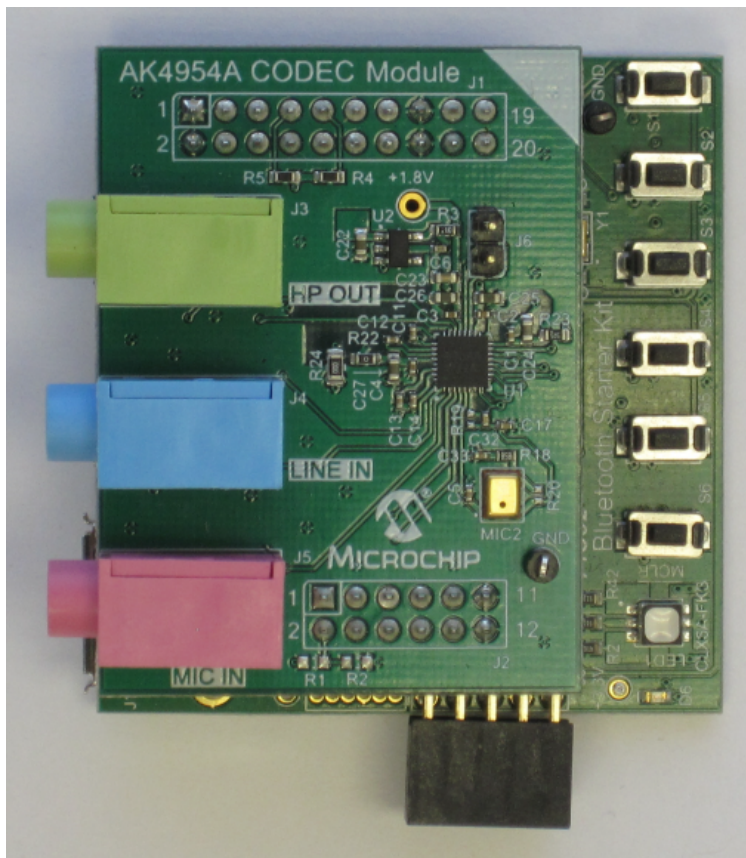
## ***bt\_sk+4954***

PIC32 Bluetooth Starter Kit with the AK4954 Audio Codec BSP

## Description

This BSP is intended for the PIC32 Bluetooth Audio Development Kit with the AK4954 Audio Codec. The following figure illustrates the hardware configuration.





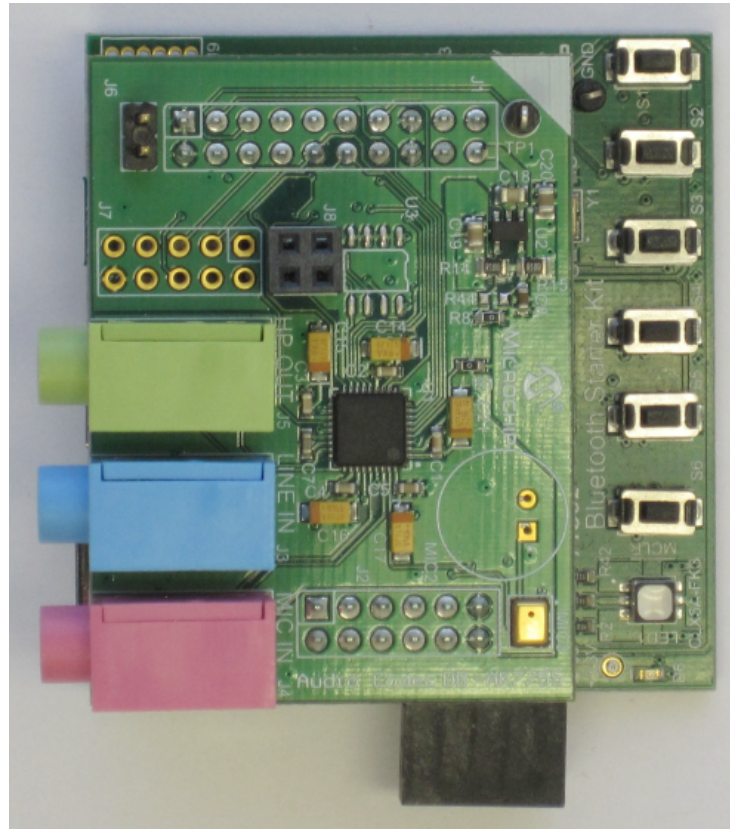
### ***bt\_sk+ak7755***

PIC32 Bluetooth Starter Kit with the AK7755 Audio Codec BSP.

### **Description**

This BSP is intended for the PIC32 Bluetooth Audio Development Kit with the AK7755 Audio Codec.

The following figure illustrates the hardware configuration.



### chipkit\_wf32

chipKIT™ WF32™ Wi-Fi Development Board BSP.

#### Description

This BSP is intended for the chipKIT™ WF32™ Wi-Fi Development Board.



**Note:**

If a USB Host application is used, the board *will not* be able to power the USB device without one of the following:

Using an external power supply (9V or greater) connected to J17

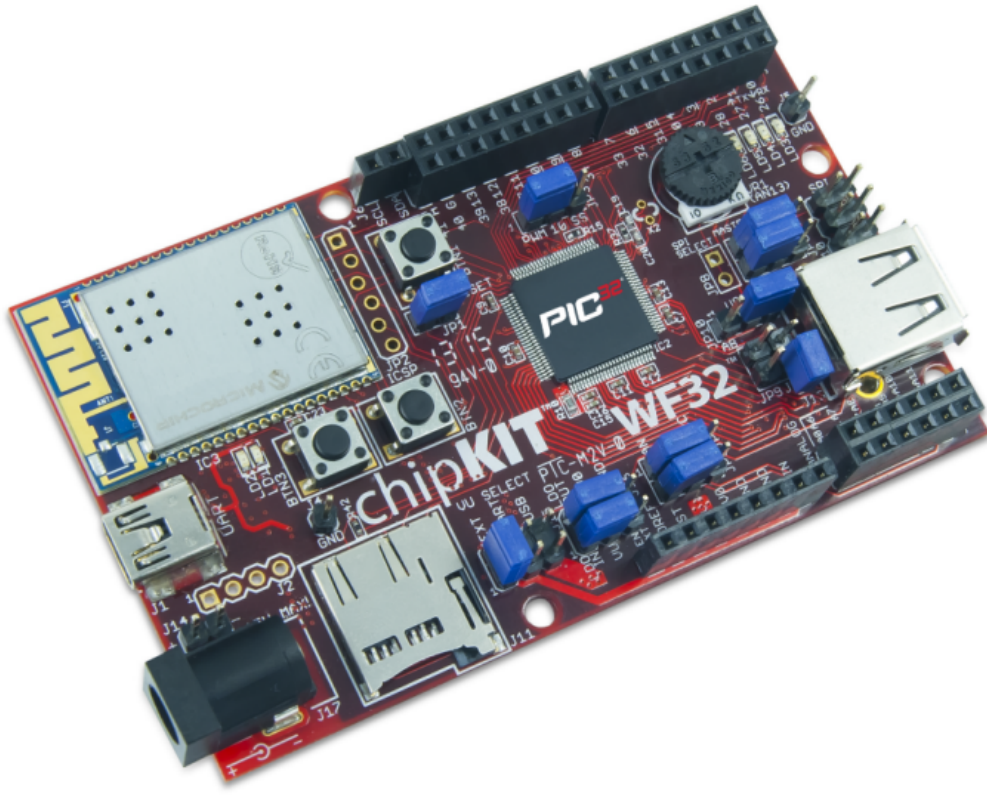
- or -

Bypassing the on-board voltage regulator by removing the jumpers on J16 and only connecting VU to 5V0.



**Warning:** Do not connect an external power supply in this configuration, or the 5V rail on the board will be supplied with the external voltage directly, which could result in damage to the board.

The following figure illustrates the hardware configuration.



## chipkit\_wifire

chipKIT™ Wi-FIRE Development Board BSP.

### Description

This BSP is intended for the chipKIT Wi-FIRE Development Board.



**Note:**

If a USB Host application is used, the board *will not* be able to power the USB device without one of the following:

Using an external power supply (9V or greater) connected to J15

- or -

Bypassing the on-board voltage regulator by removing the jumpers on J17 and only connecting VU to 5V0.



**Warning:** Do not connect an external power supply in this configuration, or the 5V rail on the board will be supplied with the external voltage directly, which could result in damage to the board.

The following figure illustrates the hardware configuration.



### ***pic32\_gdb\_ef***

PIC32 Graphics Discovery Development Board BSP.

#### **Description**

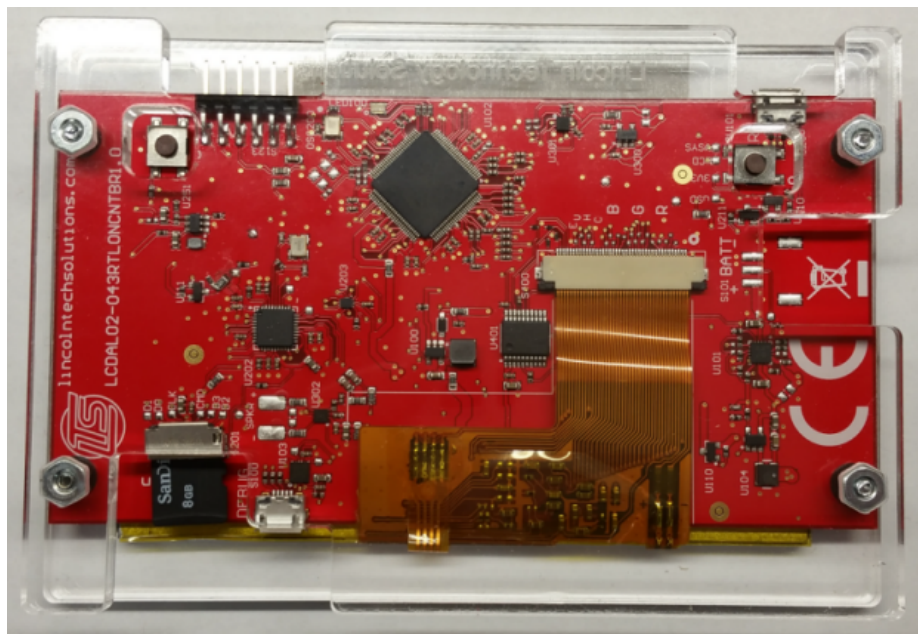
This BSP is intended for the the PIC32 Graphics Discovery Development Board and the PIC32MZ EF Starter Kit.



**Note:**

Please contact your local Microchip sales office for information on obtaining this development board.

The following figure illustrates the hardware configuration.



### ***pic32mk\_gp\_db***

PIC32MK GP Development Board BSP

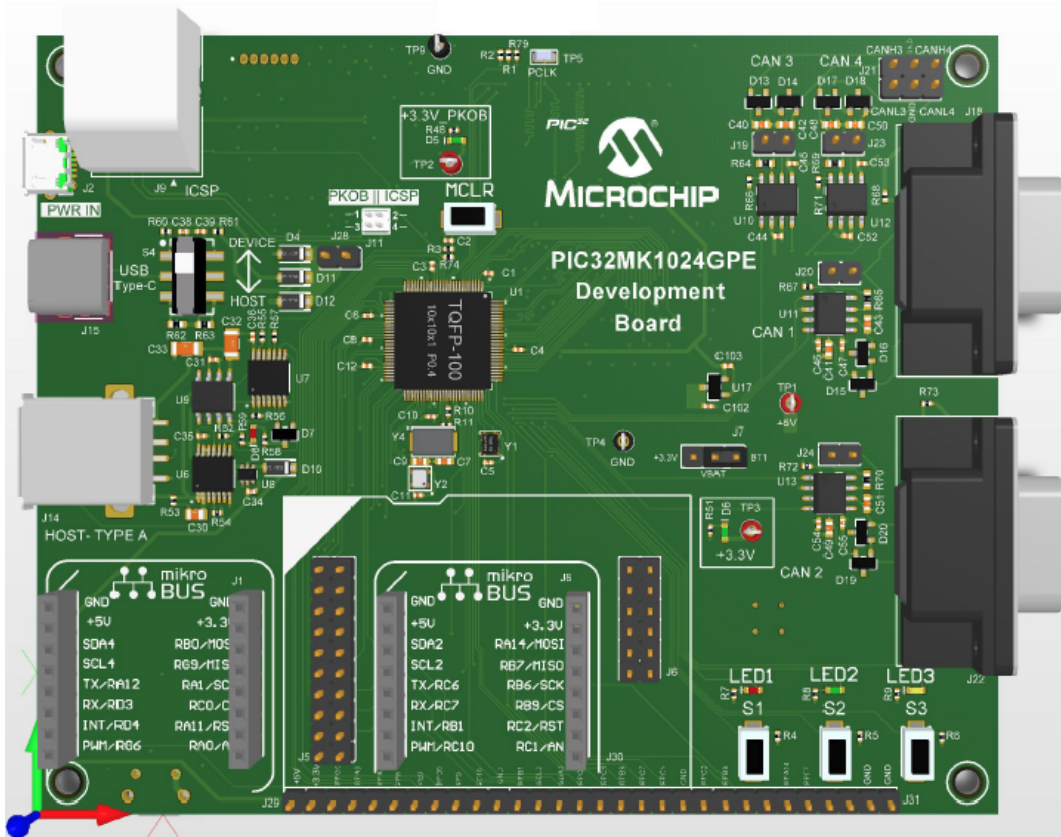
#### **Description**

This BSP is intended for the PIC32MK GP Development Board.



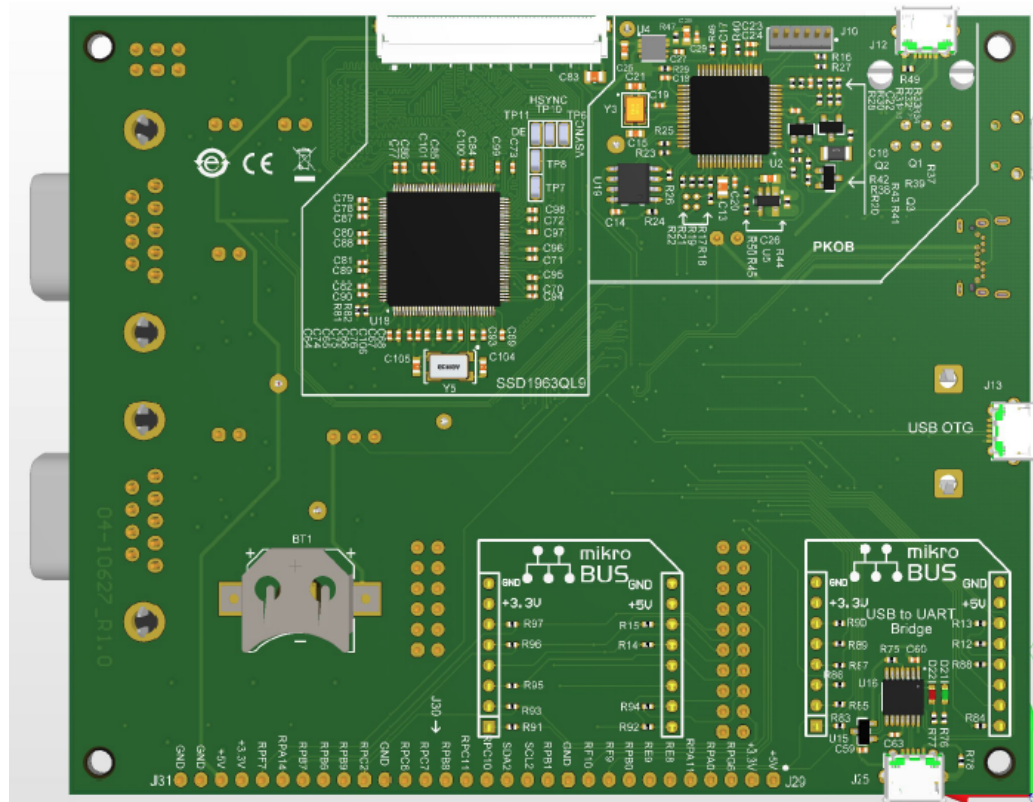
The following figures illustrate the hardware configuration.

### Top View





## Bottom View

***pic32mk\_gp\_db+wqvga\_mxt***

PIC32MK GP Development Board (with SSD1963 Graphics Controller) and a 480x272 WQVGA maXTouch Display Module BSP.

**Description**

This BSP is intended for the PIC32MK GP Development Board (with SSD1963 Graphics Controller) connected to a High-Performance 4.3" WQVGA Display Module with maXTouch.



**Note:** The display plus an interposer for use with the Multimedia Expansion Board II (MEBII) can be ordered from Microchip using the part number: AC320005-4.

The following figure illustrates the hardware configuration.



### *pic32mk\_gp\_db+vvga\_mxt*

PIC32MK GP Development Board (with SSD1963 Graphics Controller) and a High-Performance WVGA Display Module with maXTouch.

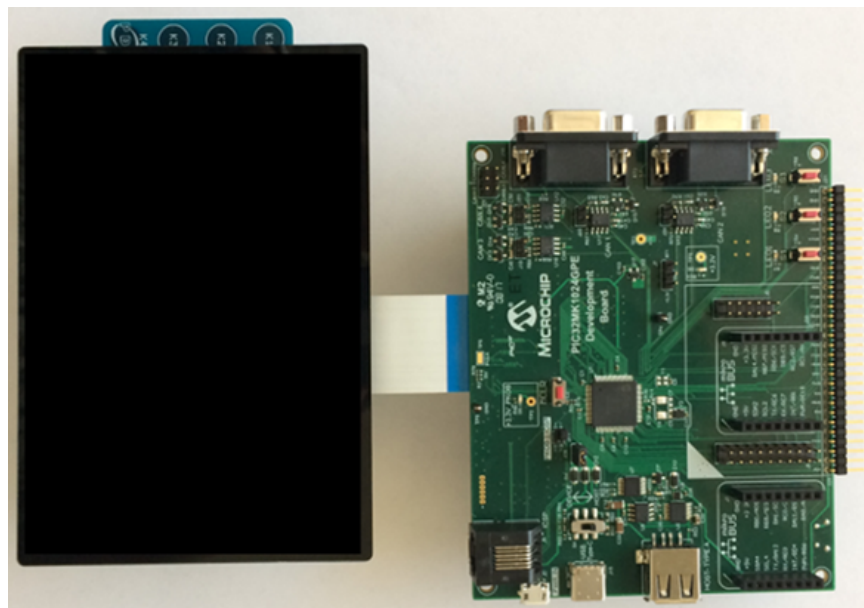
#### Description

This BSP is intended for the PIC32MK GP Development Board (with SSD1963 Graphics Controller) connected to a High-Performance WVGA Display Module with maXTouch.



**Note:** The display plus an interposer for use with the Multimedia Expansion Board II (MEB II) can be ordered from Microchip using the part number: AC320005-5.

The following figure illustrates the hardware configuration.

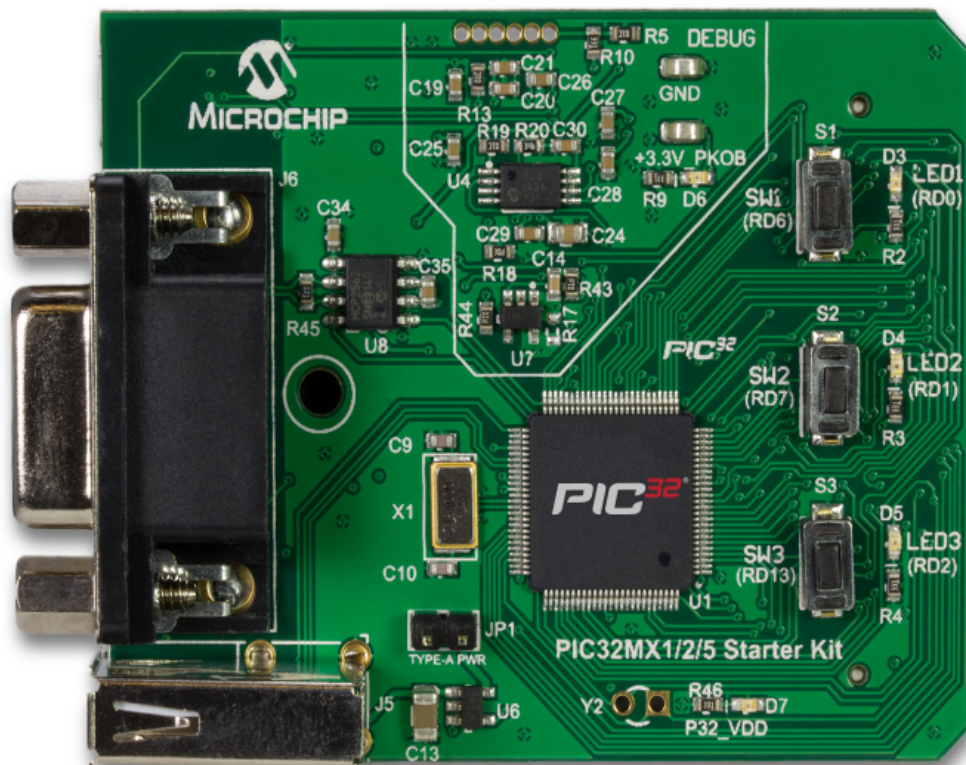


### *pic32mx\_125\_sk*

PIC32MX1/2/5 Starter Kit BSP.

## Description

This BSP is intended for the PIC32MX1/2/5 Starter Kit.  
The following figure illustrates the hardware configuration.

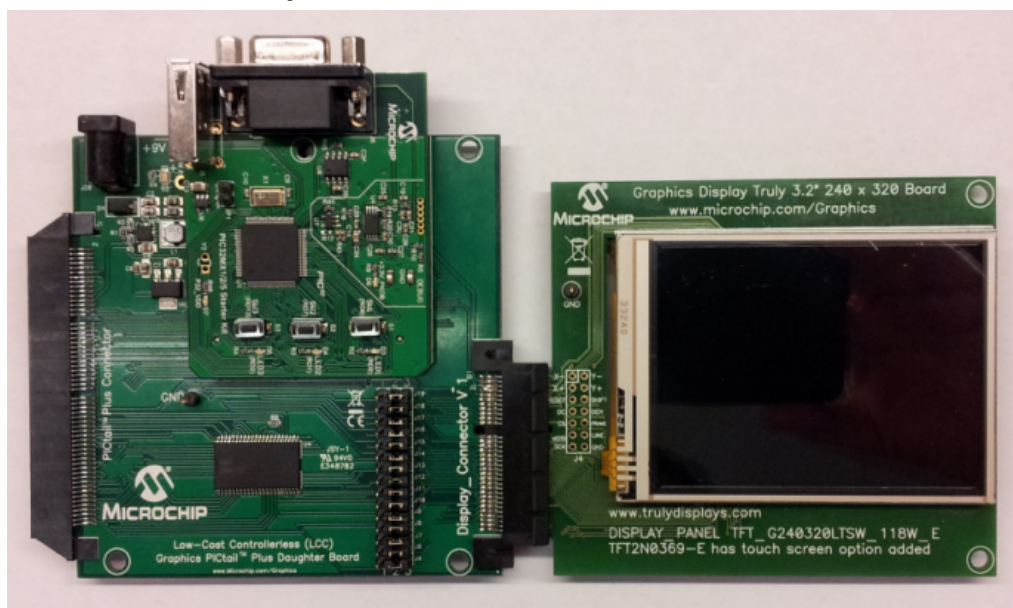


## *pic32mx\_125\_sk+lcc\_pictail+qvga*

PIC32MX1/2/5 Starter Kit plus the Low-Cost Controllerless (LCC) Graphics PICTail Plus Daughter Board with the Graphics Display Truly 3.2" 320x240 Board BSP.

## Description

This BSP is intended for the Low-Cost Controllerless (LCC) Graphics PICTail Plus Daughter Board with the Graphics Display Truly 3.2" 320x240 Board connected to the PIC32MX1/2/5 Starter Kit.  
The following figure illustrates the hardware configuration.





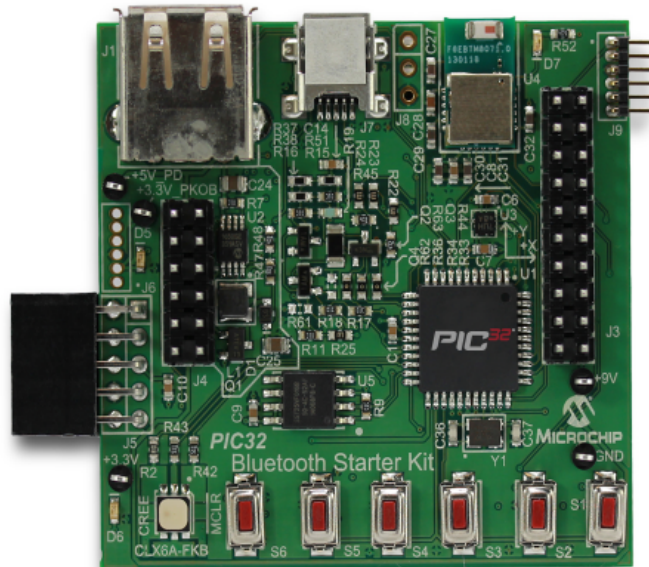
## ***pic32mx\_bt\_sk***

PIC32 Bluetooth Starter Kit BSP.

### **Description**

This BSP is intended for the PIC32 Bluetooth Starter Kit.

The following figure illustrates the hardware configuration.



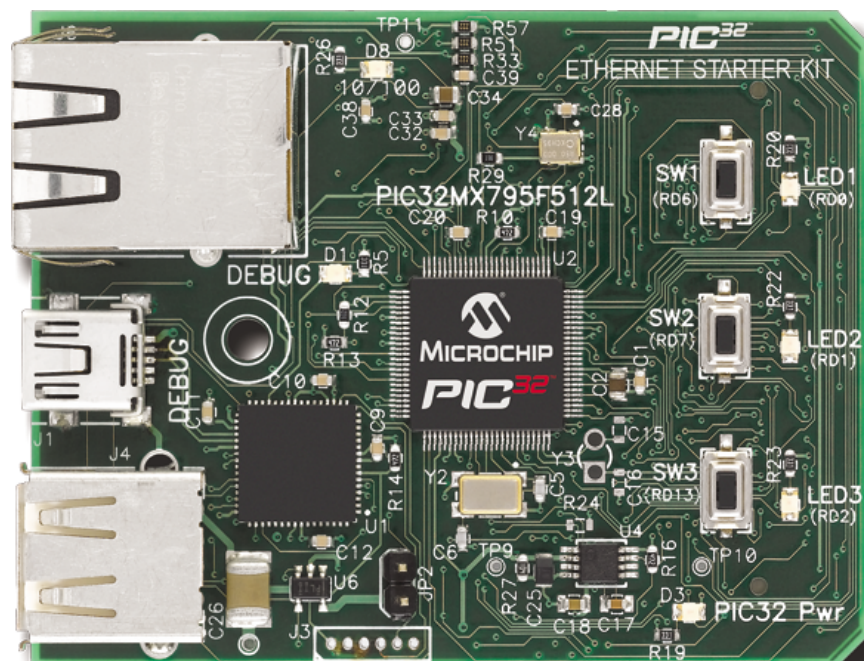
## ***pic32mx\_eth\_sk***

PIC32 Ethernet Starter Kit BSP.

### **Description**

This BSP is intended for the PIC32 Ethernet Starter Kit.

The following figure illustrates the hardware configuration.



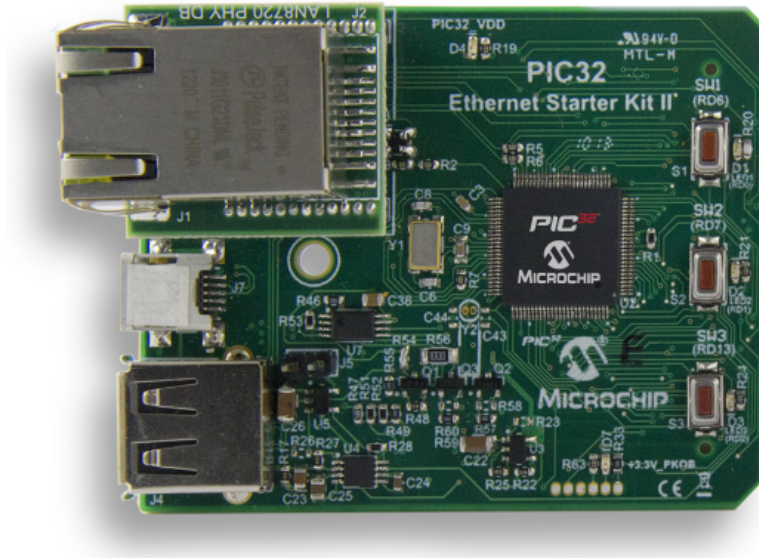
### ***pic32mx\_eth\_sk2***

PIC32 Ethernet Starter Kit II BSP.

#### **Description**

This BSP is intended for the PIC32 Ethernet Starter Kit II.

The following figure illustrates the hardware configuration.



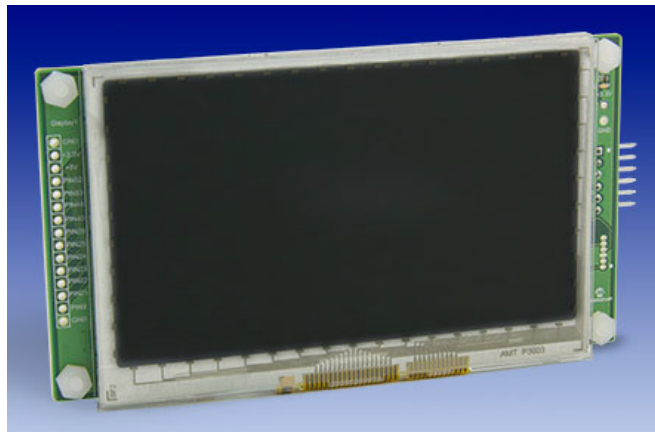
### ***pic32mx\_pcap\_db***

PIC32 GUI Development Board with Projected Capacitive Touch BSP.

#### **Description**

This BSP is intended for the PIC32 GUI Development Board with Projected Capacitive Touch.

The following figure illustrates the hardware configuration.



### ***pic32mx\_usb\_digital\_audio\_ab***

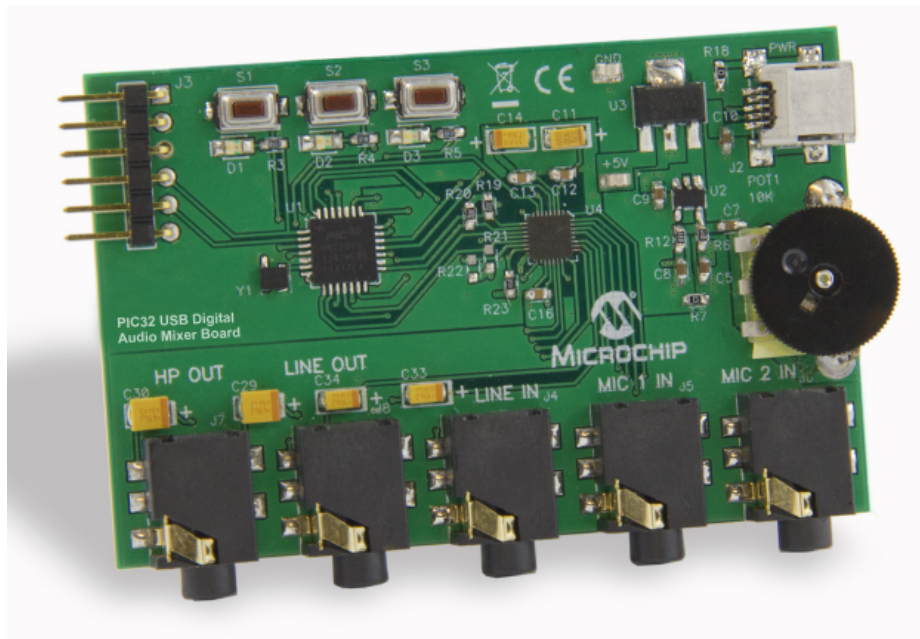
PIC32 USB Digital Audio Accessory Board BSP.

#### **Description**

This BSP is intended for the PIC32 USB Digital Audio Accessory Board.

The following figure illustrates the hardware configuration.





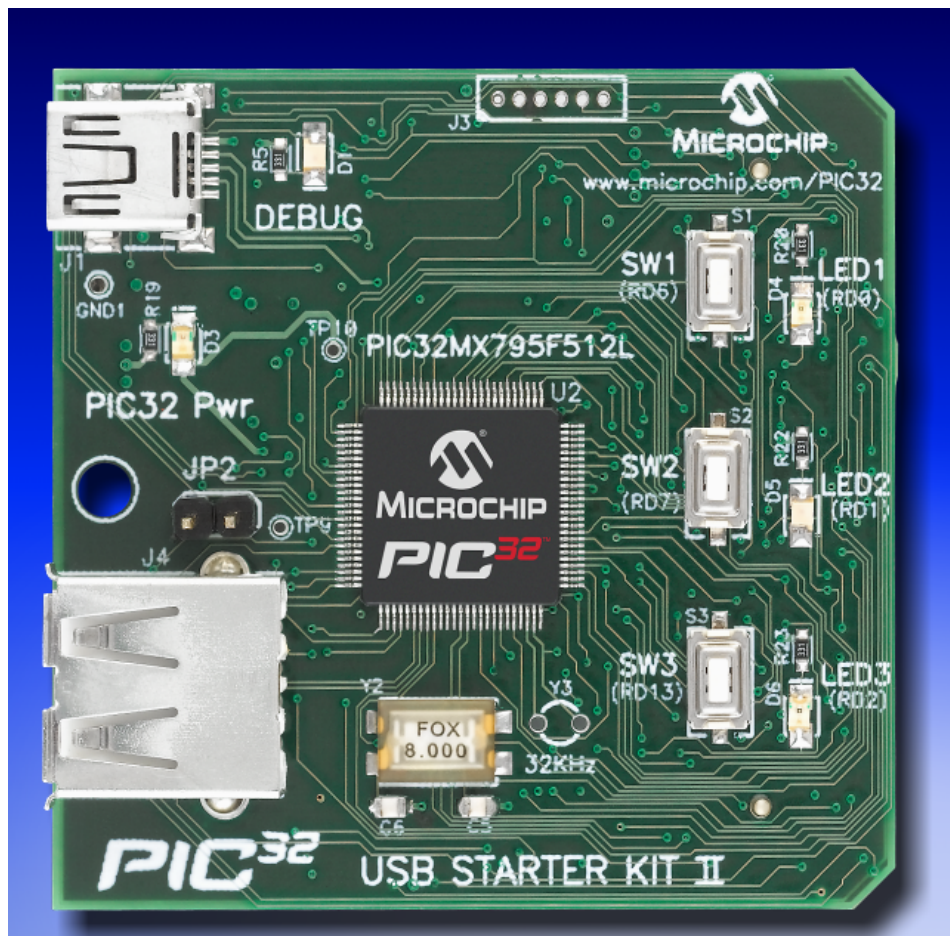
### ***pic32mx\_usb\_sk2***

PIC32 USB Starter Kit II BSP.

#### **Description**

This BSP is intended for the PIC32 USB Starter Kit II.

The following figure illustrates the hardware configuration.



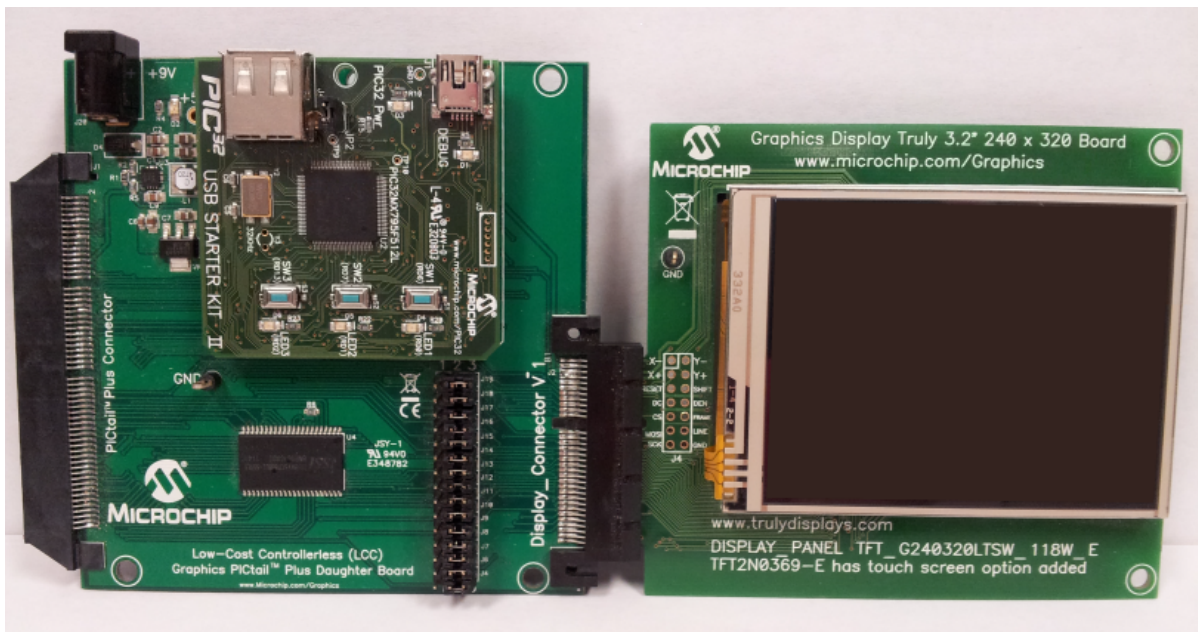
### ***pic32mx\_usb\_sk2+lcc\_pictail+qvga***

PIC32 USB Starter Kit II plus the Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board with the Graphics Display Truly 3.2" 320x240 Board BSP.

#### **Description**

This BSP is intended for the Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board with the Graphics Display Truly 3.2" 320x240 Board connected to the PIC32 USB Starter Kit II.

The following figure illustrates the hardware configuration.



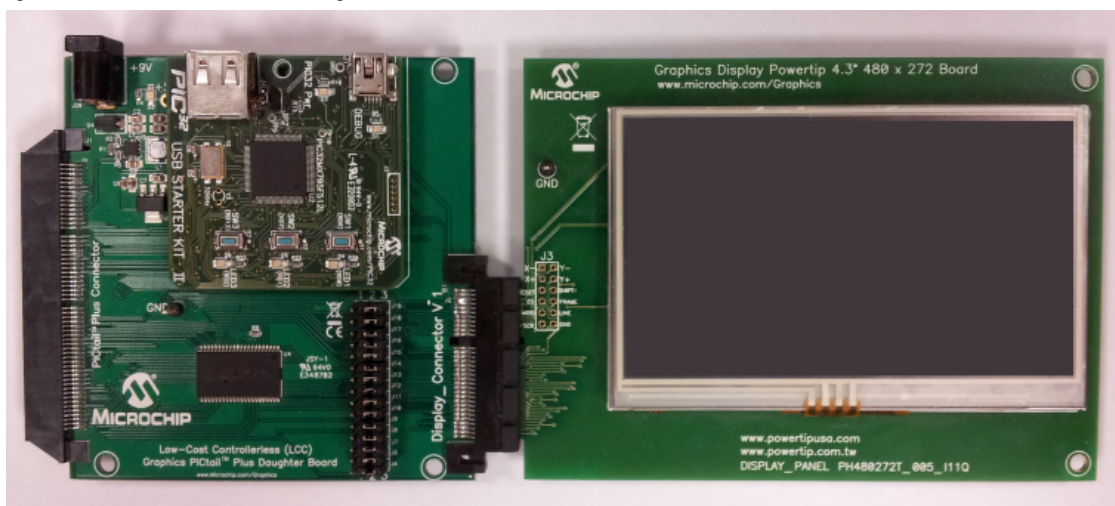
### ***pic32mx\_usb\_sk2+lcc\_pictail+wqvga***

PIC32 USB Starter Kit II plus the Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board with Graphics Display Powertip 4.3" 480x272 Board BSP.

#### **Description**

This BSP is intended for the Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board with the Graphics Display Powertip 4.3" 480x272 Board connected to the PIC32 USB Starter Kit II.

The following figure illustrates the hardware configuration.



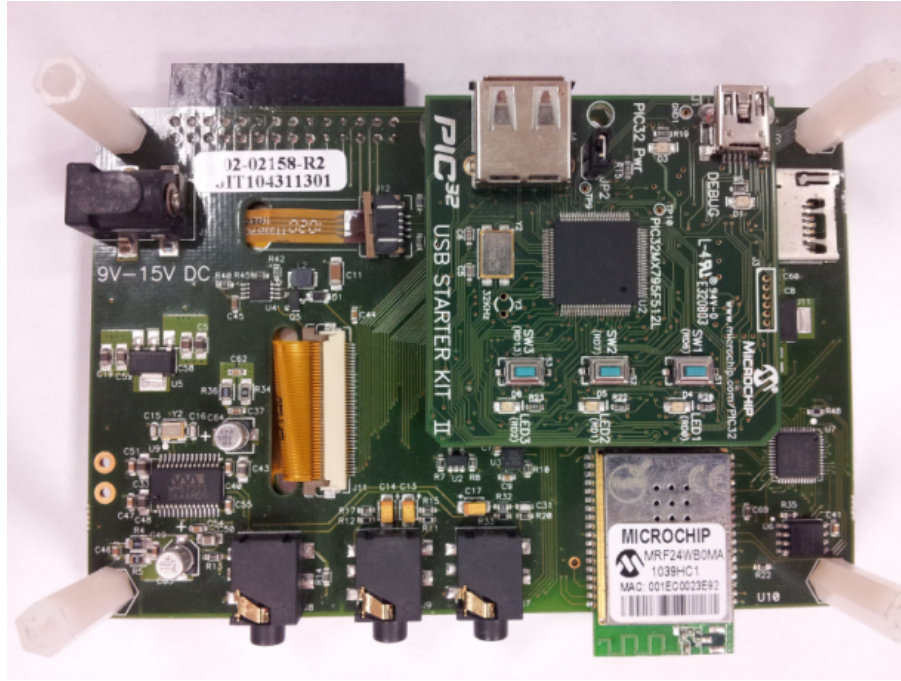


### ***pic32mx\_usb\_sk2+meb***

PIC32 USB Starter Kit II plus MEB BSP.

#### **Description**

This BSP is intended for the Multimedia Expansion Board (MEB) connected to the PIC32 USB Starter Kit II. The following figure illustrates the hardware configuration.

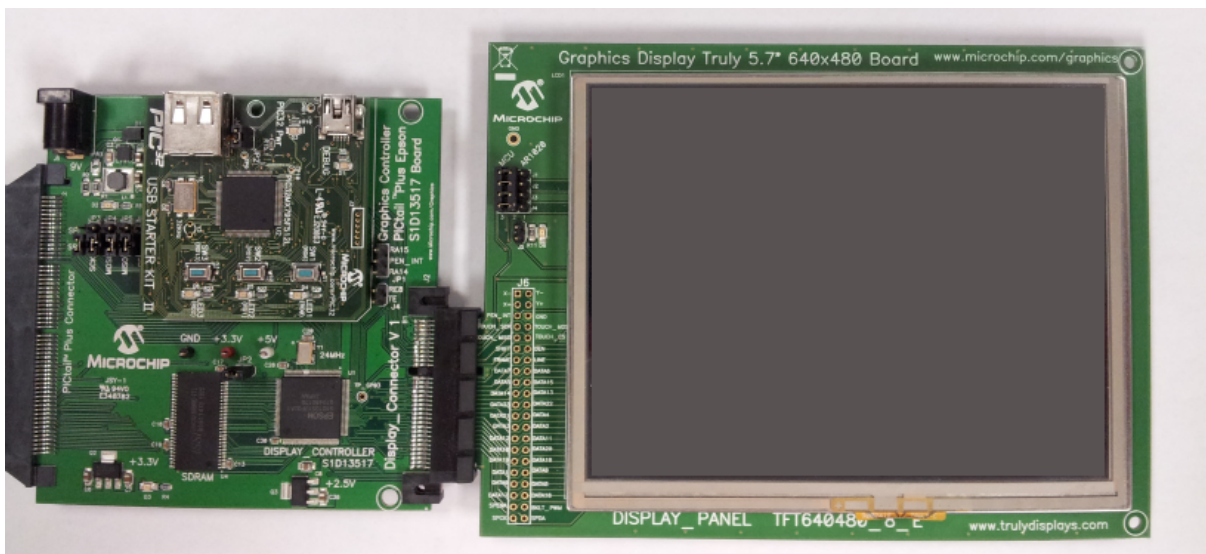


### ***pic32mx\_usb\_sk2+s1d\_pictail+vga***

PIC32 USB Starter Kit II plus the Graphics Controller Pictail Plus Epson S1D13517 Daughter Board with Graphics Display Truly 5.7" 640x480 Board BSP.

#### **Description**

This BSP is intended for the Graphics Controller Pictail Plus Epson S1D13517 Daughter Board with the Graphics Display Truly 5.7" 640x480 Board connected to the PIC32 USB Starter Kit II. The following figure illustrates the hardware configuration.





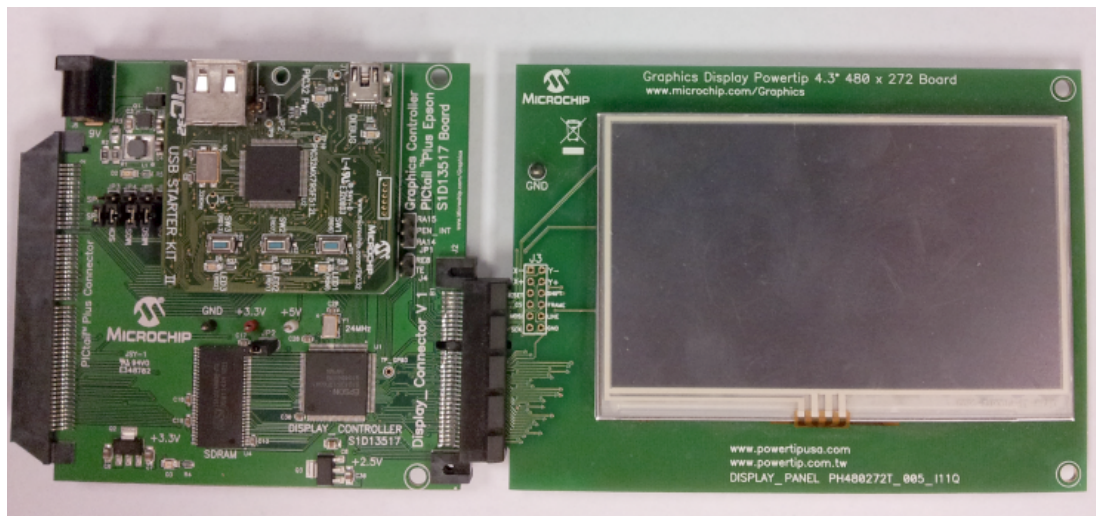
### ***pic32mx\_usb\_sk2+s1d\_pictail+wqvga***

PIC32 USB Starter Kit II plus the Graphics Controller PICtail Plus Epson S1D13517 Daughter Board with the Graphics Display Powertip 4.3" 480x272 Board BSP.

#### **Description**

This BSP is intended for the Graphics Controller PICtail Plus Epson S1D13517 Daughter Board with the Graphics Display Powertip 4.3" 480x272 Board connected to the PIC32 USB Starter Kit II.

The following figure illustrates the hardware configuration.



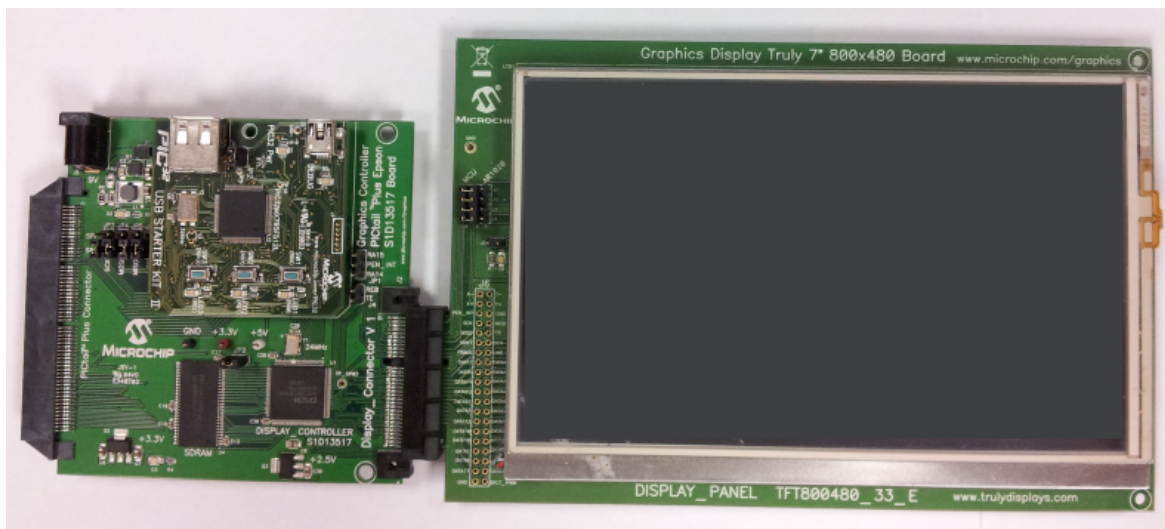
### ***pic32mx\_usb\_sk2+s1d\_pictail+wvga***

PIC32 USB Starter Kit II plus the Graphics Controller PICtail Plus Epson S1D13517 Daughter Board with Graphics Display Truly 7" 800x480 Board BSP.

#### **Description**

This BSP is intended for the Graphics Controller PICtail Plus Epson S1D13517 Daughter Board with Graphics Display Truly 7" 800x480 Board connected to the PIC32 USB Starter Kit II.

The following figure illustrates the hardware configuration.



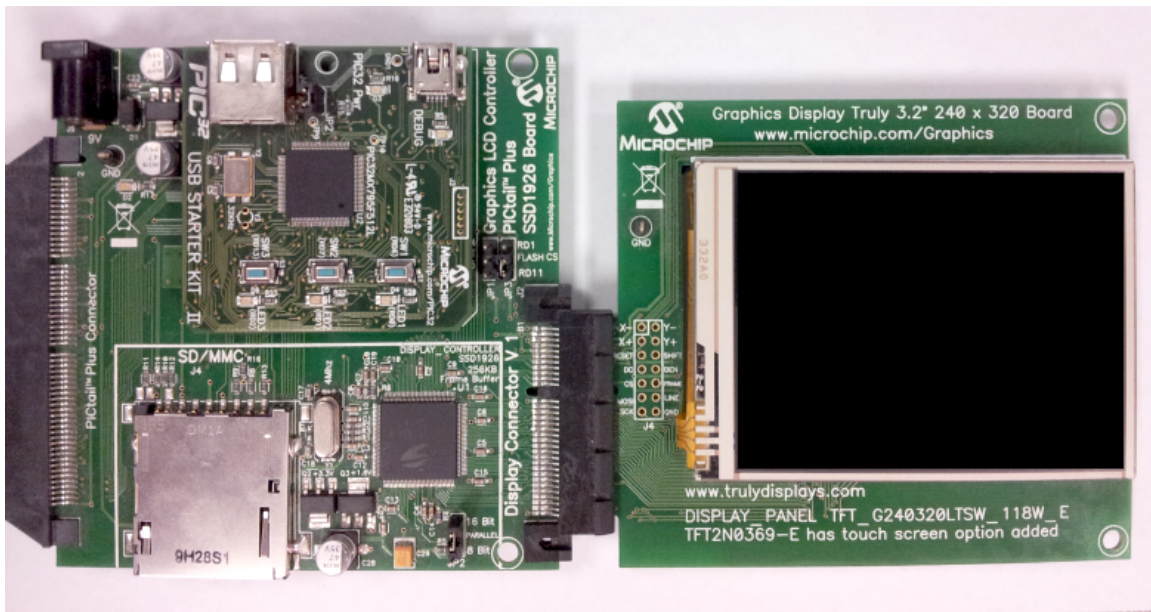
### ***pic32mx\_usb\_sk2+ssd\_pictail+qvga***

PIC32 USB Starter Kit II plus the Graphics LCD Controller PICtail Plus SSD1926 Daughter Board with Graphics Display Truly 3.2" 320x240 Board BSP.

## Description

This BSP is intended for the Graphics LCD Controller PICtail Plus SSD1926 Daughter Board with Graphics Display Truly 3.2" 320x240 Board connected to the PIC32 USB Starter Kit II.

The following figure illustrates the hardware configuration.



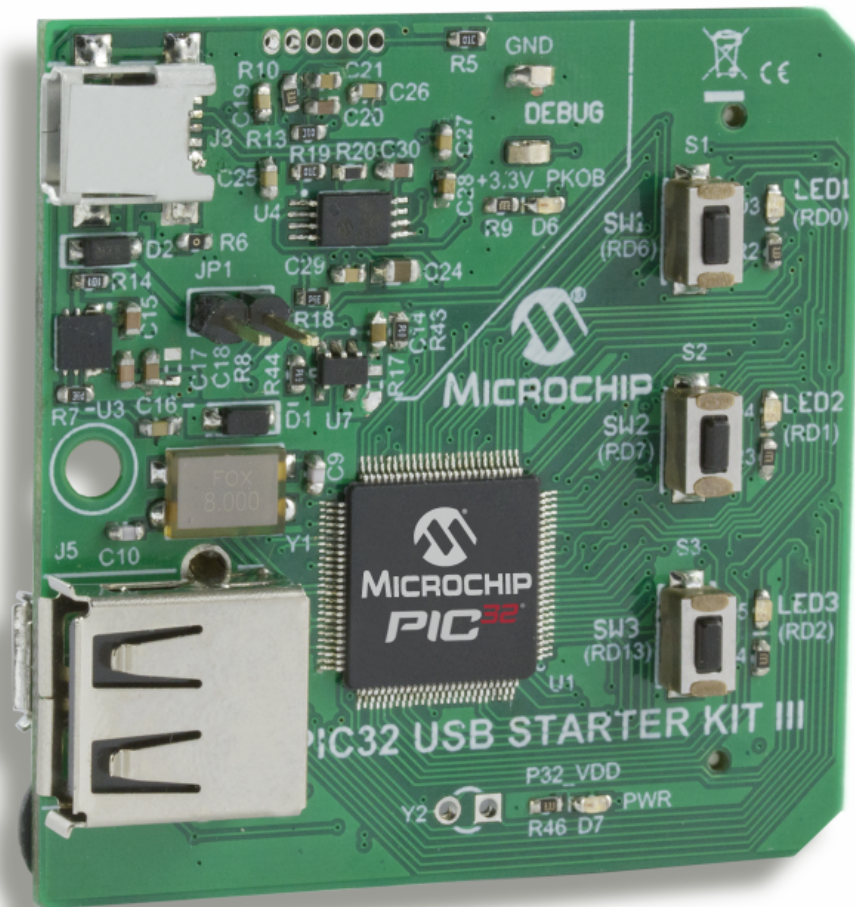
## `pic32mx_usb_sk3`

PIC32 USB Starter Kit III BSP.

## Description

This BSP is intended for the PIC32 USB Starter Kit III.

The following figure illustrates the hardware configuration.



### ***pic32mx\_usb\_sk3+lcc\_pictail+wqvga***

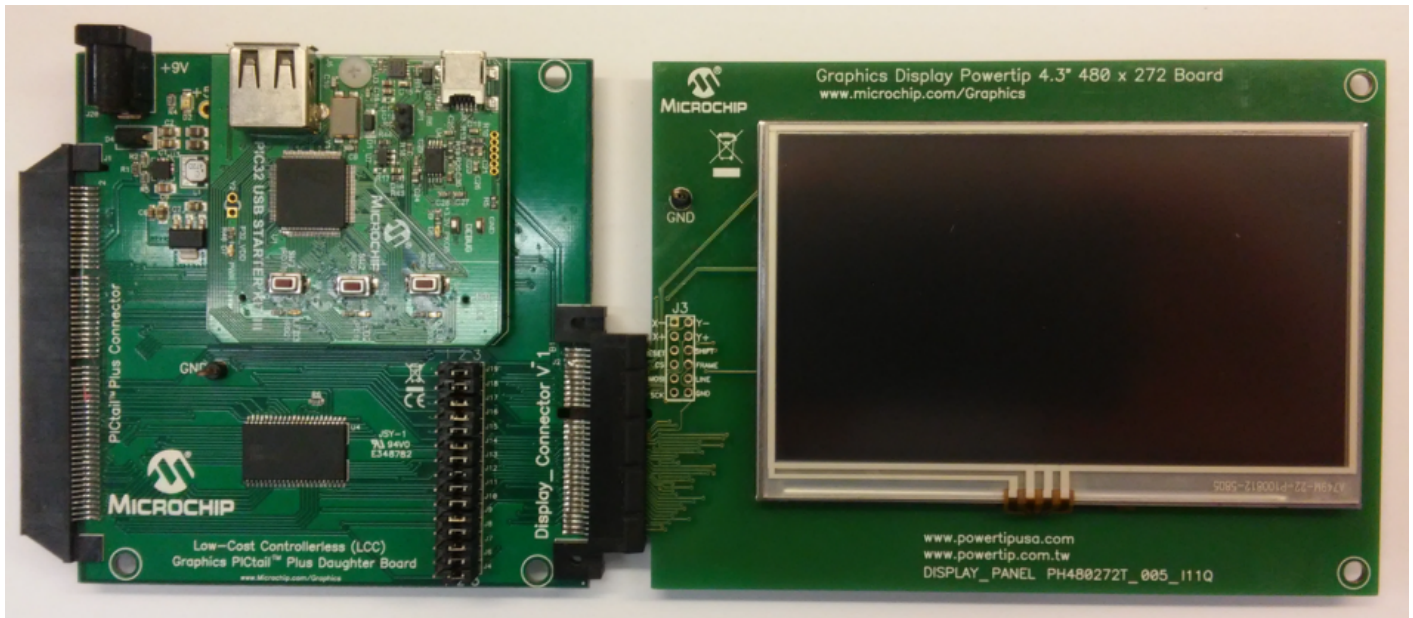
PIC32 USB Starter Kit III plus the Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board with Graphics Display Powertip 4.3" 480x272 Board BSP.

#### **Description**

This BSP is intended for the Low-Cost Controllerless (LCC) Graphics PICtail Plus Daughter Board with the Graphics Display Powertip 4.3" 480x272 Board connected to the PIC32 USB Starter Kit III.

The following figures illustrates the hardware configuration.





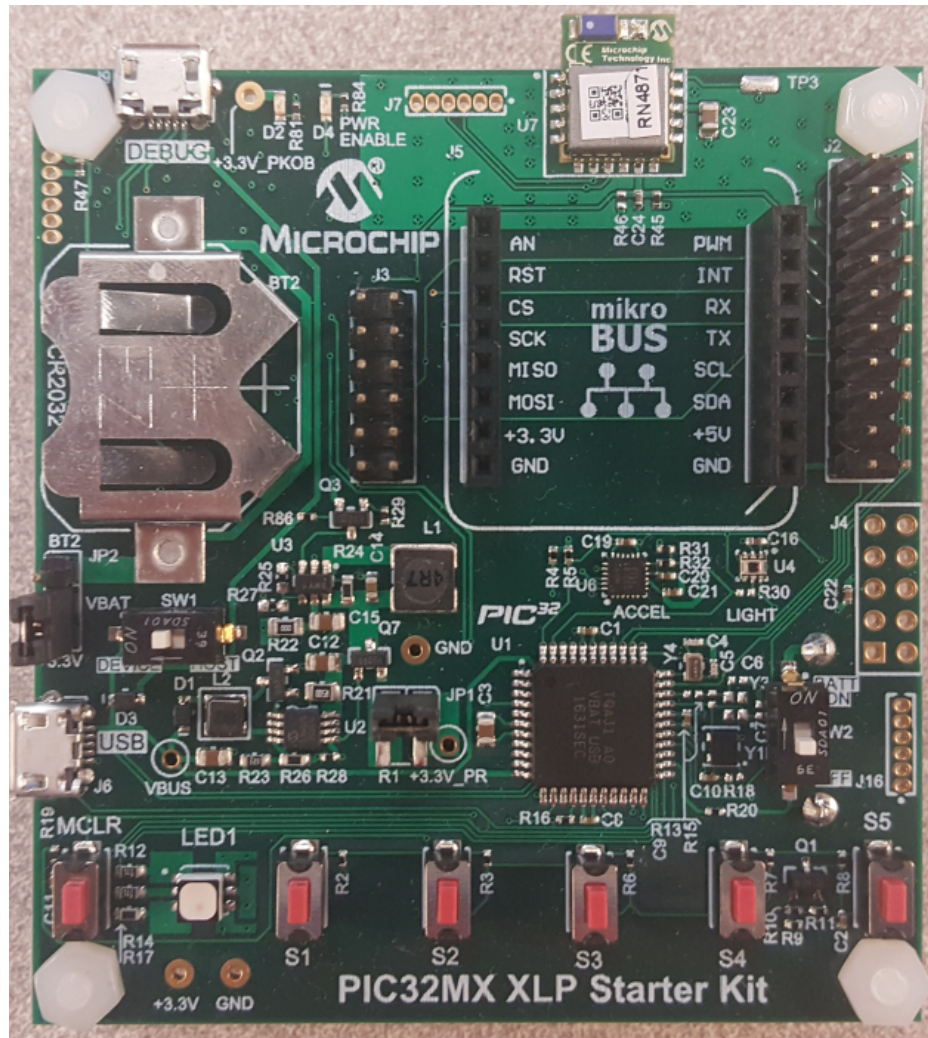
## ***pic32mx\_xlp\_sk***

PIC32MX XLP Starter Kit BSP.

### **Description**

This BSP is intended for the PIC32MX XLP Starter Kit.  
The following figures illustrates the hardware configuration.

**Top View**



Bottom View



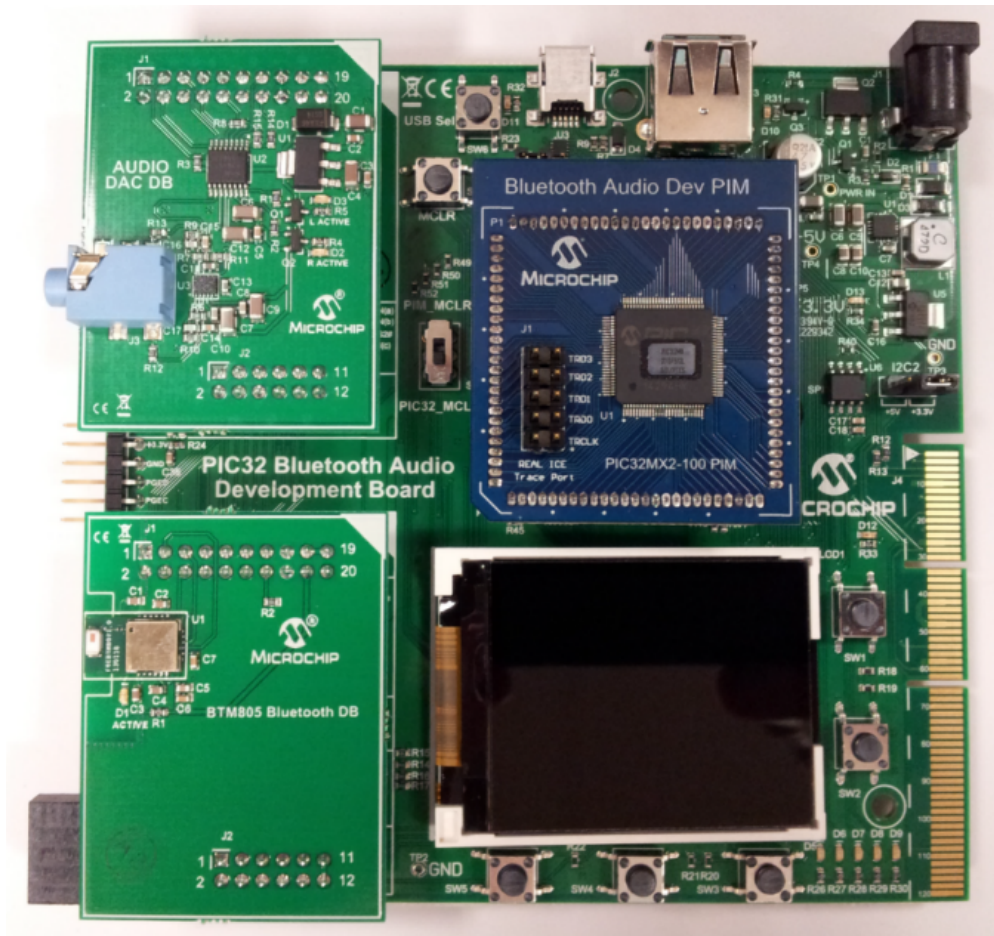
### ***pic32mx270f512l\_pim+bt\_audio\_dk***

PIC32MX270F512L Plug-in Module (PIM) plus PIC32 Bluetooth Audio Development Kit.

#### **Description**

This BSP is intended for the PIC32MX270F512L Plug-in Module (PIM) connected to the PIC32 Bluetooth Audio Development Kit. The following figure illustrates the hardware configuration.





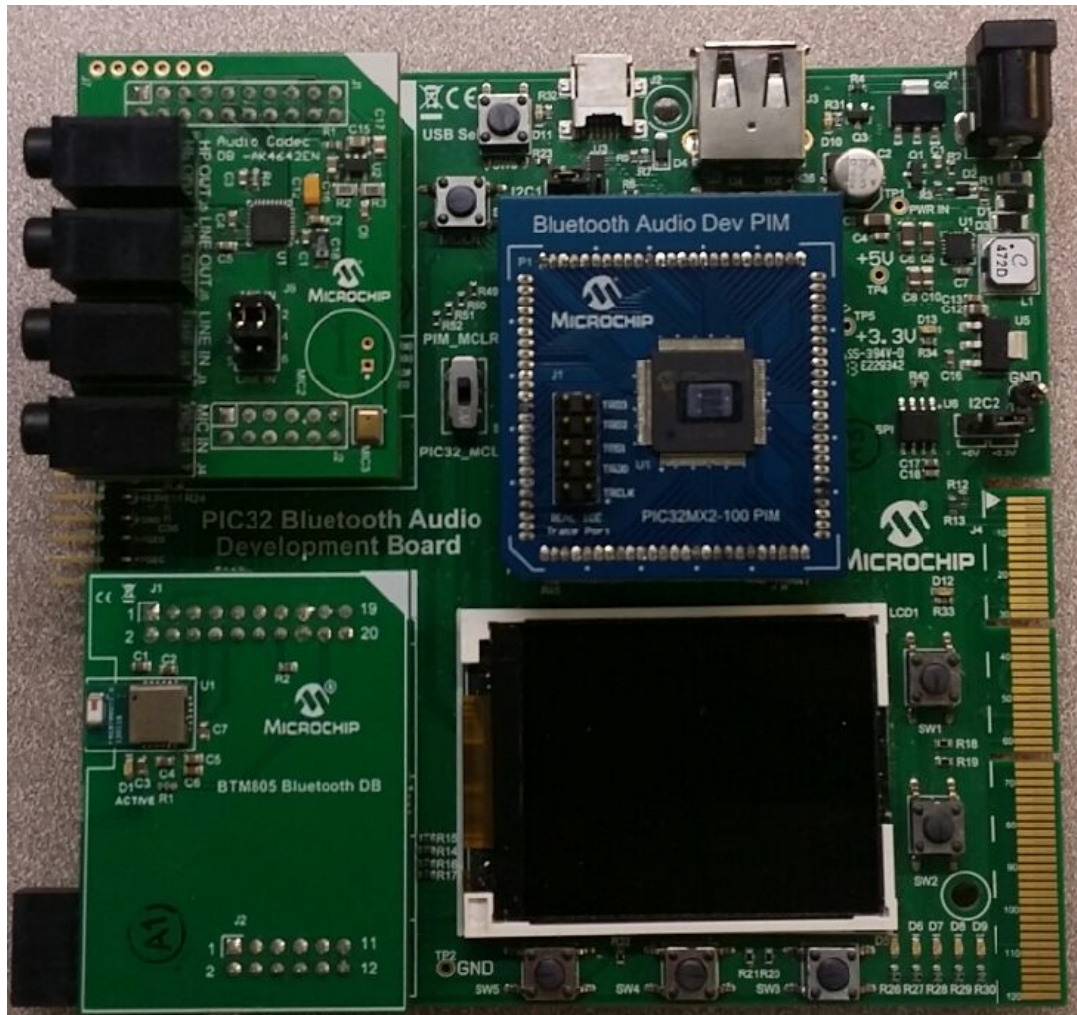
### **pic32mx\_270f512l\_pim+ bt\_audio\_dk+ak4642**

PIC32MX270F512L Plug-in Module (PIM) plus PIC32 Bluetooth Audio Development Kit with the AK4642 Audio Codec.

### **Description**

This BSP is intended for the PIC32MX270F512L Plug-in Module (PIM) connected to the PIC32 Bluetooth Audio Development Kit with the Audio Codec Daughter Board AK4642EN.

The following figure illustrates the hardware configuration.



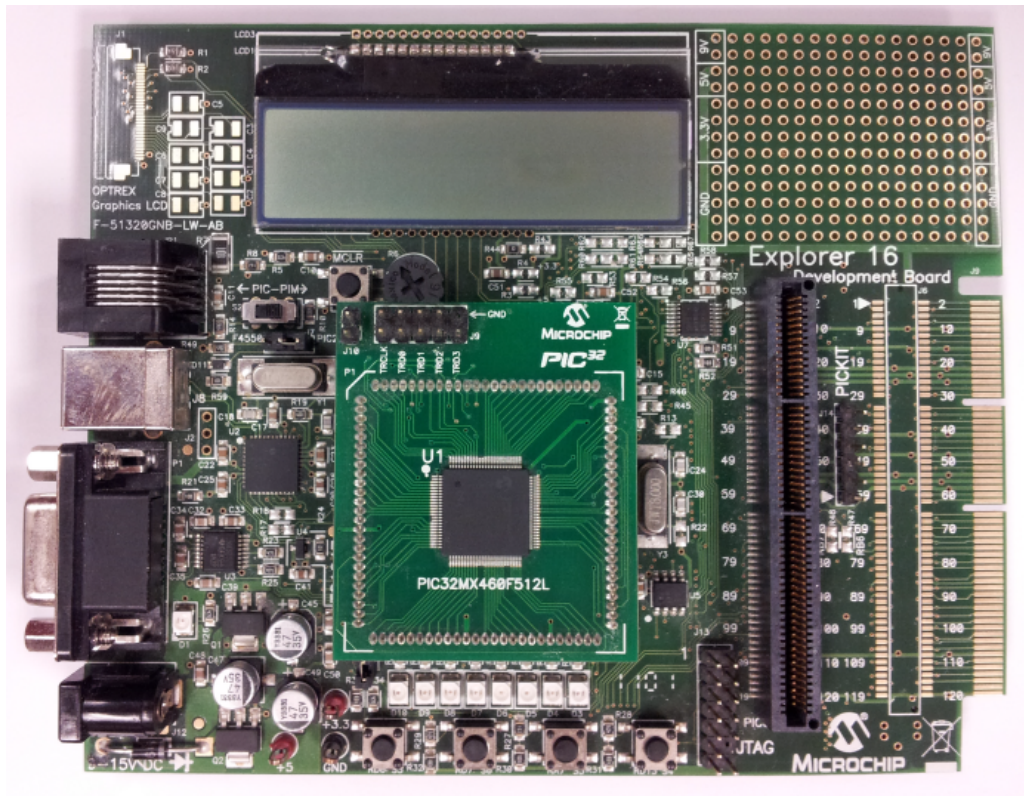
### ***pic32mx460\_pim+e16***

PIC32MX460F512L Plug-in Module (PIM) plus Explorer 16 Development Board BSP.

#### **Description**

This BSP is intended for the PIC32MX460F512L Plug-in Module (PIM) connected to the Explorer 16 Development Board. The following figure illustrates the hardware configuration.





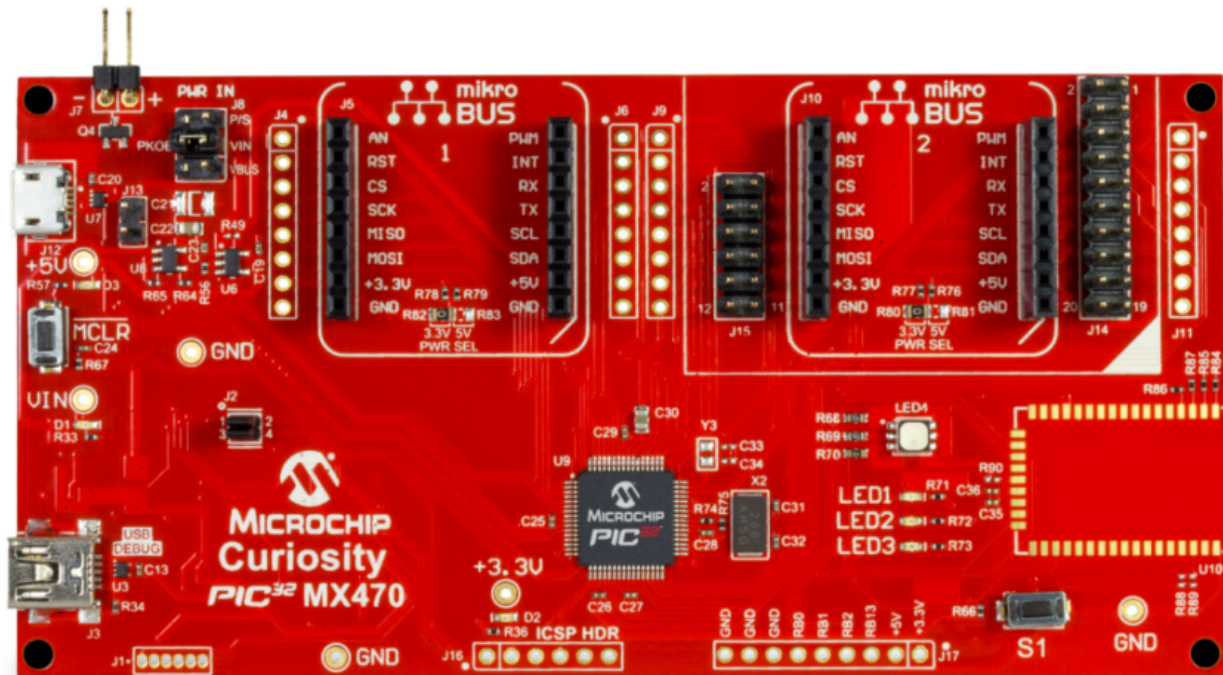
### **pic32mx470\_curiosity**

PIC32MX470 Curiosity Development Board BSP.

#### **Description**

This BSP is intended for the PIC32MX470 Curiosity Development Board.

The following figure illustrates the hardware configuration.

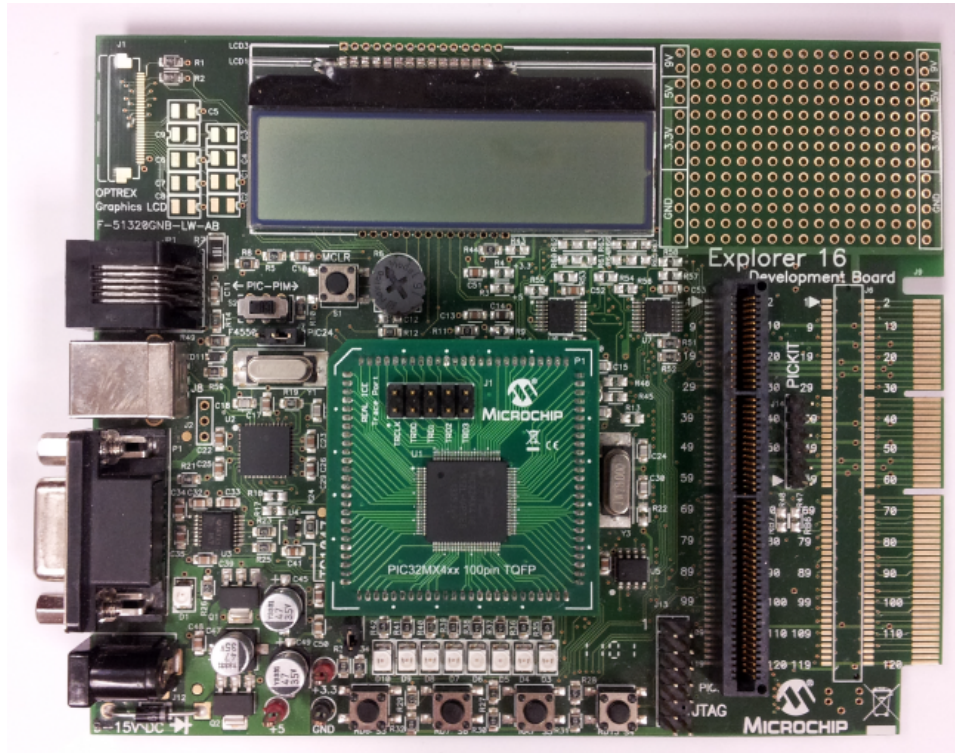


### ***pic32mx470\_pim+e16***

PIC32MX450/470F512L Plug-in Module (PIM) plus Explorer 16 Development Board BSP.

#### **Description**

This BSP is intended for the PIC32MX450/470F512L Plug-in Module (PIM) connected to the Explorer 16 Development Board. The following figure illustrates the hardware configuration.



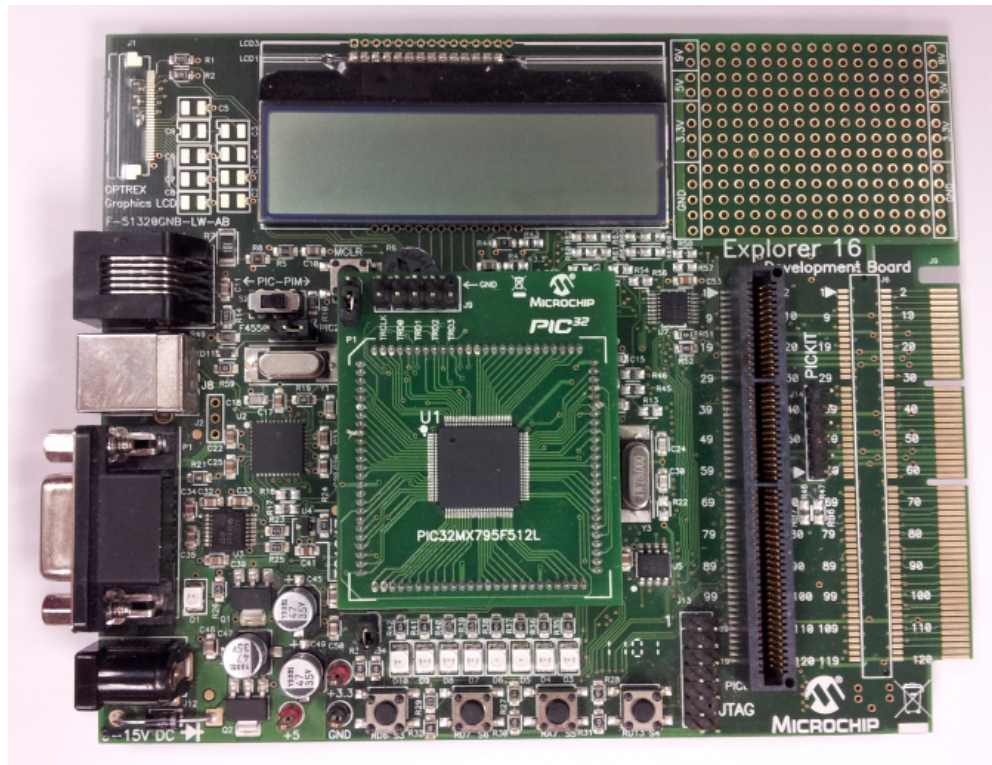
### ***pic32mx795\_pim+e16***

PIC32MX795F512L Plug-in Module (PIM) plus Explorer 16 Development Board BSP.

#### **Description**

This BSP is intended for the PIC32MX795F512L CAN-USB Plug-in Module (PIM) connected to the Explorer 16 Development Board. The following figure illustrates the hardware configuration.





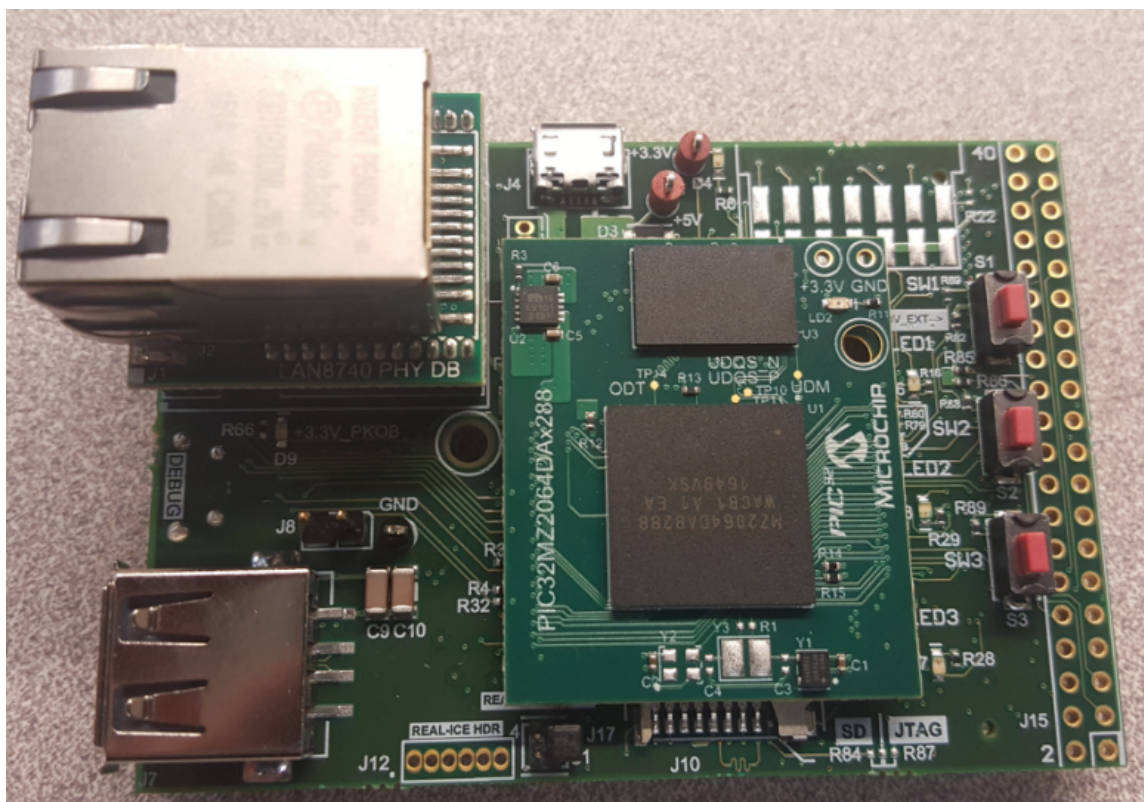
### `pic32mz_da_sk_extddr`

PIC32MZ Embedded Graphics with External DRAM (DA) Starter Kit BSP.

### Description

This BSP is intended for the PIC32MZ Embedded Graphics with External DRAM (DA) Starter Kit.

The following figure illustrates the hardware configuration.





***pic32mz\_da\_sk\_extddr+meb2***

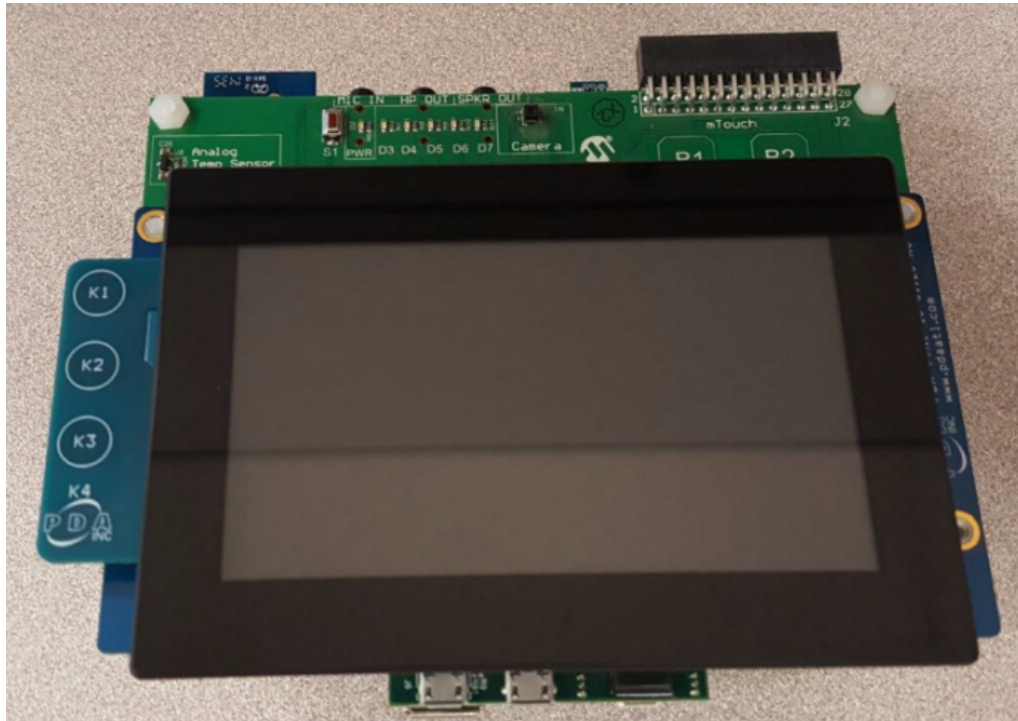
PIC32MZ Embedded Graphics with External DRAM (DA) Starter Kit plus MEB II BSP.

**Description**

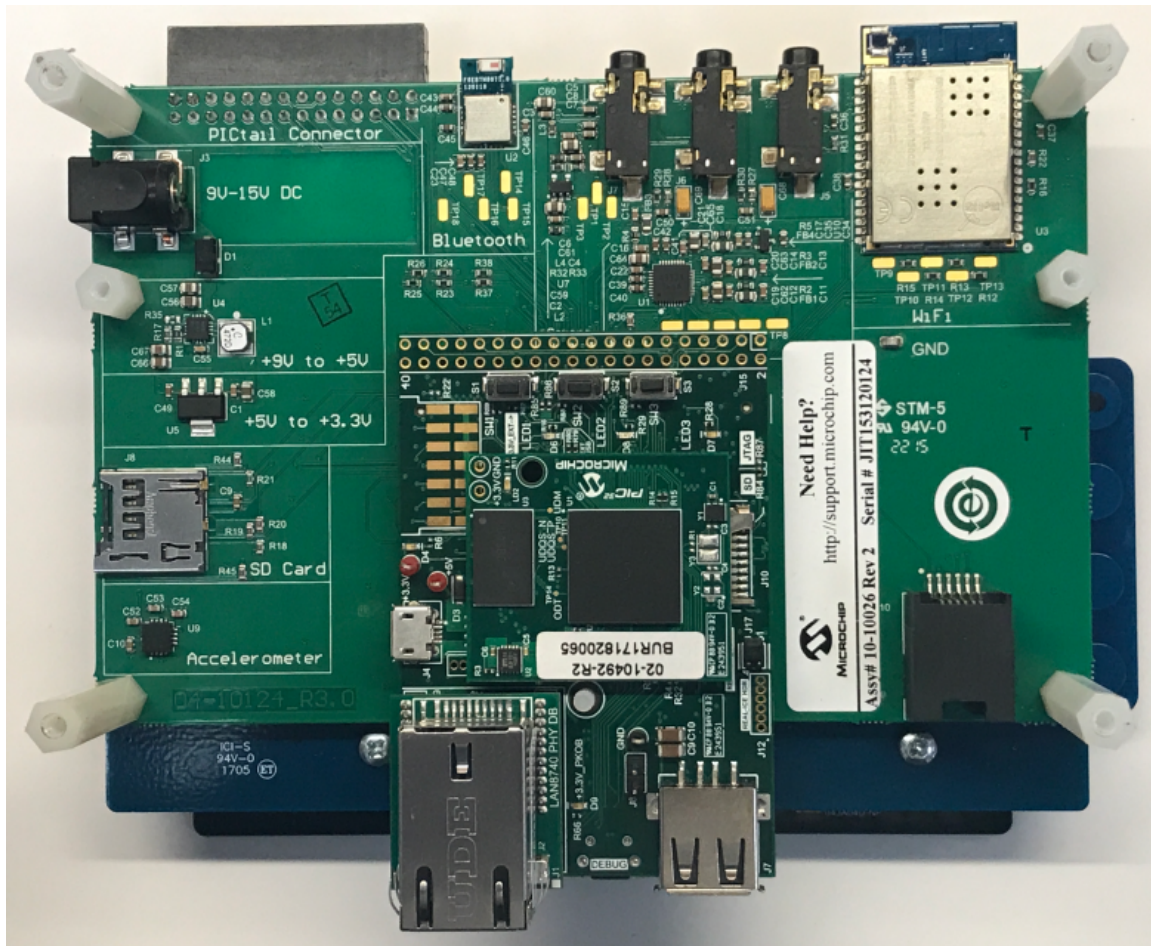
This BSP is intended for the Multimedia Expansion Board II (MEB II) connected to the PIC32MZ Embedded Graphics with External DRAM (DA) Starter Kit.

The following figures illustrate the hardware configuration.

**Top View**



**Bottom View**



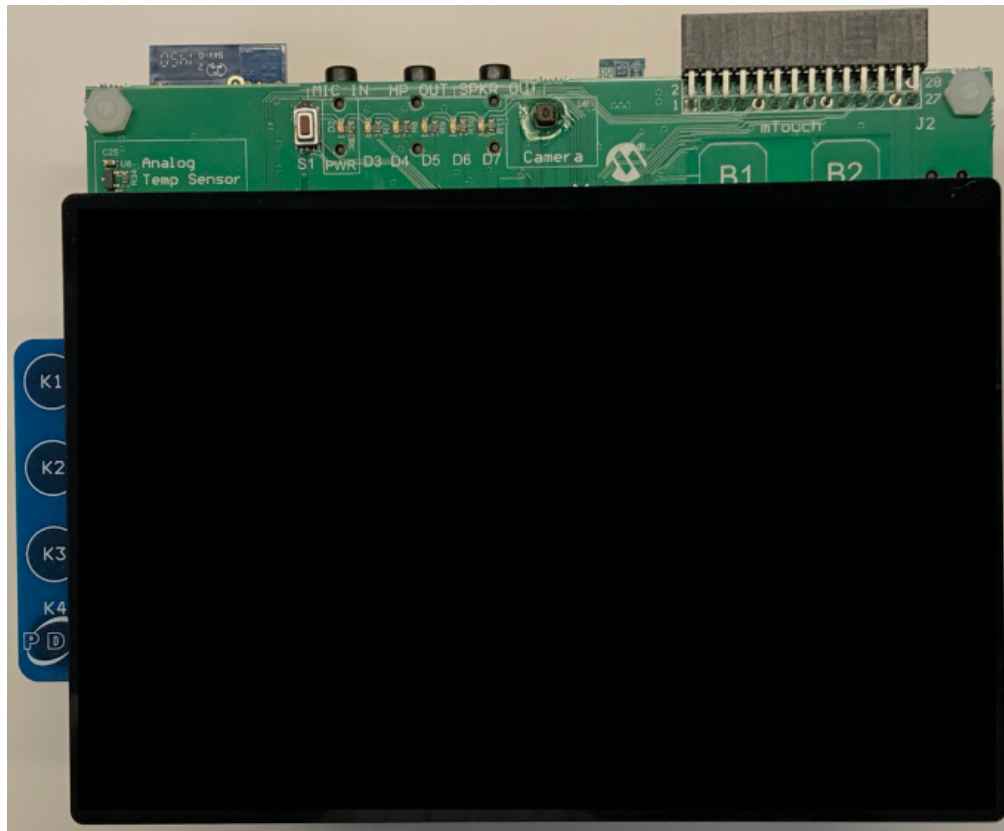
### ***pic32mz\_da\_sk\_extddr+meb2+vwga***

PIC32MZ Embedded Graphics with External DRAM (DA) Starter Kit plus MEB II and 5" WVGA PDA Display Board BSP.

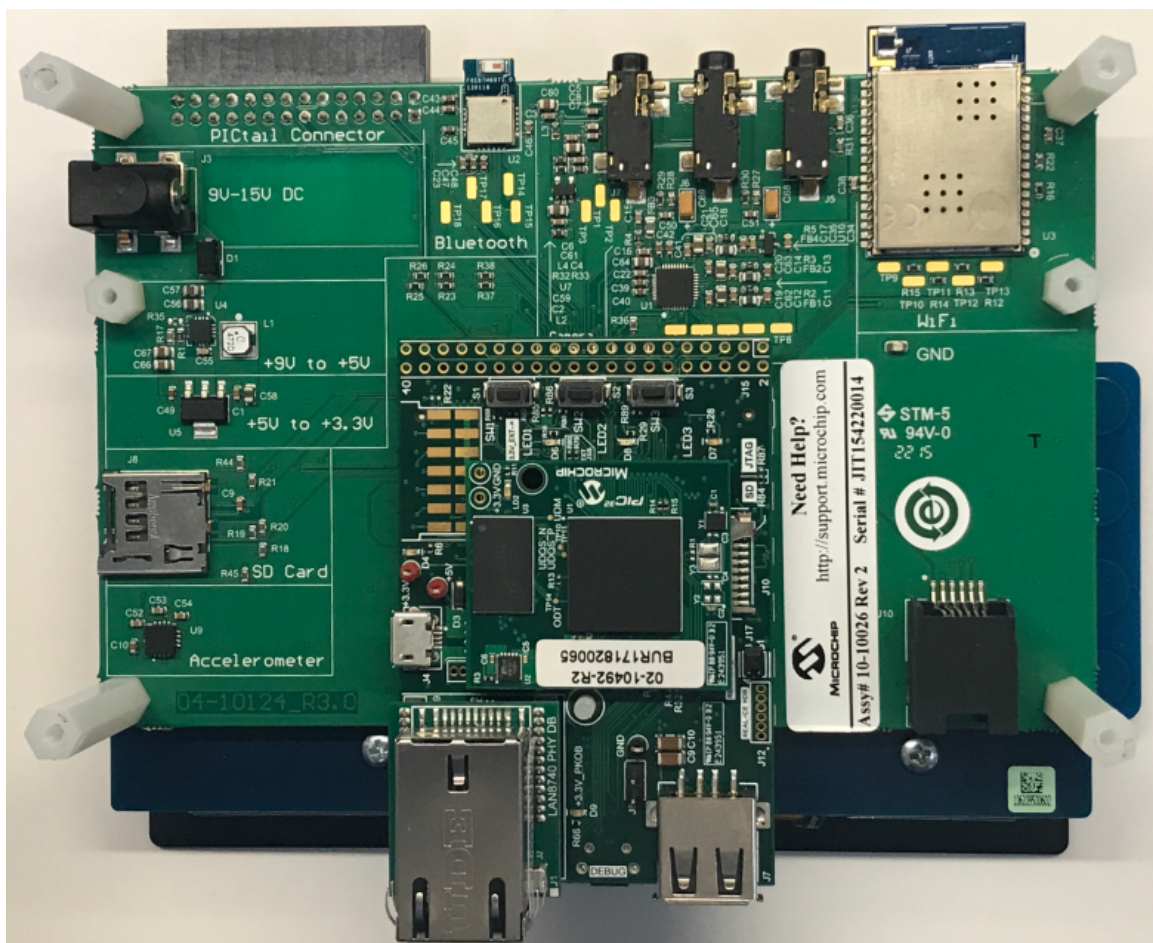
### **Description**

This BSP is intended for the Multimedia Expansion Board II (MEB II) with the High-Performance WVGA Display Module with maXTouch connected to the PIC32MZ Embedded Graphics with External DRAM (DA) Starter Kit.

**Top View**



Bottom View





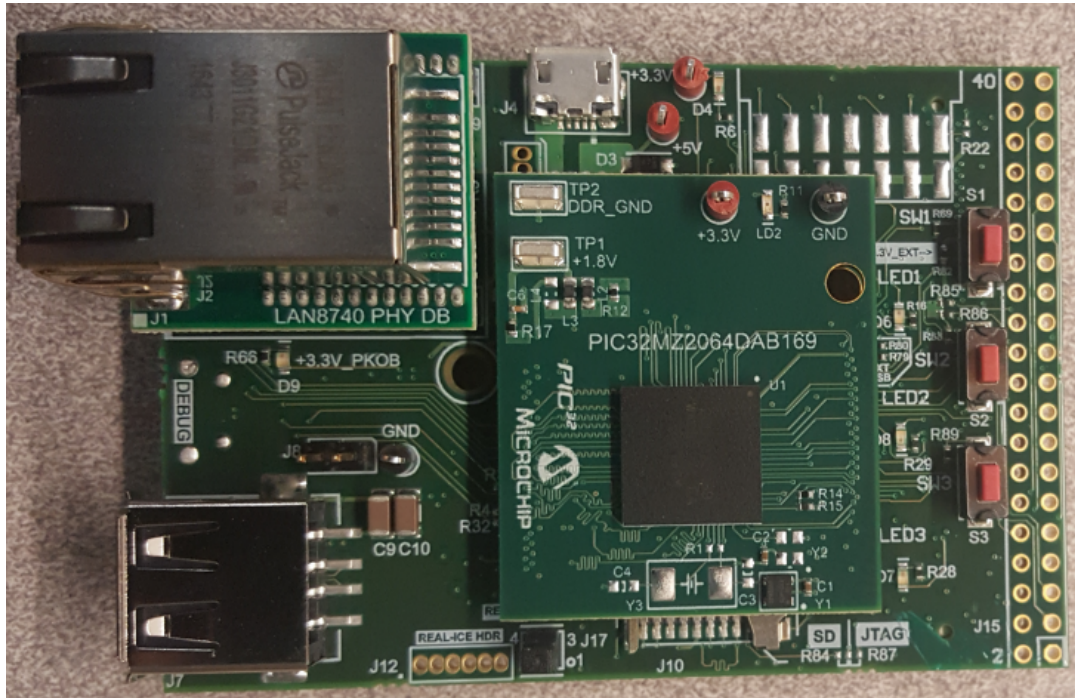
### ***pic32mz\_da\_sk\_intddr***

PIC32MZ Embedded Graphics with Stacked DRAM (DA) Starter Kit BSP.

#### **Description**

This BSP is intended for the PIC32MZ Embedded Graphics with External DRAM (DA) Starter Kit.

The following figure illustrates the hardware configuration.



### ***pic32mz\_da\_sk\_intddr+meb2***

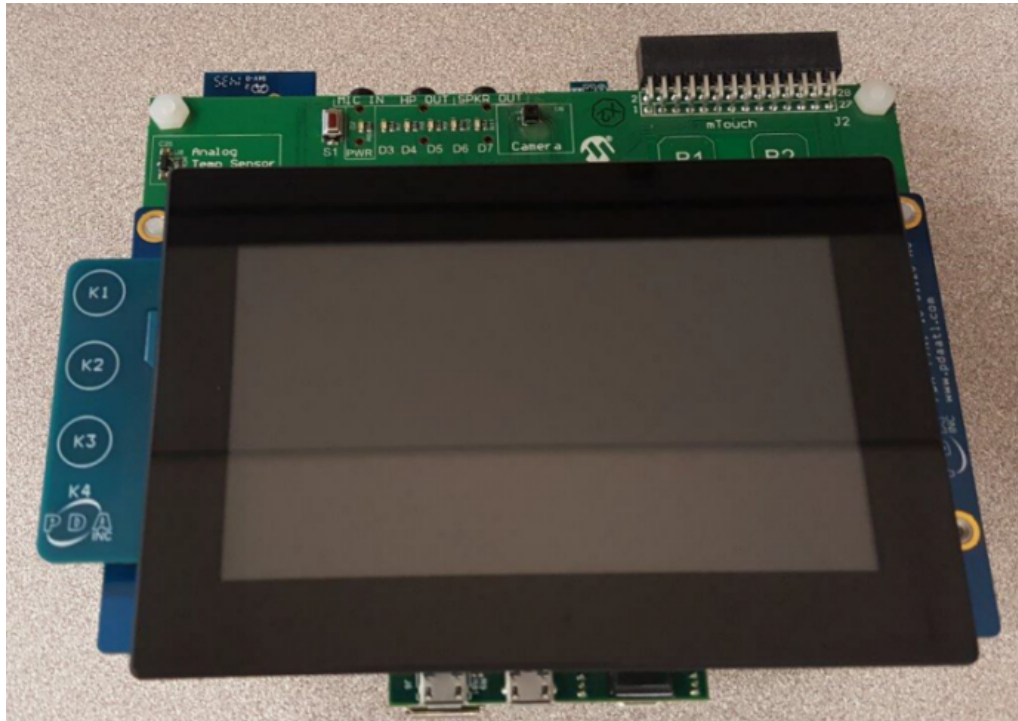
PIC32MZ Embedded Graphics with Internal DRAM (DA) Starter Kit plus MEB II BSP.

#### **Description**

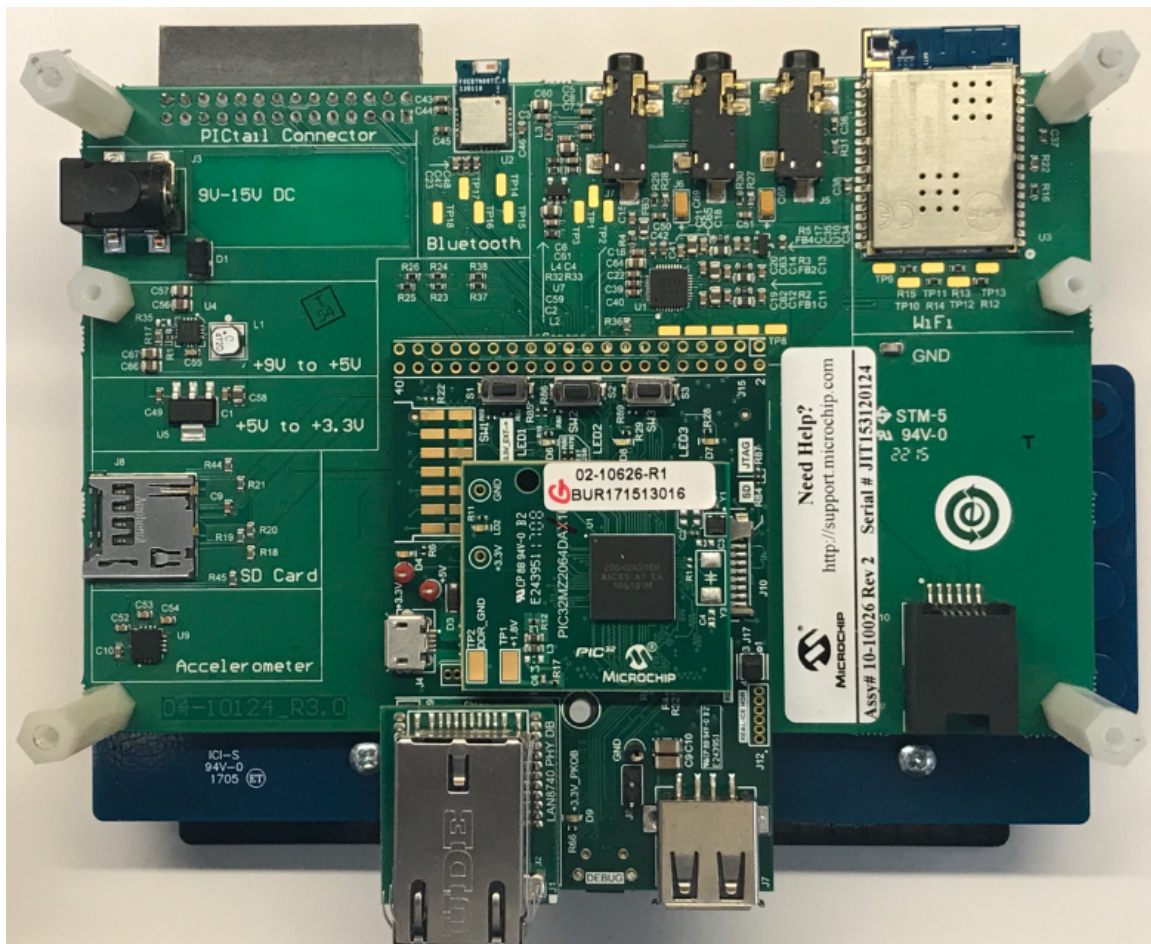
This BSP is intended for the Multimedia Expansion Board II (MEB II) connected to the PIC32MZ Embedded Graphics with Internal DRAM (DA) Starter Kit.

The following figures illustrate the hardware configuration.

**Top View**



Bottom View



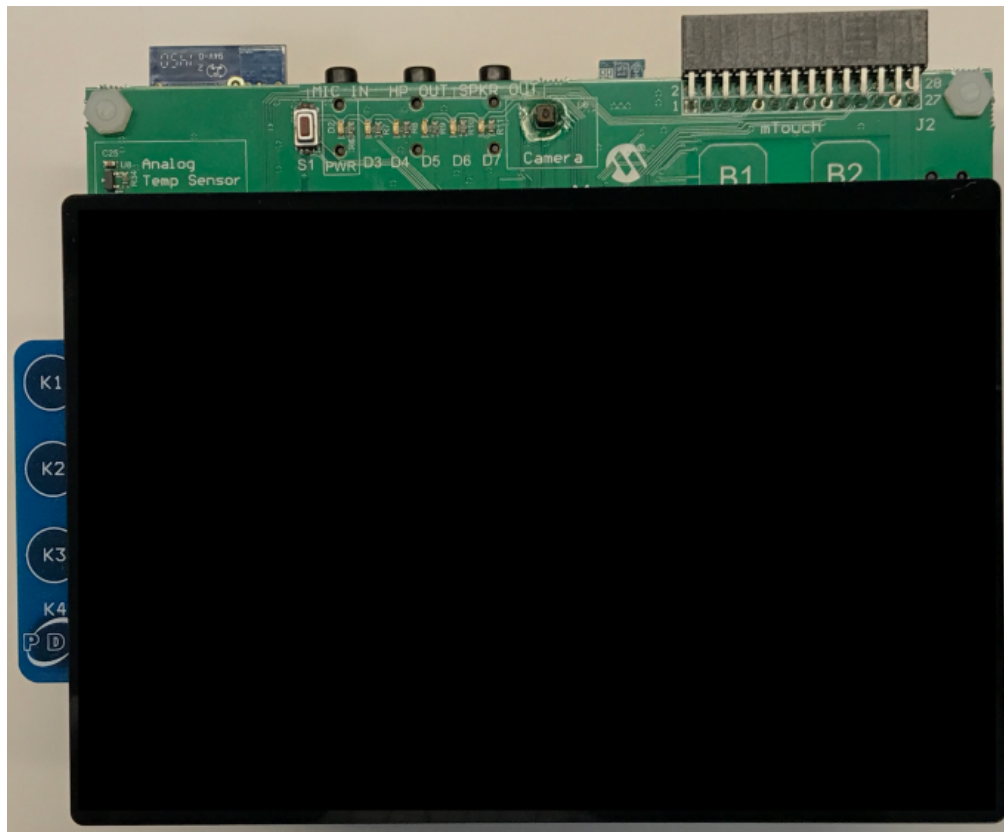
### ***pic32mz\_da\_sk\_intddr+meb2+wwga***

PIC32MZ Embedded Graphics with Internal DRAM (DA) Starter Kit plus MEB II and 5" WVGA PDA Display Board BSP.

## Description

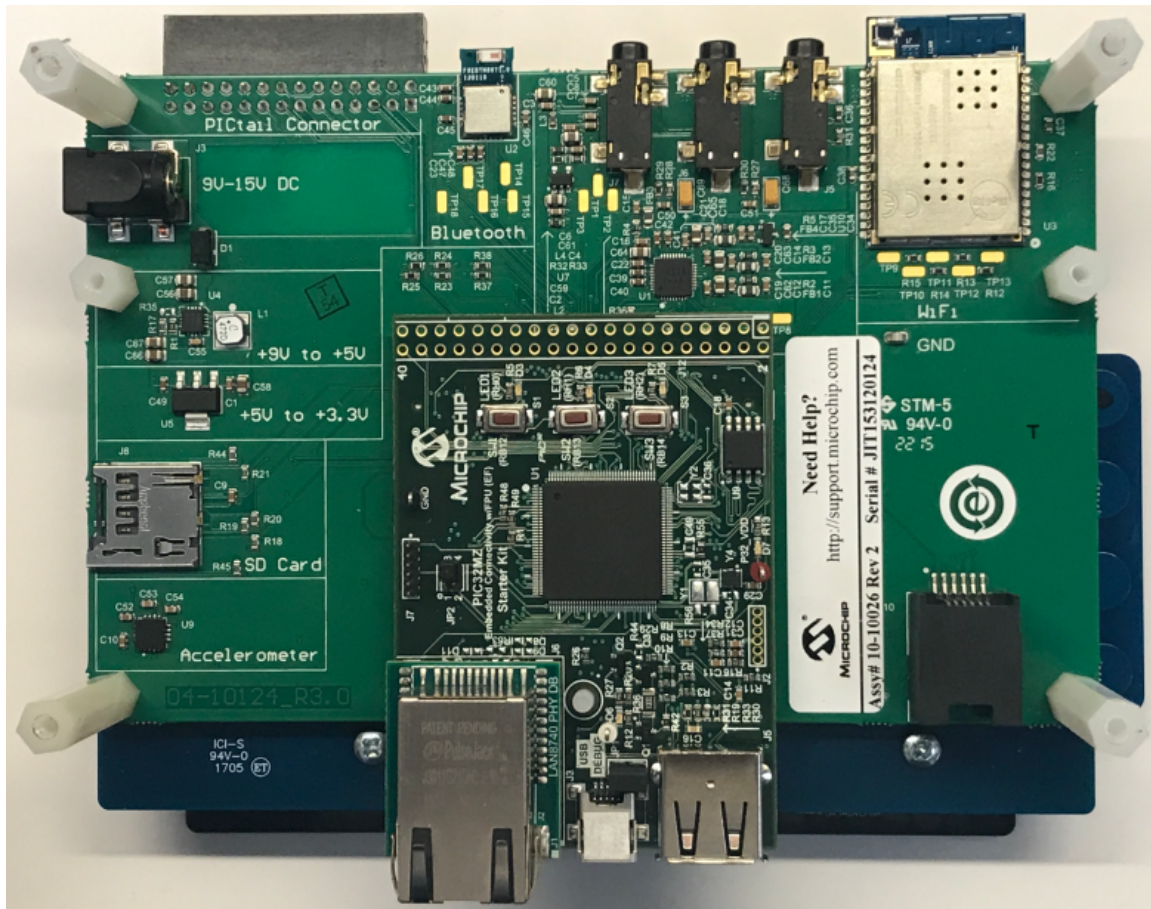
This BSP is intended for the Multimedia Expansion Board II (MEB II) with the High-Performance WVGA Display Module with maXTouch connected to the PIC32MZ Embedded Graphics with Internal DRAM (DA) Starter Kit.

**Top View**



**Bottom View**





### **pic32mz\_da\_sk\_noddr+meb2**

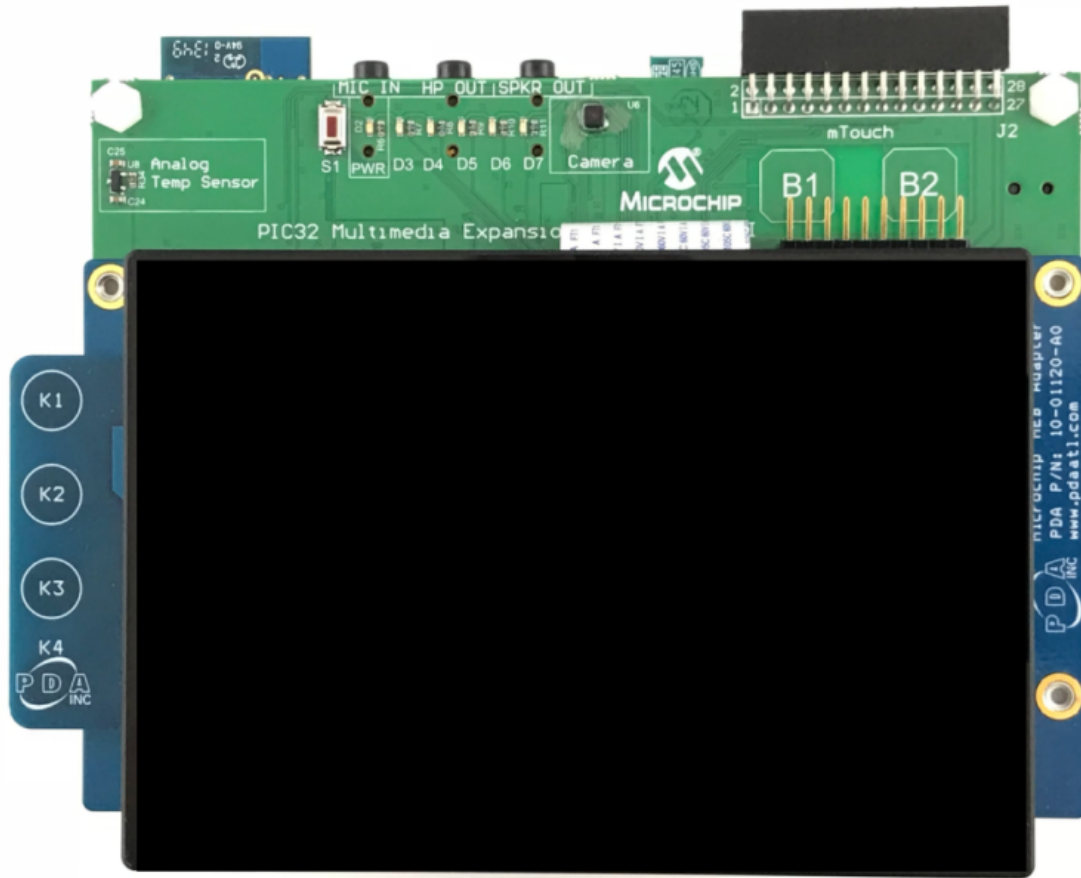
PIC32MZ Embedded Graphics with Disabled DRAM (DA) Starter Kit plus Multimedia Expansion Board II (MEB II) BSP.

### **Description**

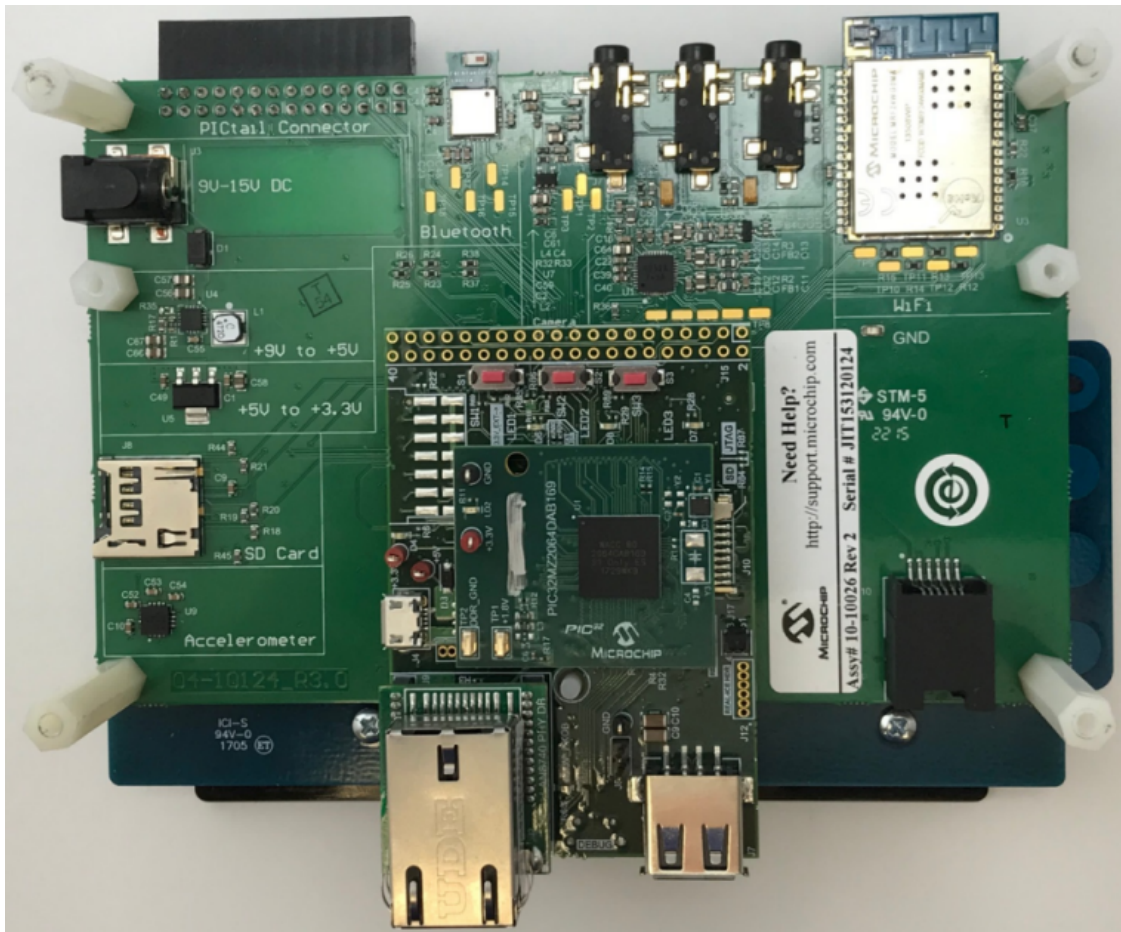
This BSP is intended for the Multimedia Expansion Board II (MEB II) connected to the PIC32MZ Embedded Graphics with Disabled DRAM (DA) Starter Kit.

The following figures illustrate the hardware configuration.

#### **Top View**



Bottom View



### ***pic32mz\_da\_sk\_noddr+meb2+wvga***

PIC32MZ Embedded Graphics with Disabled DRAM (DA) Starter Kit plus MEB II and 5" WVGA PCAP Display Board BSP.

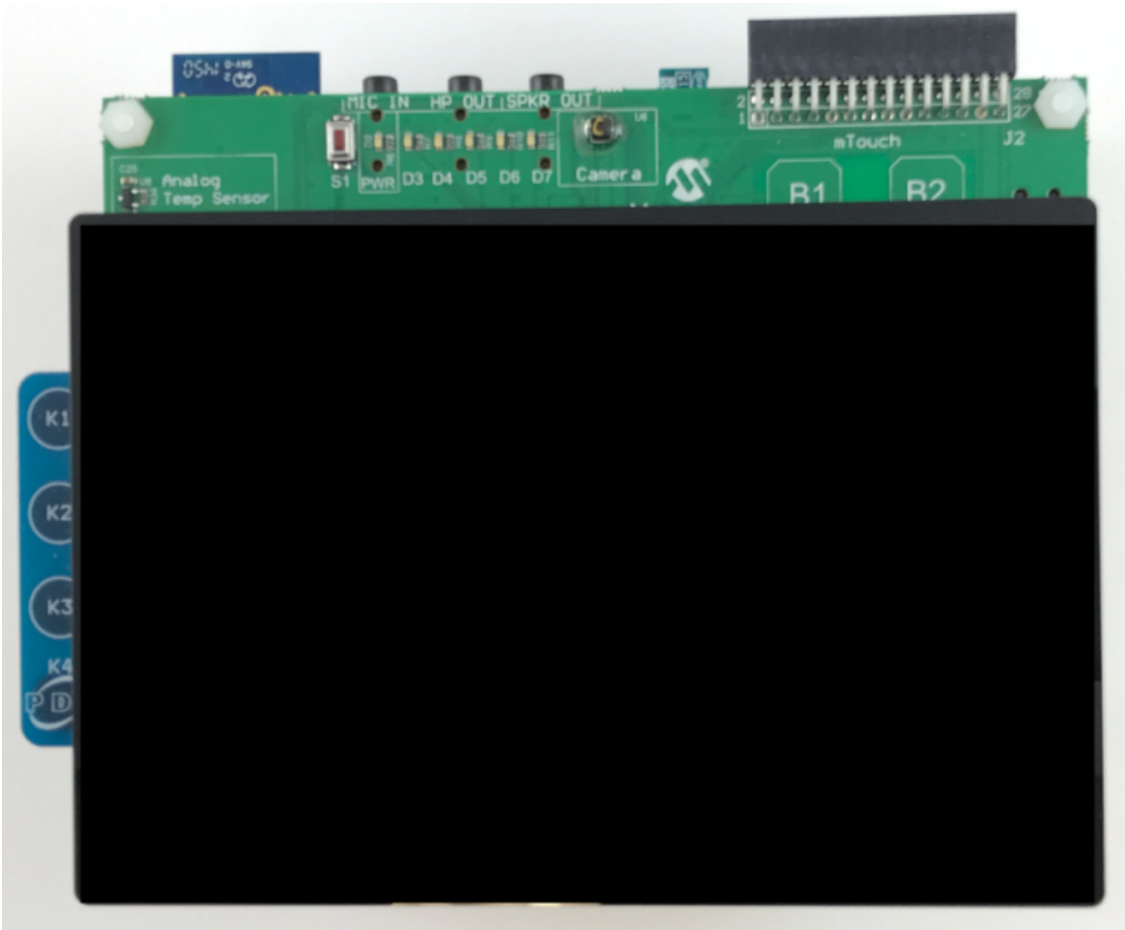
#### **Description**

This BSP is intended for the Multimedia Expansion Board II (MEB II) with the High-Performance WVGA Display Module with maXTouch connected to the PIC32MZ Embedded Graphics with Disabled DRAM (DA) Starter Kit.

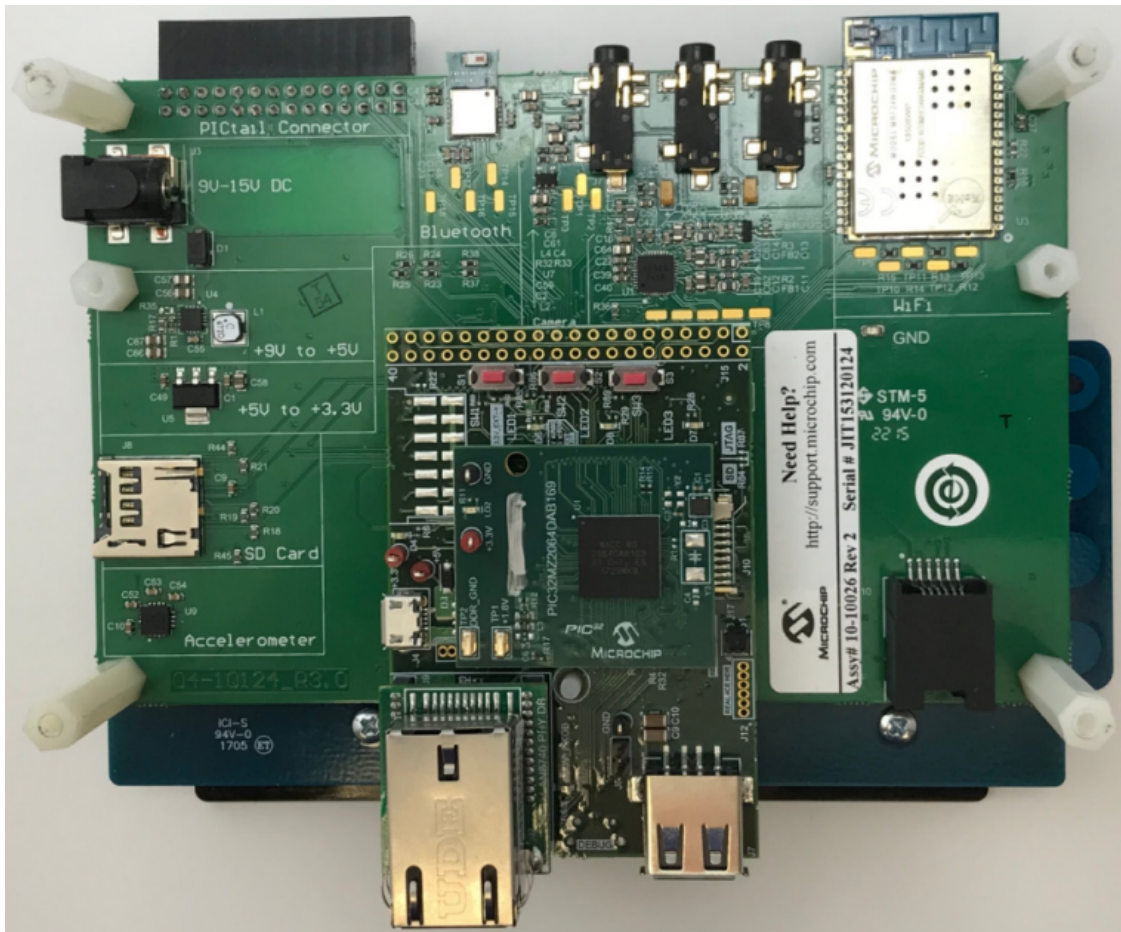
The following figures illustrate the hardware configuration.

#### **Top View**





Bottom View



### ***pic32mz\_ec\_pim+bt\_audio\_dk***

PIC32MZ2048ECH144 Audio Plug-in Module (PIM) plus PIC32 Bluetooth Audio Development Kit BSP.

#### **Description**

This BSP is intended for the PIC32MZ2048ECH144 Audio Plug-in Module (PIM) connected to the PIC32 Bluetooth Audio Development Kit. The following figure illustrates the hardware configuration.



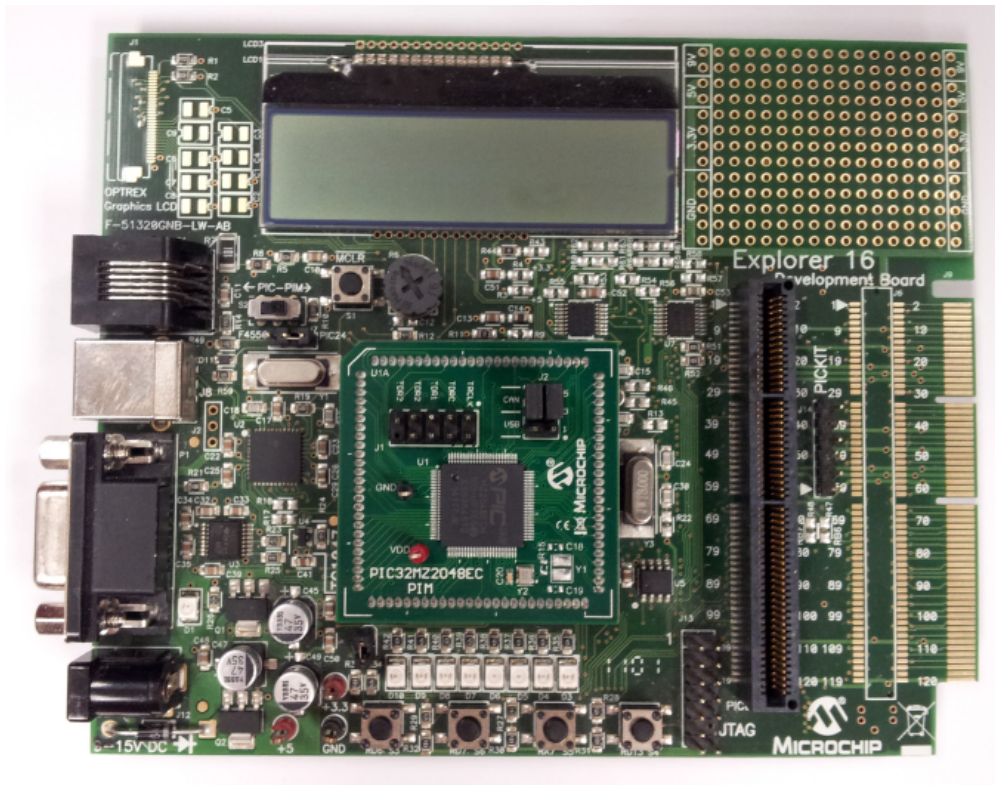
### **`pic32mz_ec_pim+e16`**

PIC32MZ2048ECH100 Plug-in Module (PIM) plus Explorer 16 Development Board BSP.

#### **Description**

This BSP is intended for the PIC32MZ2048ECH100 Plug-in Module (PIM) connected to the Explorer 16 Development Board. The following figure illustrates the hardware configuration.



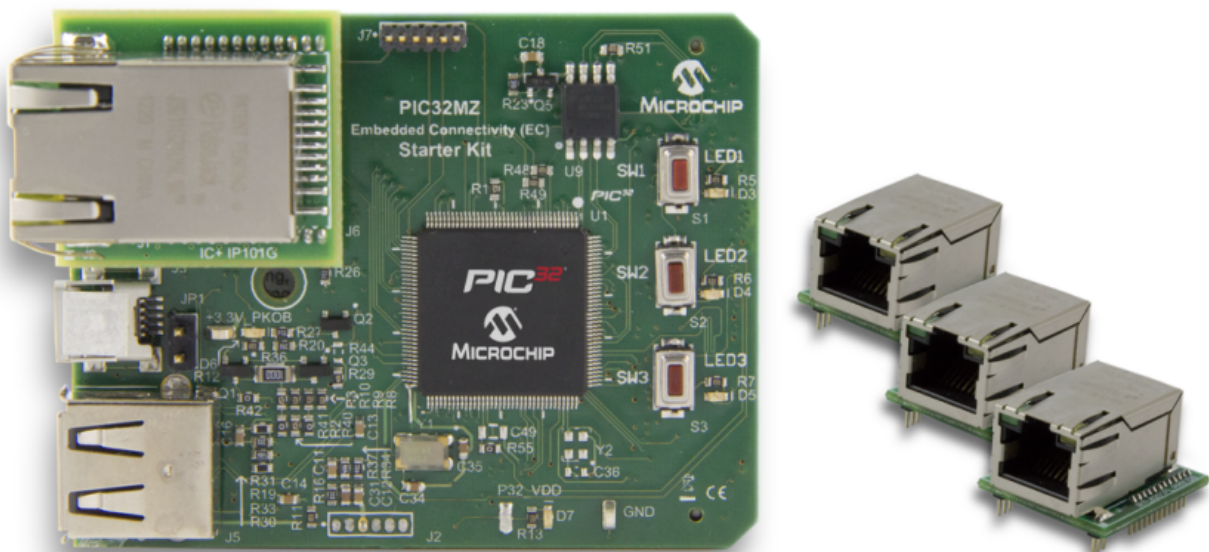


***pic32mz\_ec\_sk***

PIC32MZ EC Starter Kit BSP.

### Description

This BSP is intended for the PIC32MZ Embedded Connectivity (EC) Starter Kit. The following figure illustrates the hardware configuration.

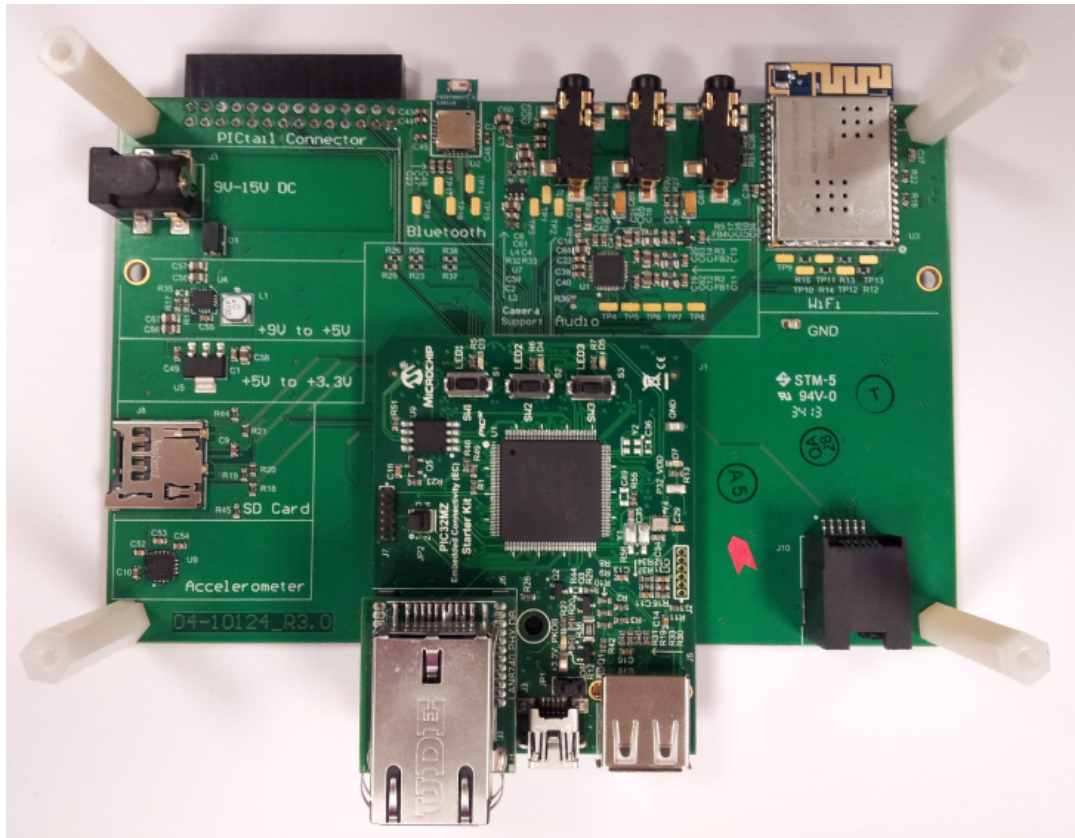


**pic32mz\_ec\_sk+meb2**

PIC32MZ EC Starter Kit plus MEB II BSP.

## Description

This BSP is intended for the Multimedia Expansion Board II (MEB II) with the PIC32MZ Embedded Connectivity (EC) Starter Kit. The following figure illustrates the hardware configuration.



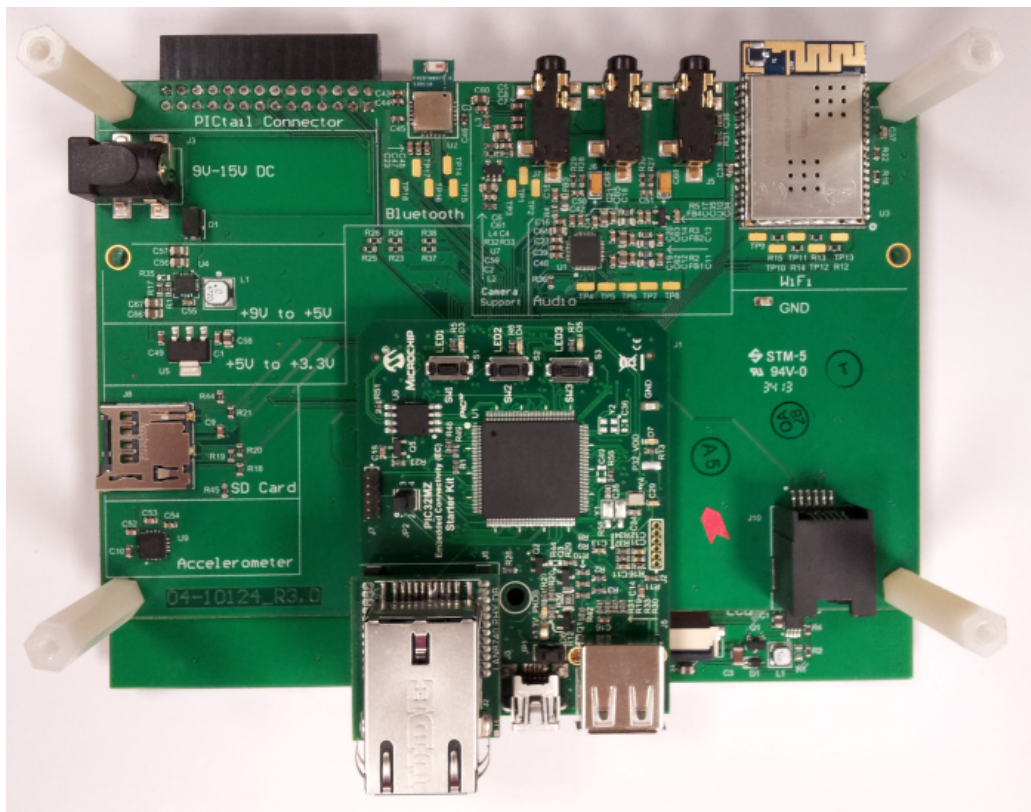
### ***pic32mz\_ec\_sk+meb2+wvga***

PIC32MZ EC Starter Kit plus MEB II and 5" WVGA PCAP Display Board BSP.

## Description

This BSP is intended for the Multimedia Expansion Board II (MEB II) with the High-Performance WVGA Display Module with maXTouch connected to the PIC32MZ Embedded Connectivity (EC) Starter Kit.

The following figure illustrates the hardware configuration.



### ***pic32mz\_ec\_sk+s1d\_pictail+vga***

PIC32MZ EC Starter Kit plus Graphics Controller PICTail Plus Epson S1D13517 Daughter Board with Graphics Display Truly 5.7" 640x480 Board.

### **Description**

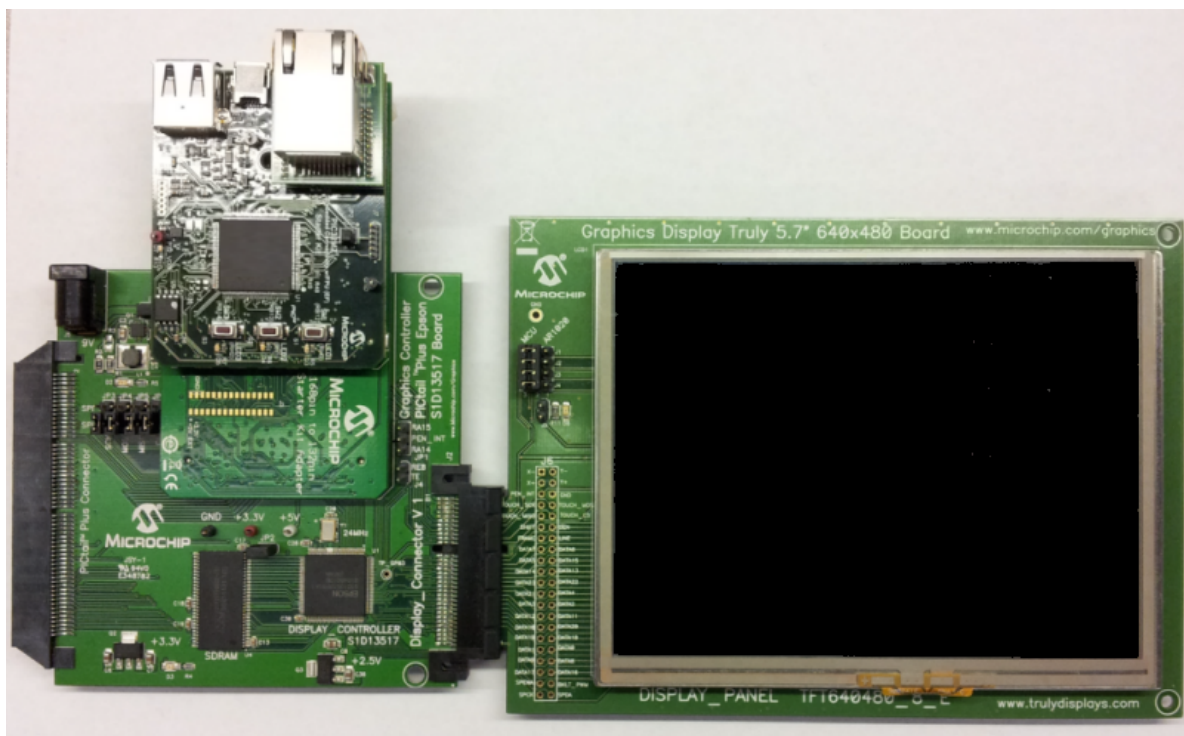
This BSP is intended for the Graphics Controller PICTail Plus Epson S1D13517 Daughter Board with the Graphics Display Truly 5.7" 640x480 Board connected to the PIC32MZ Embedded Connectivity (EC) Starter Kit.



**Note:**

The starter kit shown in the following figure is the PIC32MZ EF Starter Kit. The PIC32MZ EC and PIC32MZ EF starter kits are identical with the exception of the on-board device, so the hardware configuration is the same regardless of which starter kit is used.





### ***pic32mz\_ec\_sk+s1d\_pictail+wqvga***

PIC32MZ EC Starter Kit plus Graphics Controller PICTail Plus Epson S1D13517 Daughter Board with Graphics Display Powertip 4.3" 480x272 Board BSP.

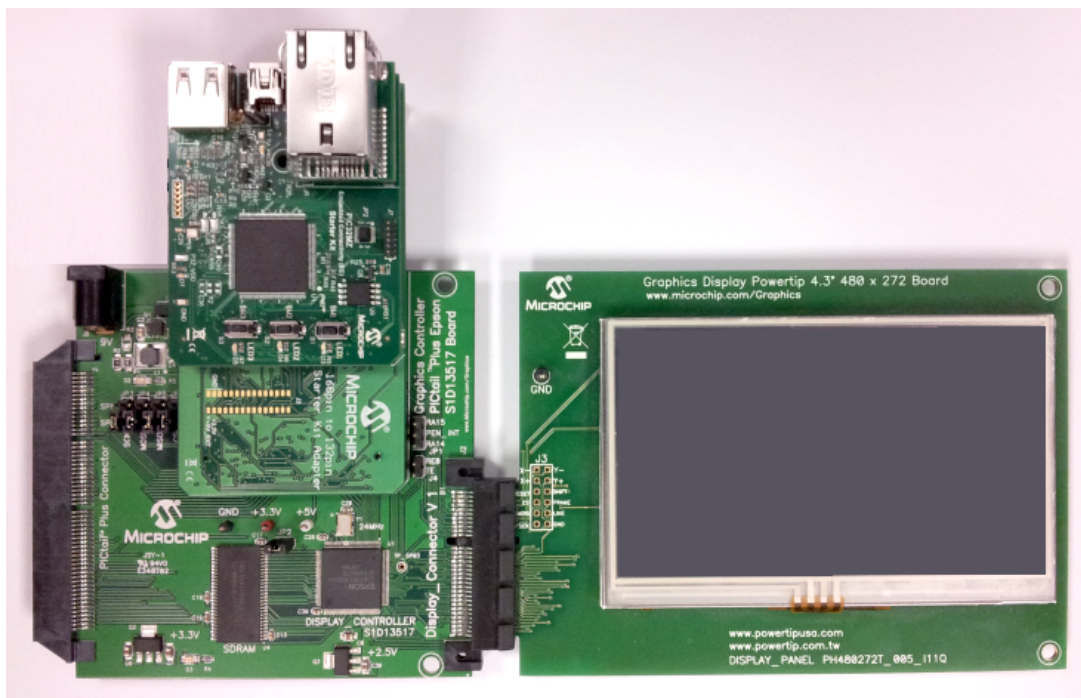
### **Description**

This BSP is intended for the Graphics Controller PICTail Plus Epson S1D13517 Daughter Board with the Graphics Display Powertip 4.3" 480x272 Board connected to the PIC32MZ Embedded Connectivity (EC) Starter Kit with the PIC32MZ Starter Kit Adapter Board.



**Note:** The PIC32MZ EC Adapter Board is required when using the Graphics Controller PICTail Plus Epson S1D13517 Daughter Board with the PIC32MZ EC Starter Kit.

The following figure illustrates the hardware configuration.

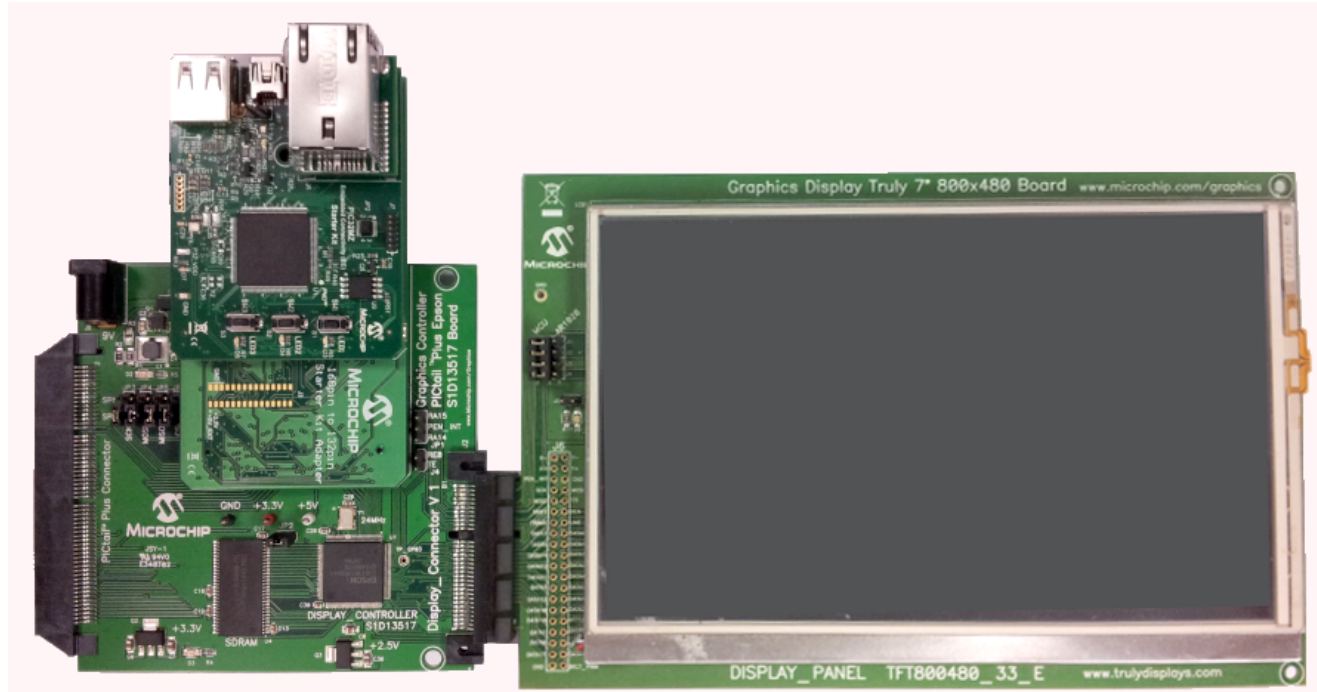


### ***pic32mz\_ec\_sk+s1d\_pictail+wvga***

PIC32MZ EC Starter Kit plus Graphics Controller PICtail Plus Epson S1D13517 Daughter Board and Graphics Display Truly 7" 800x480 Board BSP.

#### **Description**

This BSP is intended for the Graphics Controller PICtail Plus Epson S1D13517 Daughter Board with Graphics Display Truly 7" 800x480 Board connected to the PIC32MZ Embedded Connectivity (EC) Starter Kit.



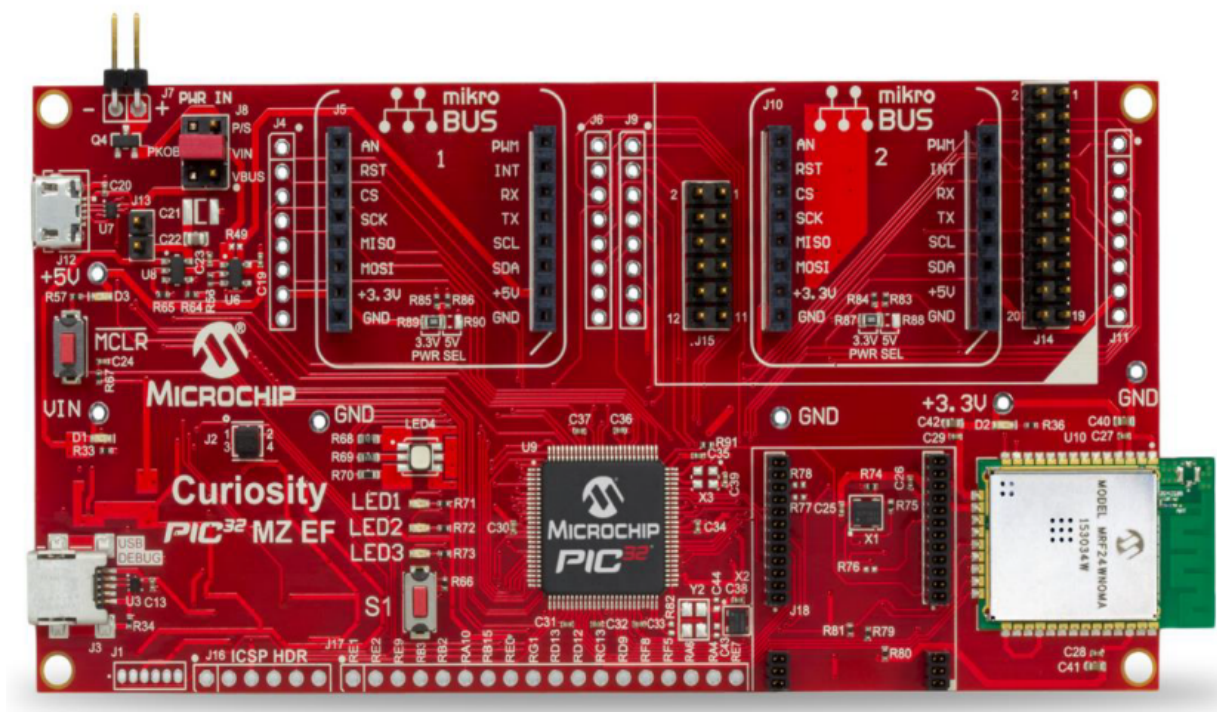
### ***pic32mz\_ef\_curiosity***

PIC32MZ EF Curiosity Development Board BSP.

#### **Description**

This BSP is intended for the PIC32MZ EF Curiosity Development Board.

The following figure illustrates the hardware configuration.



### *pic32mz\_ef\_pim+bt\_audio\_dk*

PIC32MZ2048EFH144 Audio Plug-in Module (PIM) plus PIC32 Bluetooth Audio Development Kit BSP.

### Description

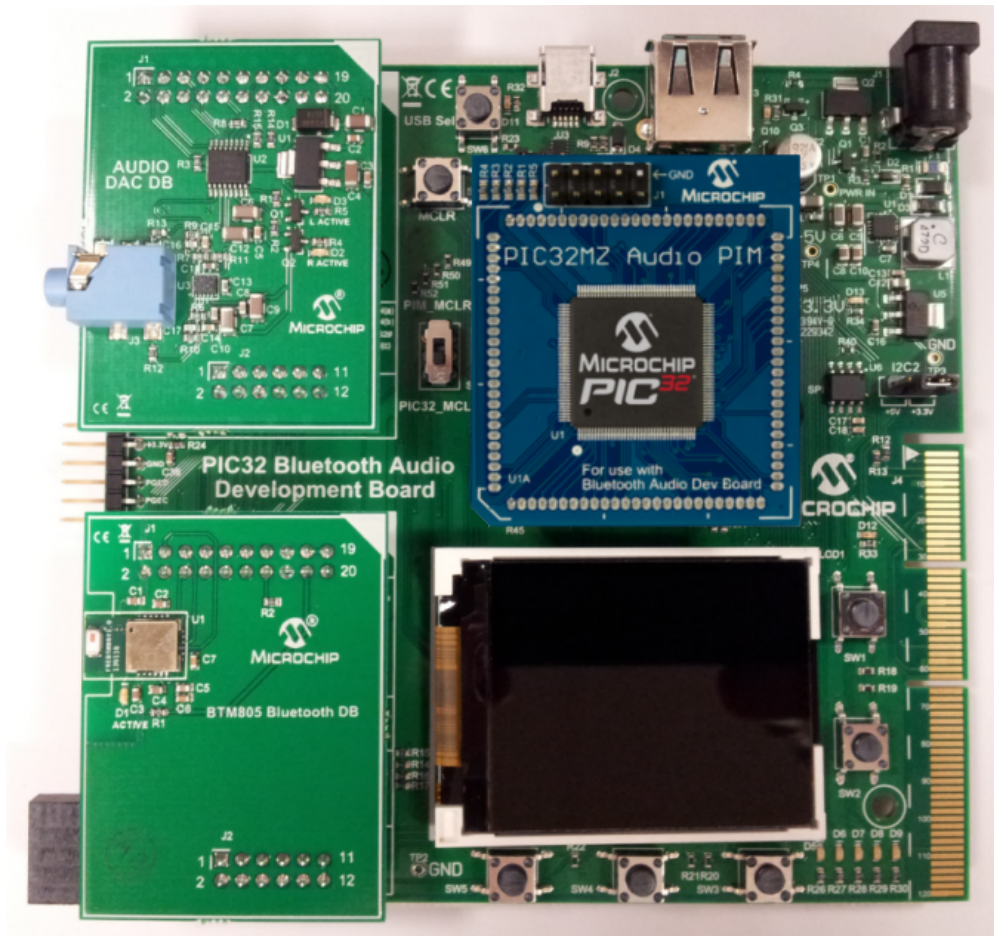
This BSP is intended for the PIC32MZ2048EFH144 Audio Plug-in Module (PIM) connected to the PIC32 Bluetooth Audio Development Kit.



#### Note:

The PIM shown in the following figure is the PIC32MZEC2048. The PIC32MZ EC and PIC32MZ EF PIMs are identical with the exception of the on-board device, so the hardware configuration is the same regardless of which PIM is used.



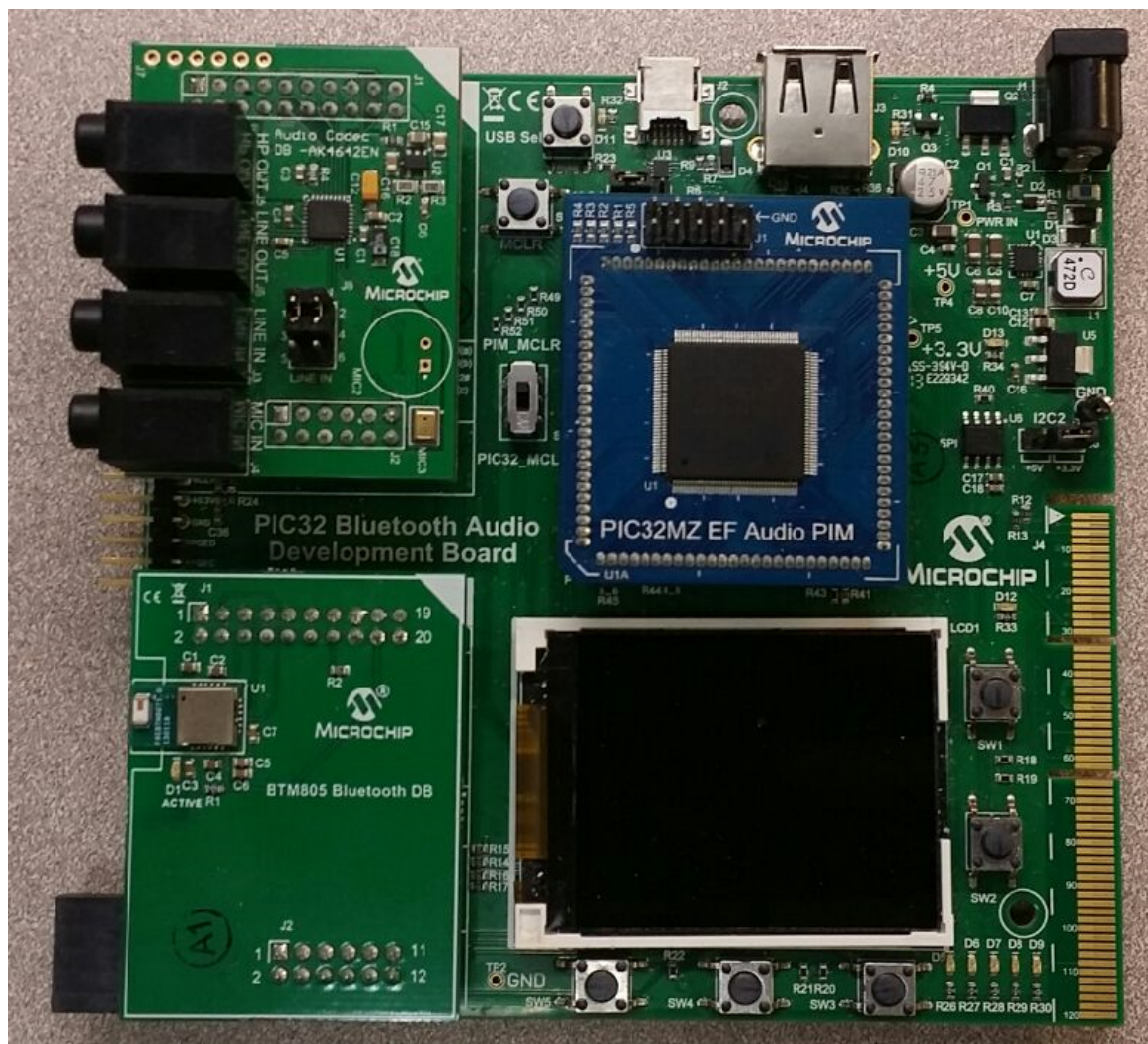


### ***pic32mz\_ef\_pim+bt\_audio\_dk+ak4642***

PIC32MZ2048EFH144 Audio Plug-in Module (PIM) plus PIC32 Bluetooth Audio Development Kit and AK4642 Audio Codec BSP.

#### **Description**

This BSP is intended for the PIC32MZ2048EFH144 Audio Plug-in Module (PIM) connected to the PIC32 Bluetooth Audio Development Kit with the AK4642 Audio Codec.



### pic32mz\_ef\_pim+e16

PIC32MZ2048EFH100 Plug-in Module (PIM) plus Explorer 16 Development Board BSP.

### Description

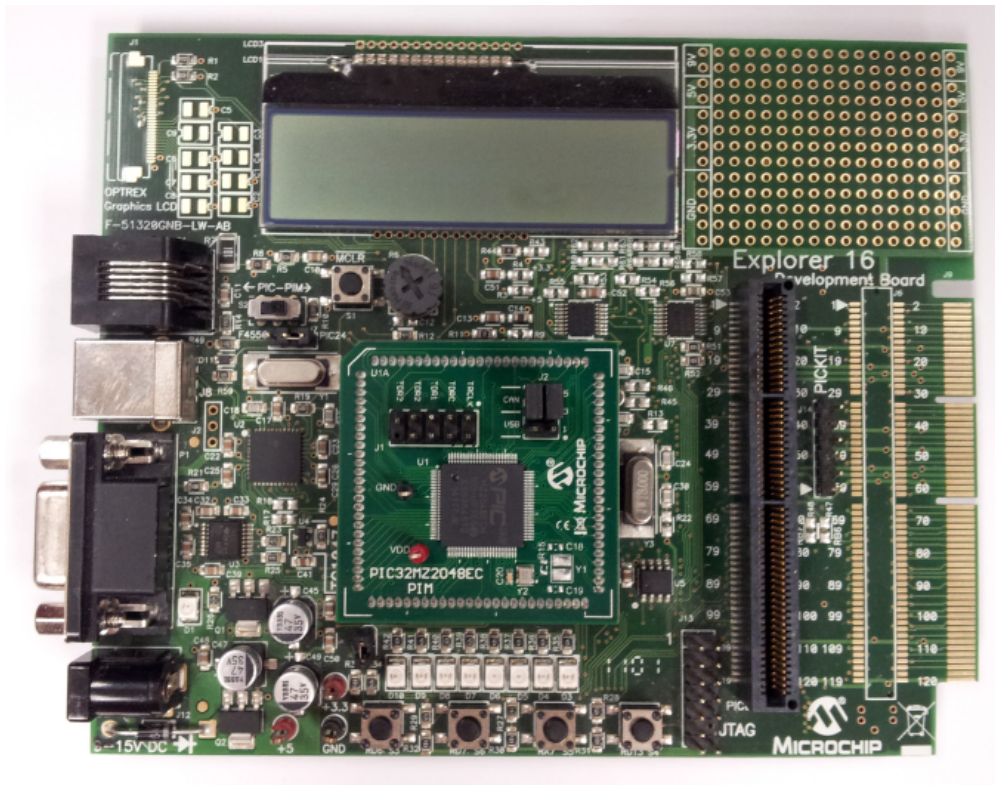
This BSP is intended for the PIC32MZ2048EFH100 Plug-in Module (PIM) connected to the Explorer 16 Development Board.



**Note:**

The PIM shown in the following figure is the PIC32MZEC2048. The PIC32MZ EC and PIC32MZ EF PIMs are identical with the exception of the on-board device, so the hardware configuration is the same regardless of which PIM is used.





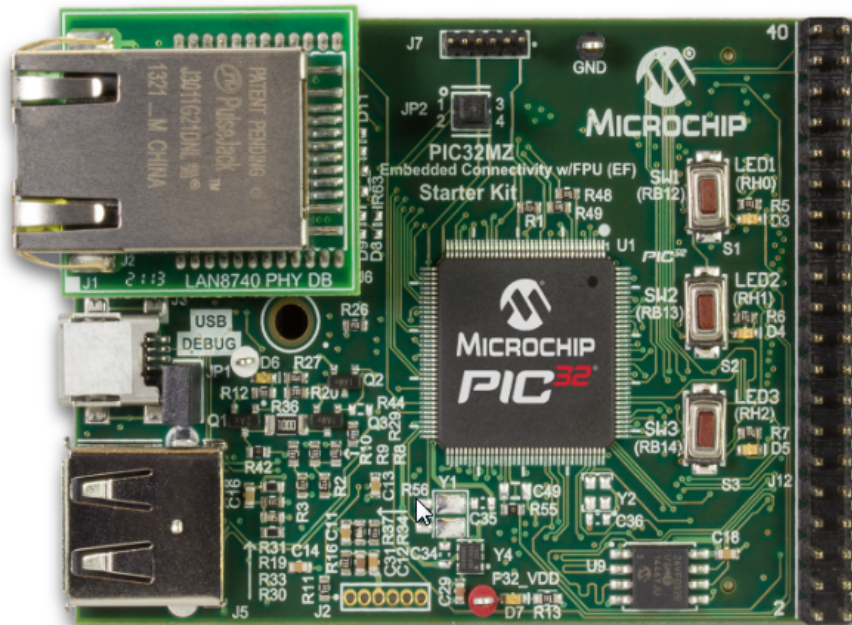
### **`pic32mz_ef_sk`**

PIC32MZ EF Starter Kit BSP.

### **Description**

This BSP is intended for the PIC32MZ Embedded Connectivity (EF) Starter Kit.

The following figure illustrates the hardware configuration.





### ***pic32mz\_ef\_sk+maxtouch\_xplained\_pro\_3\_5***

PIC32MZ Embedded Connectivity with Floating Point Unit (EF) Starter Kit with the maXTouch Xplained Pro 3.5" display BSP.

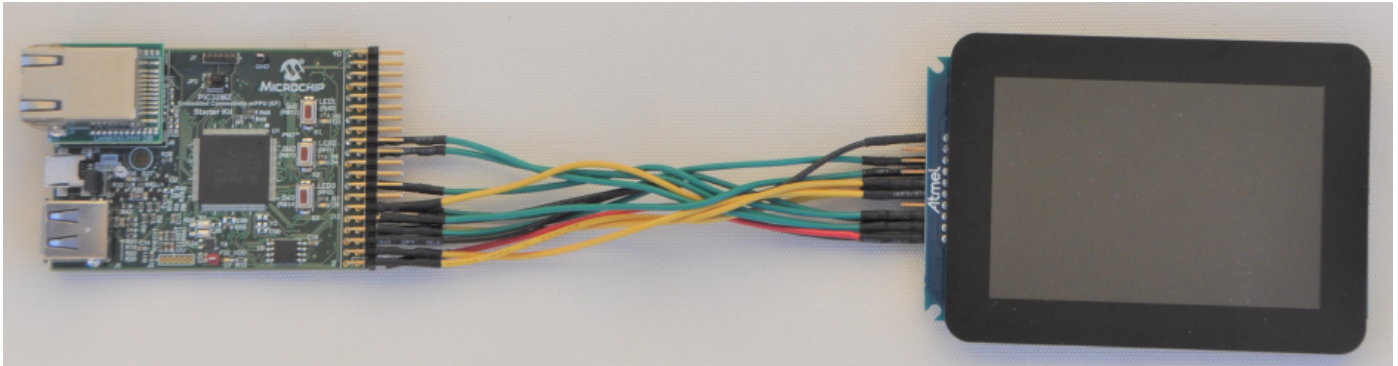
#### **Description**

This BSP is intended for the PIC32MZ Embedded Connectivity (EF) Starter Kit the maXTouch Xplained Pro 3.5" display. The following figure illustrates the hardware configuration.



**Note:**

Refer to the `aria_quickstart` demonstration for detailed information on pin connections between the starter kit and the display.



### ***pic32mz\_ef\_sk+meb2***

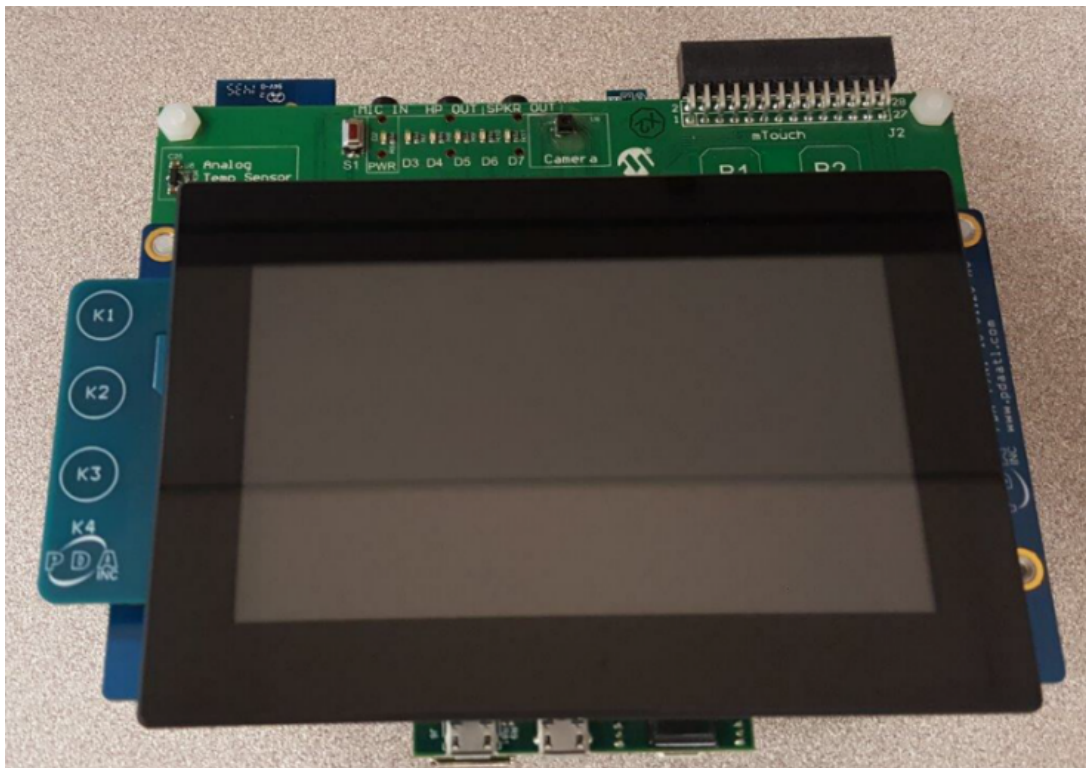
PIC32MZ EF Starter Kit plus MEB II BSP.

#### **Description**

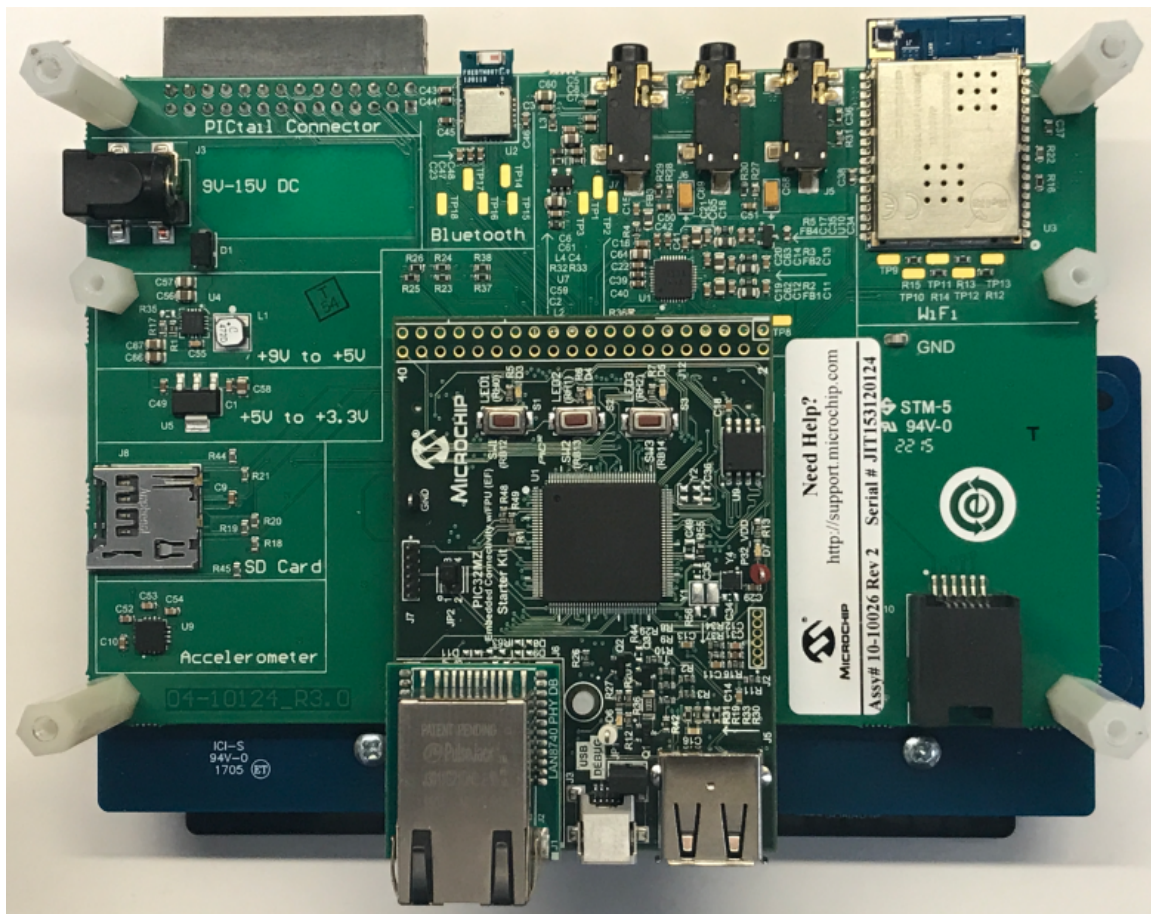
This BSP is intended for the Multimedia Expansion Board II (MEB II) connected to the PIC32MZ Embedded Connectivity with Floating Point Unit (EF) Starter Kit.

The following figure illustrates the hardware configuration.

**Top View**



Bottom View



### ***pic32mz\_ef\_sk+meb2+wgva***

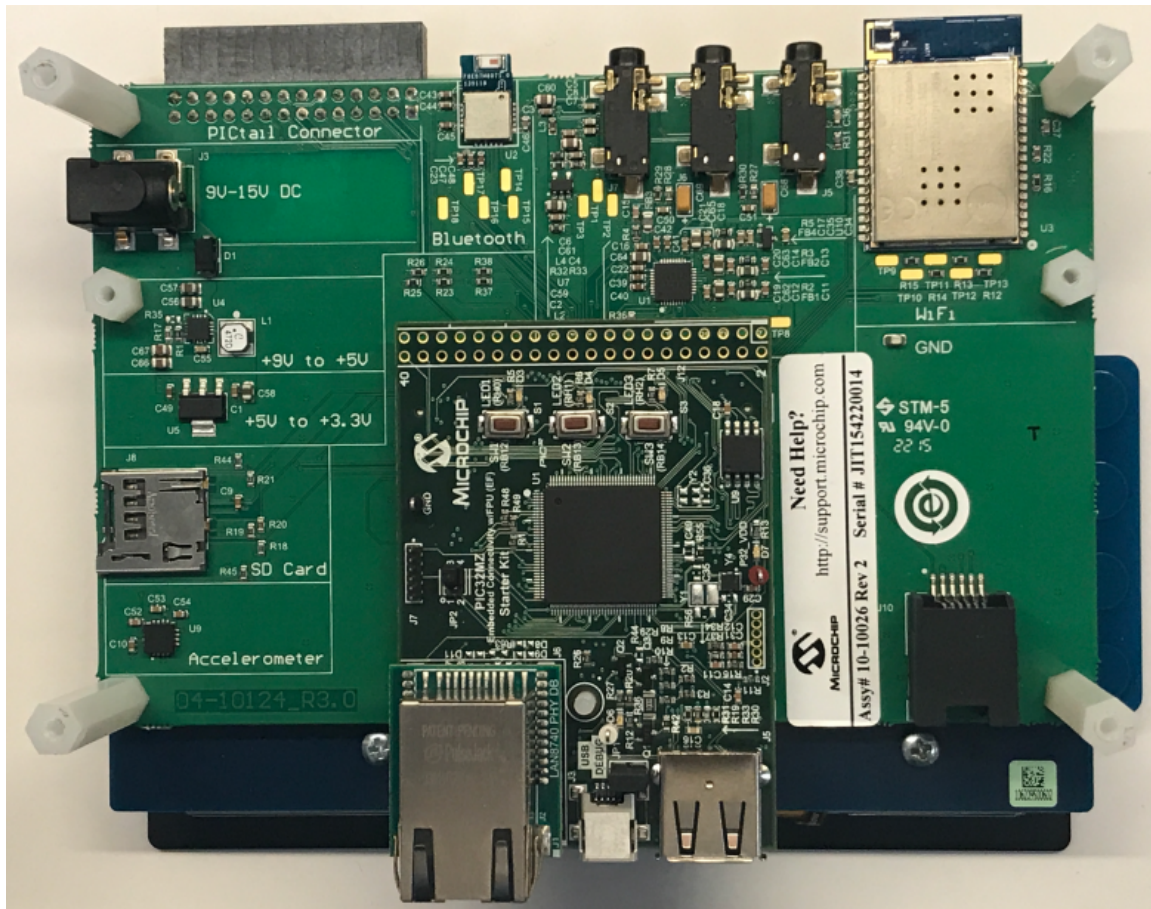
PIC32MZ EF Starter Kit plus MEB II and 5" WVGA PDA Display Board BSP.

### **Description**

This BSP is intended for the Multimedia Expansion Board II (MEB II) with the High-Performance WVGA Display Module with maXTouch connected to the PIC32MZ Embedded Connectivity with Floating Point Unit (EF) Starter Kit.

Bottom View





### **pic32mz\_ef\_sk+s1d\_pictail+vga**

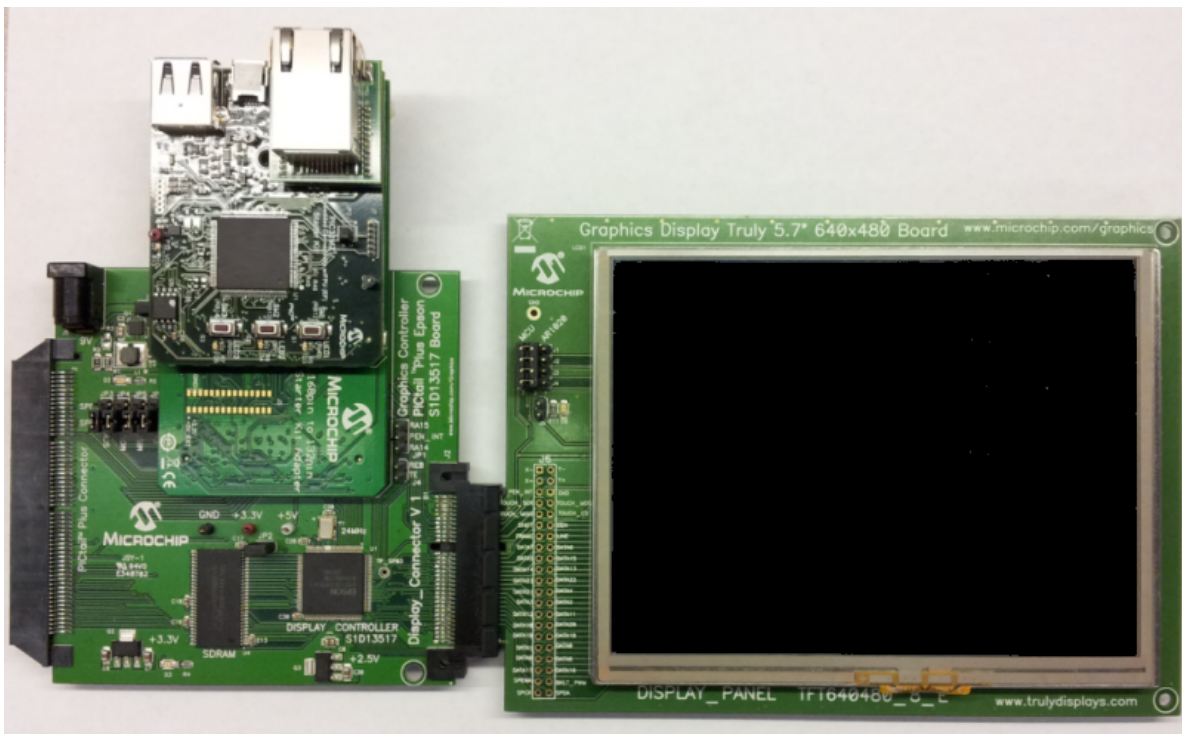
PIC32MZ EF Starter Kit plus Graphics Controller PICTail Plus Epson S1D13517 Daughter Board with Graphics Display Truly 5.7" 640x480 Board.

### **Description**

This BSP is intended for the Graphics Controller PICTail Plus Epson S1D13517 Daughter Board with the Graphics Display Truly 5.7" 640x480 Board connected to the PIC32MZ Embedded Connectivity with Floating Point Unit (EF) Starter Kit.

The following figure illustrates the hardware configuration.





### ***pic32mz\_ef\_sk+s1d\_pictail+wqvga***

PIC32MZ EF Starter Kit plus Graphics Controller PICtail Plus Epson S1D13517 Daughter Board with Graphics Display Powertip 4.3" 480x272 Board BSP.

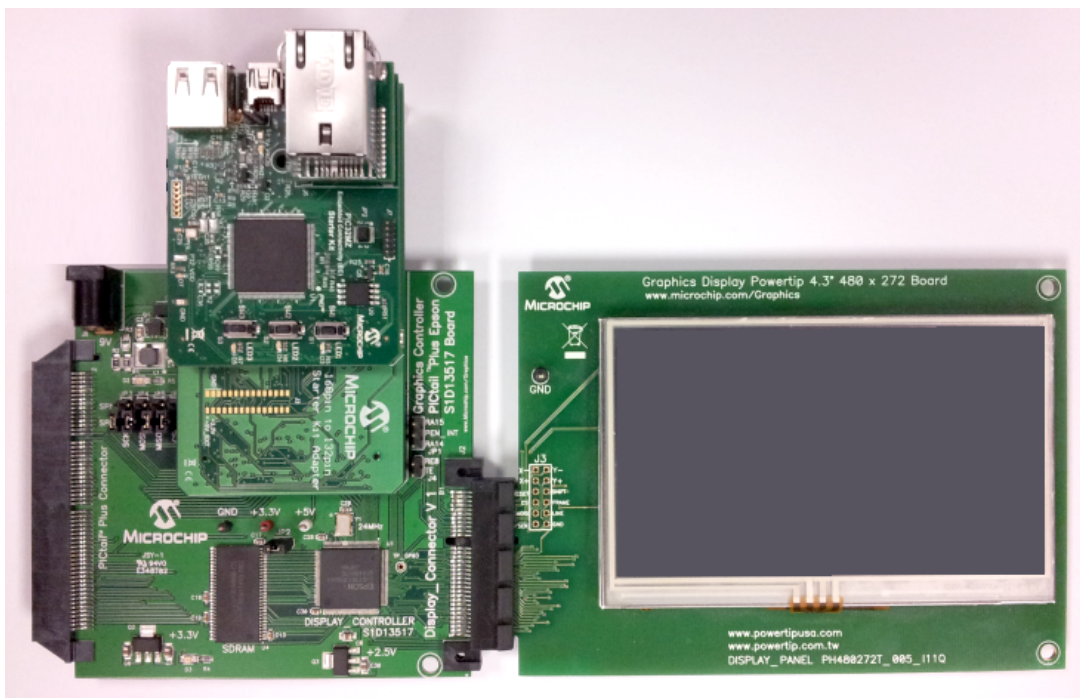
### **Description**

This BSP is intended for the Graphics Controller PICtail Plus Epson S1D13517 Daughter Board with the Graphics Display Powertip 4.3" 480x272 Board connected to the PIC32MZ Embedded Connectivity with Floating Point Unit (EF) Starter Kit with the PIC32MZ Starter Kit Adapter Board.



**Note:** The starter kit shown in the following figure is the PIC32MZ EC Starter Kit. The PIC32MZ EC and PIC32MZ EF starter kits are identical with the exception of the on-board device, so the hardware configuration is the same regardless of which starter kit is used.

The following figure illustrates the hardware configuration.



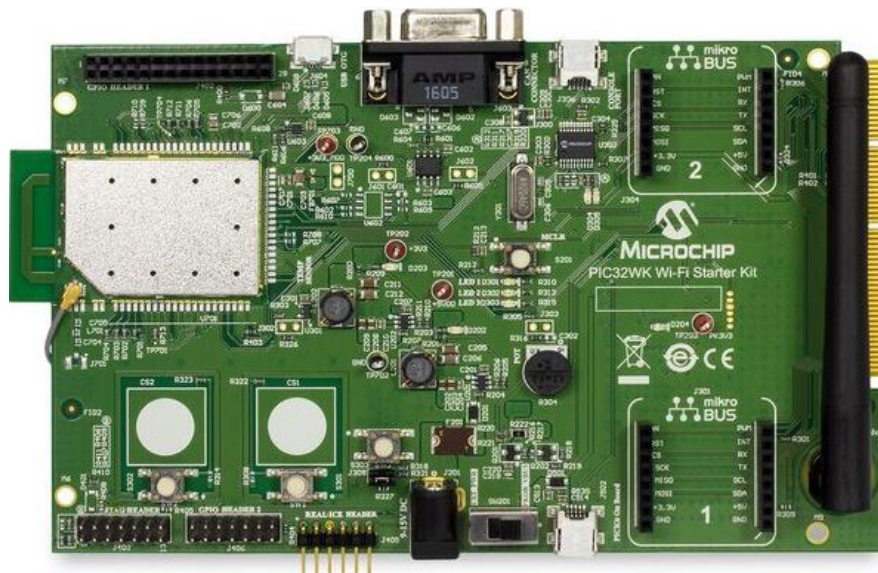
## ***pic32wk\_gbp\_gpd\_sk+module***

PIC32WK Wi-Fi Starter Kit BSP.

### **Description**

This BSP is intended for the PIC32WK Wi-Fi Starter Kit.

The following figure illustrates the hardware configuration.



## **Library Interface**

Provides information on BSP functions, structs, types, and macros.

### **Custom Named GPIO Macros**

Name	Description
<a href="#">custom_gpio_nameOff</a>	Turns off the custom-named GPIO pin.
<a href="#">custom_gpio_nameOn</a>	Turns on the custom-named GPIO pin.
<a href="#">custom_gpio_nameStateGet</a>	Gets the current value of the custom-named GPIO pin.
<a href="#">custom_gpio_nameStateSet</a>	Sets a new value on the custom-named GPIO pin.
<a href="#">custom_gpio_nameToggle</a>	Toggles the custom-named GPIO pin.
<a href="#">custom_gpio_pin_PIN</a>	Identifies the bit position within the port channel associated with a custom-named GPIO pin.
<a href="#">custom_gpio_pin_PIN_MASK</a>	Identifies the bit mask within the port channel associated with a custom-named GPIO pin.
<a href="#">custom_gpio_pin_PORT</a>	Identifies the port channel associated with a custom-named GPIO pin.

### **Custom Named LED Macros**

Name	Description
<a href="#">custom_led_nameOff</a>	Turns off the custom-named LED.
<a href="#">custom_led_nameOn</a>	Turns on the custom-named LED.
<a href="#">custom_led_nameStateGet</a>	Gets the current state of the custom-named LED.
<a href="#">custom_led_nameToggle</a>	Toggles the custom-named LED.

### **Custom Named Switch State Macros**


Name	Description
<a href="#">custom_switch_nameStateGet</a>	Gets the current value of the custom-named switch.

### **Data Types and Constants**






Name	Description
<a href="#">BSP_LED</a>	Defines the names of the LEDs available on the selected board.

	<a href="#">BSP_SWITCH_STATE</a>	Defines possible states of the switches on this board.
	<a href="#">BSP_LED_STATE</a>	Enumerates the supported LED states.
	<a href="#">BSP_SWITCH</a>	Defines the switches available on this board.
	<a href="#">BSP_OSC_FREQUENCY</a>	Defines the frequency value of crystal/oscillator used on the board


## Initialization Functions

	Name	Description
	<a href="#">BSP_Initialize</a>	Performs the necessary actions to initialize a board.

## LED Control Functions

	Name	Description
	<a href="#">BSP_LEDOff</a>	Turns OFF the specified LED.
	<a href="#">BSP_LEDOn</a>	Turns on the specified LED.
	<a href="#">BSP_LEDStateGet</a>	Returns the current state of the LED.
	<a href="#">BSP_LEDStateSet</a>	Controls the state of the LED.
	<a href="#">BSP_LEDToggle</a>	Toggles the current state of the LED.

## Switch State Functions

	Name	Description
	<a href="#">BSP_SwitchStateGet</a>	Returns the current state of the specified switch.

## Description

The header file for each BSP library uses the file named `bsp_config.h`.

## Initialization Functions

### BSP\_Initialize Function

Performs the necessary actions to initialize a board.

#### File

`bsp_help.h`

#### C

```
void BSP_Initialize();
```

#### Returns

None.

#### Description

This function initializes the LED and Switch ports on the board. This function must be called by the user before using any APIs present on this BSP.

#### Remarks

None.

#### Preconditions

None.

#### Example

```
//Initialize the BSP
BSP_Initialize();
```

#### Function

```
void BSP_Initialize ( void )
```

## LED Control Functions



## BSP\_LEDOff Function

Turns OFF the specified LED.

### File

bsp\_help.h

### C

```
void BSP_LEDOff(BSP_LED led);
```

### Returns

None.

### Description

This function turns OFF the specified LED.

### Remarks

None.

### Preconditions

[BSP\\_Initialize\(\)](#) should have been called.

### Example

```
// Turn off LED1 on the board
BSP_LEDOff(BSP_LED_1);
```

### Parameters

Parameters	Description
led	The name of the LED to turn off.

### Function

```
void BSP_LEDOff ( BSP_LED led );
```

## BSP\_LEDOn Function

Turns on the specified LED.

### File

bsp\_help.h

### C

```
void BSP_LEDOn(BSP_LED led);
```

### Returns

None.

### Description

This function turns on the specified LED.

### Remarks

None.

### Preconditions

[BSP\\_Initialize\(\)](#) should have been called.

### Example

```
// Turn on LED1 on the board
BSP_LEDOn(BSP_LED_1);
```

## Parameters

Parameters	Description
led	The name of the LED to turn on

## Function

```
void BSP_LEDOn ( BSP_LED led );
```

## BSP\_LEDStateGet Function

Returns the current state of the LED.

## File

bsp\_help.h

## C

```
BSP_LED_STATE BSP_LEDStateGet (BSP_LED led);
```

## Returns

BSP\_LED\_STATE\_OFF If the LED is off. BSP\_LED\_STATE\_ON If the LED is on.

## Description

This function returns the current state of the LED.

## Remarks

None.

## Preconditions

[BSP\\_Initialize\(\)](#) should have been called.

## Example

```
// Check if LED2 is off.
if (BSP_LED_STATE_OFF == BSP_LEDStateGet (BSP_LED_2)
{
    // Turn the LED on.
    BSP_LEDStateSet (BSP_LED_2, BSP_LED_STATE_ON);
}
```

## Parameters

Parameters	Description
led	The name of the LED to whose state to obtain.

## Function

```
BSP_LED_STATE BSP_LEDStateGet ( BSP_LED led );
```

## BSP\_LEDStateSet Function

Controls the state of the LED.

## File

bsp\_help.h

## C

```
void BSP_LEDStateSet (BSP_LED led, BSP_LED_STATE state);
```

## Returns

None.

## Description

This function allows the caller to specify the state of the LED.

## Remarks

None.

## Preconditions

`BSP_Initialize()` should have been called.

## Example

```
// Turn on LED1 on the board
BSP_LEDStateSet(BSP_LED_1, BSP_LED_STATE_ON);

// Turn off LED2 on the board
BSP_LEDStateSet(BSP_LED_2, BSP_LED_STATE_OFF);
```

## Parameters

Parameters	Description
led	The name of the LED to control.
state	The state to which the LED will be set: <code>BSP_LED_STATE_OFF</code> turns the LED off. <code>BSP_LED_STATE_ON</code> turns the LED on.

## Function

```
void BSP_LEDStateSet ( BSP_LED led, BSP_LED_STATE state );
```

## BSP\_LEDToggle Function

Toggles the current state of the LED.

## File

bsp\_help.h

## C

```
void BSP_LEDToggle(BSP_LED led);
```

## Returns

None.

## Description

This function toggles the current state of the LED.

- If the current state is `BSP_LED_STATE_ON` it changes to `BSP_LED_STATE_OFF`.
- If the current state is `BSP_LED_STATE_OFF` it changes to `BSP_LED_STATE_ON`.

## Remarks

None.

## Preconditions

`BSP_Initialize()` should have been called.

## Example

```
// Toggle LED1 on the board.
BSP_LEDToggle(BSP_LED_1);

// Toggle LED2 on the board.
BSP_LEDToggle(BSP_LED_2);

// Toggle state of LED3.
BSP_LEDToggle(BSP_LED_3);
```

## Parameters

Parameters	Description
led	The name of LED to toggle.

## Function

```
void BSP_LEDToggle ( BSP_LED led );
```



## Switch State Functions

### BSP\_SwitchStateGet Function

Returns the current state of the specified switch.

#### File

bsp\_help.h

#### C

```
BSP_SWITCH_STATE BSP_SwitchStateGet(BSP_SWITCH bspSwitch);
```

#### Returns

BSP\_SWITCH\_STATE\_PRESSED if the switch is currently pressed. BSP\_SWITCH\_STATE\_RELEASED if the switch is not currently pressed (i.e. is released).

#### Description

This function returns the current state of the specified switch.

#### Remarks

Unless otherwise documented by the selected BSP, this function does not perform switch debouncing.

#### Preconditions

[BSP\\_Initialize\(\)](#) should have been called.

#### Example

```
// Check the state of the switch.
if(BSP_SWITCH_STATE_PRESSED == BSP_SwitchStateGet(BSP_SWITCH_1))
{
    // This means that Switch 1 on the board is pressed.
}
```

#### Parameters

Parameters	Description
switch	The name of the switch whose state to obtain.

#### Function

```
BSP_SWITCH_STATE BSP_SwitchStateGet ( BSP_SWITCH switch );
```

## Custom Named GPIO Macros

### custom\_gpio\_nameOff Macro

Turns off the custom-named GPIO pin.

#### File

bsp\_help.h

#### C

```
#define custom_gpio_nameOff
```

#### Description

Custom-Named GPIO Off Macro

This is a function-like macro that turns off a custom-named GPIO pin.

#### Remarks

There is no actual macro or function named "custom\_gpio\_nameOff". It is given here as a place holder to indicate that the BSP can define custom-named macros to turn off GPIO pins with custom names for any port pins set to the "GPIO\_OUT" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named GPIO off macros with names that matching the names of GPIO pins provided

on the selected board. The user can add additional custom GPIO off macros by selecting the "GPIO\_OUT" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

## custom\_gpio\_nameOn Macro

Turns on the custom-named GPIO pin.

### File

bsp\_help.h

### C

```
#define custom_gpio_nameOn
```

### Description

Custom-Named GPIO On Macro

This is a function-like macro that turns on a custom-named GPIO pin.

### Remarks

There is no actual macro or function named "custom\_gpio\_nameOn". It is given here as a place holder to indicate that the BSP can define custom-named macros to turn on GPIO pins with custom names for any port pins set to the "GPIO\_OUT" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named GPIO on macros with names that matching the names of GPIO pins provided on the selected board. The user can add additional custom GPIO on macros by selecting the "GPIO\_OUT" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

## custom\_gpio\_nameStateGet Macro

Gets the current value of the custom-named GPIO pin.

### File

bsp\_help.h

### C

```
#define custom_gpio_nameStateGet
```

### Description

Custom-Named GPIO Get Macro

This is a function-like macro that gets the current value of a custom-named GPIO pin.

### Remarks

There is no actual macro or function named "custom\_gpio\_nameStateGet". It is given here as a place holder to indicate that the BSP can define custom-named macros to get the current state of GPIO pins with custom names for any port pins set to the "GPIO\_IN" or "GPIO\_OUT" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named GPIO state-get macros with names that matching the names of GPIO pins provided on the selected board. The user can add additional custom GPIO state-get macros by selecting the "GPIO\_IN" or "GPIO\_OUT" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

## custom\_gpio\_nameStateSet Macro

Sets a new value on the custom-named GPIO pin.

### File

bsp\_help.h

### C

```
#define custom_gpio_nameStateSet(state)
```

### Description

Custom-Named GPIO Set Macro

This is a function-like macro that sets a new value on a custom-named GPIO pin.

## Remarks

There is no actual macro or function named "custom\_gpio\_nameStateSet". It is given here as a place holder to indicate that the BSP can define custom-named macros to set the current state of GPIO pins with custom names for any port pins set to the "GPIO\_OUT" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named GPIO state-set macros with names that matching the names of GPIO pins provided on the selected board. The user can add additional custom GPIO state-set macros by selecting the "GPIO\_OUT" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

## Parameters

Parameters	Description
state	A boolean value. true if the pin is to be turned on. false if the pin is to be turned off.

## custom\_gpio\_nameToggle Macro

Toggles the custom-named GPIO pin.

## File

bsp\_help.h

## C

```
#define custom_gpio_nameToggle
```

## Description

Custom-Named GPIO Toggle Macro

This is a function-like macro that toggles a custom-named GPIO pin.

## Remarks

There is no actual macro or function named "custom\_gpio\_nameToggle". It is given here as a place holder to indicate that the BSP can define custom-named macros to toggle GPIO pins with custom names for any port pins set to the "GPIO\_OUT" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named GPIO pin toggle macros with names that matching the names of GPIO pins provided on the selected board. The user can add additional custom GPIO pin toggle macros by selecting the "GPIO\_OUT" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

## custom\_gpio\_pin\_PIN Macro

Identifies the bit position within the port channel associated with a custom-named GPIO pin.

## File

bsp\_help.h

## C

```
#define custom_gpio_pin_PIN PORTS_BIT_POS_Y
```

## Description

Custom GPIO Pin Port Channel Bit Position

This macro identifies the bit position within the port channel associated with a custom-named GPIO pin. This macro is intended for use with ports system service functions.

## Remarks

There is no actual macro or function named "custom\_gpio\_pin\_PIN". It is given here as a place holder to indicate that the BSP can define custom-named macros to identify the bit position within the port channel associated with the custom-named pin.

This macro will actually be defined in the system\_config.h header for the current configuration.

## custom\_gpio\_pin\_PIN\_MASK Macro

Identifies the bit mask within the port channel associated with a custom-named GPIO pin.

## File

bsp\_help.h



**C**

```
#define custom_gpio_pin_PIN_MASK (0x1 << 0)
```

**Description**

Custom GPIO Pin Mask

This macro identifies the bit mask within the port channel associated with a custom-named GPIO pin. This macro is intended for use with ports system service functions.

**Remarks**

There is no actual macro or function named "custom\_gpio\_pin\_PIN\_MASK". It is given here as a place holder to indicate that the BSP can define custom-named macros to identify the bit position within the port channel associated with the custom-named pin.

This macro will actually be defined in the system\_config.h header for the current configuration.

**custom\_gpio\_pin\_PORT Macro**

Identifies the port channel associated with a custom-named GPIO pin.

**File**

bsp\_help.h

**C**

```
#define custom_gpio_pin_PORT PORT_CHANNEL_x
```

**Description**

Custom GPIO Pin Port Channel

This macro identifies the port channel associated with a custom-named GPIO pin. This macro is intended for use with ports system service functions.

**Remarks**

There is no actual macro or function named "custom\_gpio\_pin\_PORT". It is given here as a place holder to indicate that the BSP can define custom-named macros to identify the port channel associated with the custom-named pin.

This macro will actually be defined in the system\_config.h header for the current configuration.

**Custom Named LED Macros****custom\_led\_nameOff Macro**

Turns off the custom-named LED.

**File**

bsp\_help.h

**C**

```
#define custom_led_nameOff
```

**Description**

Custom-Named LED Off Macro

This is a function-like macro that turns off a custom-named LED.

**Remarks**

There is no actual macro or function named "custom\_led\_nameOff". It is given here as a place holder to indicate that the BSP can define custom-named macros to turn off LEDs with custom names for any port pins set to the "LED\_AH" or "LED\_AL" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named LED off macros with names that matching the names of LEDs provided on the selected board. The user can add additional custom LED off macros by selecting the "LED\_AH" or "LED\_AL" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

## custom\_led\_nameOn Macro

Turns on the custom-named LED.

### File

bsp\_help.h

### C

```
#define custom_led_nameOn
```

### Description

Custom-Named LED On Macro

This is a function-like macro that turns on a custom-named LED.

### Remarks

There is no actual macro or function named "custom\_led\_nameOn". It is given here as a place holder to indicate that the BSP can define custom-named macros to turn on LEDs with custom names for any port pins set to the "LED\_AH" or "LED\_AL" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named LED on macros with names that matching the names of LEDs provided on the selected board. The user can add additional custom LED on macros by selecting the "LED\_AH" or "LED\_AL" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

## custom\_led\_nameStateGet Macro

Gets the current state of the custom-named LED.

### File

bsp\_help.h

### C

```
#define custom_led_nameStateGet
```

### Description

Custom-Named LED State Get Macro

This is a function-like macro that gets the current state of a custom named LED.

### Remarks

There is no actual macro or function named "custom\_led\_nameStateGet". It is given here as a place holder to indicate that the BSP can define custom-named macros to get the current state LEDs with custom names for any port pins set to the "LED\_AH" or "LED\_AL" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named LED get-state macros with names that matching the names of LEDs provided on the selected board. The user can add additional custom LED get- state macros by selecting the "LED\_AH" or "LED\_AL" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

## custom\_led\_nameToggle Macro

Toggles the custom-named LED.

### File

bsp\_help.h

### C

```
#define custom_led_nameToggle
```

### Description

Custom-Named LED Toggle Macro

This is a function-like macro that toggles a custom-named LED.

### Remarks

There is no actual macro or function named "custom\_led\_nameToggle". It is given here as a place holder to indicate that the BSP can define custom-named macros to toggle LEDs with custom names for any port pins set to the "LED\_AH" or "LED\_AL" function in the Pin Settings

configuration page. By default, the selected BSP may define a number of custom-named LED toggle macros with names that matching the names of LEDs provided on the selected board. The user can add additional custom LED toggle macros by selecting the "LED\_AH" or "LED\_AL" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

Custom Named Switch State Macros

custom\_switch\_nameStateGet Macro

Gets the current value of the custom-named switch.

File

bsp\_help.h

C

```
#define custom_switch_nameStateGet
```

Description

Custom-Named Switch Get Macro

This is a function-like macro that gets the current value of a custom-named switch.

Remarks

There is no actual macro or function named "custom\_switch\_nameStateGet". It is given here as a place holder to indicate that the BSP can define custom-named macros to get the current state of switches with custom names for any port pins set to the "SWITCH" function in the Pin Settings configuration page. By default, the selected BSP may define a number of custom-named switch state-get macros with names that matching the names of switches provided on the selected board. The user can add additional custom switch state-get macros by selecting the "SWITCH" function and defining a custom name for any desired pins in the MHC Pin Settings page.

This macro will actually be defined in the system\_config.h header for the current configuration.

Data Types and Constants

BSP\_LED Enumeration

Defines the names of the LEDs available on the selected board.

File

bsp\_help.h

C

```
typedef enum {
    BSP_LED_1,
    BSP_LED_2,
    BSP_LED_3,
    custom_led_name
} BSP_LED;
```

Members

Members	Description
BSP_LED_1	LED 1
BSP_LED_2	LED 2
BSP_LED_3	LED 3
custom_led_name	Custom LED Name (see Remarks).

Description

BSP LED Name.

This enumeration defines the names of the LEDs available on the selected board.

Remarks

There is no actual value named "custom\_led\_name" in the BSP\_LED enum. It is listed here as a place holder to indicate that the MHC can define custom names for any port pins set to the "LED\_AH" or "LED\_AL" function in the Pin Settings configuration page and those and those names will



then become part of this enum definition. By default, the selected BSP may define a number of custom LED names to match the names of LEDs provided on the selected board. The user can add additional custom switch names by selecting the "LED\_AH" or "LED\_AL" function and defining a custom name for any desired pins in the MHC Pin Settings page.

## BSP\_SWITCH\_STATE Enumeration

Defines possible states of the switches on this board.

### File

bsp\_help.h

### C

```
typedef enum {  
    BSP_SWITCH_STATE_PRESSED,  
    BSP_SWITCH_STATE_RELEASED  
} BSP_SWITCH_STATE;
```

### Members

Members	Description
BSP_SWITCH_STATE_PRESSED	Switch pressed
BSP_SWITCH_STATE_RELEASED	Switch not pressed

### Description

BSP Switch state.

This enumeration defines the possible states of the switches on the selected board.

### Remarks

None.

## BSP\_LED\_STATE Enumeration

Enumerates the supported LED states.

### File

bsp\_help.h

### C

```
typedef enum {  
    BSP_LED_STATE_OFF,  
    BSP_LED_STATE_ON  
} BSP_LED_STATE;
```

### Members

Members	Description
BSP_LED_STATE_OFF	LED State is on
BSP_LED_STATE_ON	LED State is off

### Description

LED State

This enumeration defines the supported LED states.

### Remarks

None.

## BSP\_SWITCH Enumeration

Defines the switches available on this board.

### File

bsp\_help.h

C

```
typedef enum {
    BSP_SWITCH_1,
    BSP_SWITCH_2,
    BSP_SWITCH_3,
    custom_switch_name
} BSP_SWITCH;
```

Members

Members	Description
BSP_SWITCH_1	SWITCH 1
BSP_SWITCH_2	SWITCH 2
BSP_SWITCH_3	SWITCH 3
custom_switch_name	Custom Switch Name (see Remarks).

Description

BSP Switch.

This enumeration defines the switches available on this board.

Remarks

There is no actual value named "custom\_switch\_name" in the BSP\_SWITCH enum. It is listed here as a place holder to indicate that the MHC can define custom names for any port pins set to the "SWITCH" function in the Pin Settings configuration page and those and those names will then become part of this enum definition. By default, the selected BSP will define a number of custom switch names to match the names of switches provided on the selected board. The user can add additional custom switch names by selecting the "SWITCH" function and defining a custom name for any desired pins in the MHC Pin Settings page.

BSP\_OSC\_FREQUENCY Macro

Defines the frequency value of crystal/oscillator used on the board

File

bsp\_help.h

C

```
#define BSP_OSC_FREQUENCY
```

Description

Oscillator Frequency

This macro defines the frequency value of the crystal/oscillator used on the board.

## Index

### B

#### Board Support Package

- bt\_audio\_dk 7
- pic32mx\_125\_sk 20
- pic32mx\_bt\_sk 22
- pic32mx\_eth\_sk 22
- pic32mx\_eth\_sk2 23
- pic32mx\_usb\_sk2 24
- pic32mx\_usb\_sk2+lcc\_pictail+qvga 25
- pic32mx\_usb\_sk2+lcc\_pictail+wqvga 25
- pic32mx\_usb\_sk2+meb 26
- pic32mx\_usb\_sk2+s1d\_pictail+vga 26
- pic32mx\_usb\_sk2+s1d\_pictail+wqvga 27
- pic32mx\_usb\_sk2+s1d\_pictail+vvga 27
- pic32mx\_usb\_sk2+ssd\_pictail+qvga 27
- pic32mx\_usb\_sk3 28
- pic32mx270f512l\_pim+bt\_audio\_dk 32
- pic32mx460\_pim+e16 34
- pic32mx470\_pim+e16 36
- pic32mx795\_pim+e16 36
- pic32mz\_ec\_pim+bt\_audio\_dk 48
- pic32mz\_ec\_pim+e16 49
- pic32mz\_ec\_sk 50
- pic32mz\_ec\_sk+meb2 50
- pic32mz\_ec\_sk+meb2+vvga 51
- pic32mz\_ec\_sk+s1d\_pictail+wqvga 53

#### Board Support Packages 7

#### Board Support Packages Help 3

#### BSP\_Initialize function 64

#### BSP\_LED enumeration 73

#### BSP\_LED\_STATE enumeration 74

#### BSP\_LEDOff function 65

#### BSP\_LEDOn function 65

#### BSP\_LEDStateGet function 66

#### BSP\_LEDStateSet function 66

#### BSP\_LEDToggle function 67

#### BSP\_OSC\_FREQUENCY macro 75

#### BSP\_SWITCH enumeration 74

#### BSP\_SWITCH\_STATE enumeration 74

#### BSP\_SwitchStateGet function 68

#### bt\_audio\_dk 7

#### bt\_audio\_dk+4642 8

#### bt\_audio\_dk+ak4954 9

#### bt\_audio\_dk+ak7755 10

#### bt\_audio\_dk+bm64 11

#### bt\_audio\_dk+bm64+ak4954 12

#### bt\_sk+4642 12

#### bt\_sk+4954 13

#### bt\_sk+ak7755 14

### C

#### chipkit\_wf32 15

#### chipkit\_wifire 16

#### Creating Custom Initial Board Configurations 6

#### custom\_gpio\_nameOff macro 68

#### custom\_gpio\_nameOn macro 69

#### custom\_gpio\_nameStateGet macro 69

#### custom\_gpio\_nameStateSet macro 69

#### custom\_gpio\_nameToggle macro 70

#### custom\_gpio\_pin\_PIN macro 70

#### custom\_gpio\_pin\_PIN\_MASK macro 70

#### custom\_gpio\_pin\_PORT macro 71

#### custom\_led\_nameOff macro 71

#### custom\_led\_nameOn macro 72

#### custom\_led\_nameStateGet macro 72

#### custom\_led\_nameToggle macro 72

#### custom\_switch\_nameStateGet macro 73

#### Customizing a BSP 4

### E

#### Extending the BSP Library Interface 4

### I

#### Introduction 3

### L

#### Library Interface 63

### M

#### Modifying the BSP Library Source Code 6

### P

#### pic32\_gdb\_ef 17

#### pic32mk\_gp\_db 17

#### pic32mk\_gp\_db+wqvga\_mxt 19

#### pic32mk\_gp\_db+vvga\_mxt 20

#### pic32mx\_125\_sk 20

#### pic32mx\_125\_sk+lcc\_pictail+qvga 21

#### pic32mx\_270f512l\_pim+ bt\_audio\_dk+ak4642 33

#### pic32mx\_bt\_sk 22

#### pic32mx\_eth\_sk 22

#### pic32mx\_eth\_sk2 23

#### pic32mx\_pcap\_db 23

#### pic32mx\_usb\_digital\_audio\_ab 23

#### pic32mx\_usb\_sk2 24

#### pic32mx\_usb\_sk2+lcc\_pictail+qvga 25

#### pic32mx\_usb\_sk2+lcc\_pictail+wqvga 25

#### pic32mx\_usb\_sk2+meb 26

#### pic32mx\_usb\_sk2+s1d\_pictail+vga 26

#### pic32mx\_usb\_sk2+s1d\_pictail+wqvga 27

#### pic32mx\_usb\_sk2+s1d\_pictail+vvga 27

#### pic32mx\_usb\_sk2+ssd\_pictail+qvga 27

#### pic32mx\_usb\_sk3 28

#### pic32mx\_usb\_sk3+lcc\_pictail+wqvga 29

#### pic32mx\_xlp\_sk 30

#### pic32mx270f512l\_pim+bt\_audio\_dk 32

#### pic32mx460\_pim+e16 34

#### pic32mx470\_curiosity 35

#### pic32mx470\_pim+e16 36

#### pic32mx795\_pim+e16 36

#### pic32mz\_da\_sk\_extddr 37

#### pic32mz\_da\_sk\_extddr+meb2 38

#### pic32mz\_da\_sk\_extddr+meb2+vvga 39

#### pic32mz\_da\_sk\_intddr 41

#### pic32mz\_da\_sk\_intddr+meb2 41



pic32mz\_da\_sk\_intddr+meb2+wvga 42  
pic32mz\_da\_sk\_noddr+meb2 44  
pic32mz\_da\_sk\_noddr+meb2+wvga 46  
pic32mz\_ec\_pim+bt\_audio\_dk 48  
pic32mz\_ec\_pim+e16 49  
pic32mz\_ec\_sk 50  
pic32mz\_ec\_sk+meb2 50  
pic32mz\_ec\_sk+meb2+wvga 51  
pic32mz\_ec\_sk+s1d\_pictail+vga 52  
pic32mz\_ec\_sk+s1d\_pictail+wqvga 53  
pic32mz\_ec\_sk+s1d\_pictail+wvga 54  
pic32mz\_ef\_curiosity 54  
pic32mz\_ef\_pim+bt\_audio\_dk 55  
pic32mz\_ef\_pim+bt\_audio\_dk+ak4642 56  
pic32mz\_ef\_pim+e16 57  
pic32mz\_ef\_sk 58  
pic32mz\_ef\_sk+maxtouch\_xplained\_pro\_3\_5 59  
pic32mz\_ef\_sk+meb2 59  
pic32mz\_ef\_sk+meb2+wvga 60  
pic32mz\_ef\_sk+s1d\_pictail+vga 61  
pic32mz\_ef\_sk+s1d\_pictail+wqvga 62  
pic32wk\_gbp\_gpd\_sk+module 63

## **U**

Using a BSP 3

## **V**

Volume II: Supported Hardware 2