# CE218136 – PSoC 6 MCU E-INK Display with CapSense (RTOS)

## Objective

This code example shows how to create a user-interface solution using an E-INK display and CapSense®.

## Overview

This code example demonstrates how to create a user-interface solution using an E-INK display with a CapSense slider and buttons. E-INK displays consume no power for image retention. Together with PSoC 6 MCU's CapSense touch sensing, an E-INK display can be used to create user interfaces that have "always-on" functionality.

This code example assumes that you are familiar with the PSoC 6 MCU and the PSoC Creator™ Integrated Design Environment (IDE). If you are new to PSoC 6 MCU, see the application note AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity.

This code example uses FreeRTOS. See PSoC 6 101: Lesson 1-4 FreeRTOS training video to learn how to create a PSoC 6 FreeRTOS project with PSoC Creator. Visit the FreeRTOS website for documentation and API references of FreeRTOS.

## Requirements

**Tool:** PSoC Creator 4.2; Peripheral Driver Library (PDL) 3.0.1

**Programming Language:** C (Arm® GCC 5.4.1)

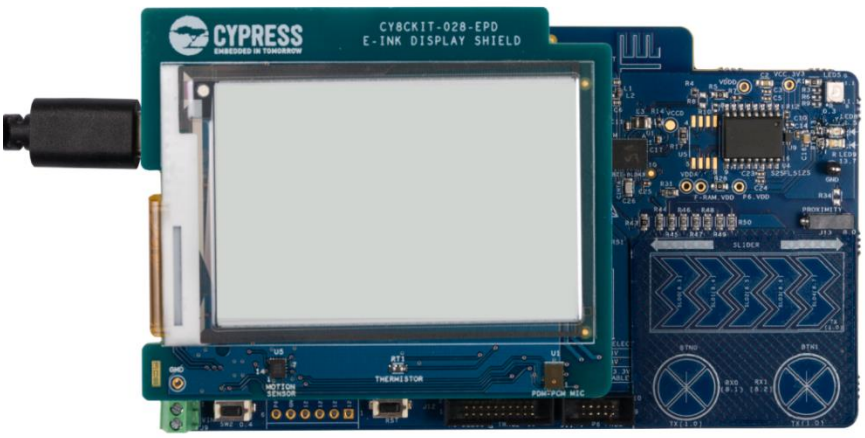
**Associated Parts:** All PSoC 6 MCUs

**Related Hardware:** CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

## Hardware Setup

Plug in the E-INK display shield on to the Pioneer Board as Figure 1 shows.

Figure 1. Hardware Setup



Set the switches and jumpers on the Pioneer Board as shown in Table 1.

Table 1. Switch and Jumper Selection

| Switch/Jumper | Position | Location |
|---|---|---|
| SW5 | 3.3 V | Front |
| SW6 | PSoC 6 BLE | Back |
| SW7 | $V_{DDD}$ / KitProg2 | Back |
| J8 | Installed | Back |

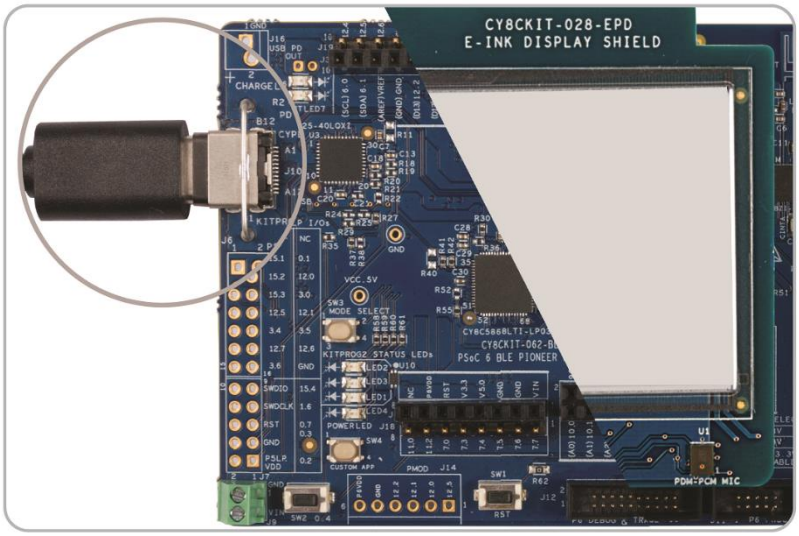
## Software Setup

Install the CY8CKIT-62-BLE PSoC 6 BLE Pioneer Kit software, which contains all the required software to evaluate this code example. No additional software setup is required.

## Operation

1. Connect the Pioneer Board to your PC using the provided USB cable through the USB connector (J10).

Figure 2. Connecting the USB Cable to the Pioneer Board




2. Program the Pioneer Board with the CE218133_EINK_CapSense project. See the CY8CKIT-062-BLE kit guide for details on how to program firmware into the device.

   The E-INK display refreshes and shows the startup screen for 3 seconds, followed by a menu that lists important information about the kit and associated software, as Figure 3 shows. LED9 (Red) turns ON if the E-INK display is not detected. In this case, check the connection between the E-INK Display Shield and the Pioneer Board, and then reset the Pioneer Board.

3. Use the CapSense slider and buttons to navigate the menu, as Figure 4 shows.

   Note that the display takes about a second to refresh the display following a touch input. LED8 (Orange) turns ON when the display is busy. Touch inputs are not processed when the display is busy. Because the main menu uses partial update for faster refreshes (for details, see the `cy_eink_update_t` parameter of the `Cy_EINK_ShowFrame` function in Appendix A, the selection arrow may have slight ghosting.

Figure 3. Main Menu

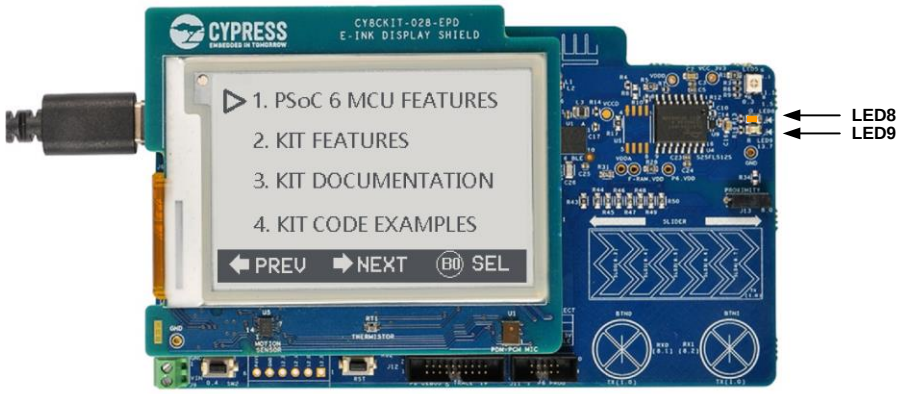
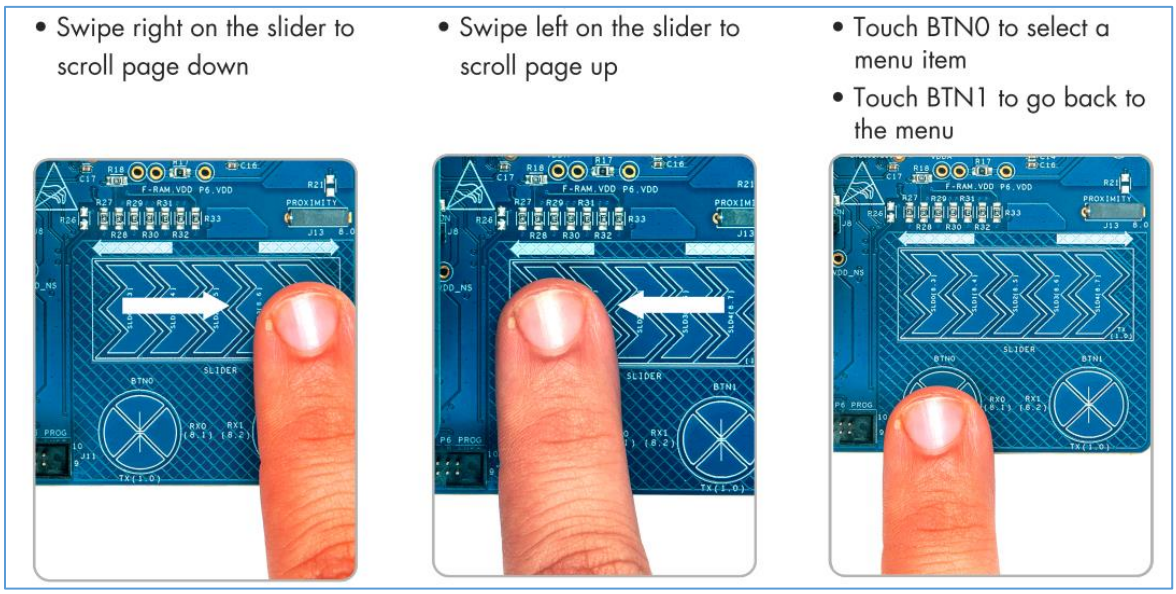



Figure 4. Menu Navigation Options




## Design and Implementation

E-INK (electronic ink) is a paper-like display technology, characterized by high contrast, wide viewing angles, and minimal standby power. Unlike conventional backlit, flat panel displays that emit light, E-INK displays reflect light like paper. This makes E-INK displays more comfortable to read, and provides a wider viewing angle than most light-emitting displays. Therefore, E-INK displays are comfortable to read even in sunlight.

This project uses a CY8CKIT-028-EPD E-INK Display Shield together with a Pioneer Board. The E-INK Shield has a 2.7-inch E-INK display with a resolution of 264×176 pixels.

For details on the Pioneer Board and E-INK Display Shield, see the Pioneer Kit Guide.

Figure 5, Figure 6, and Figure 7 show the PSoC Creator schematic of this code example.

Figure 5. TopDesign Schematic: CapSense and LED Indications

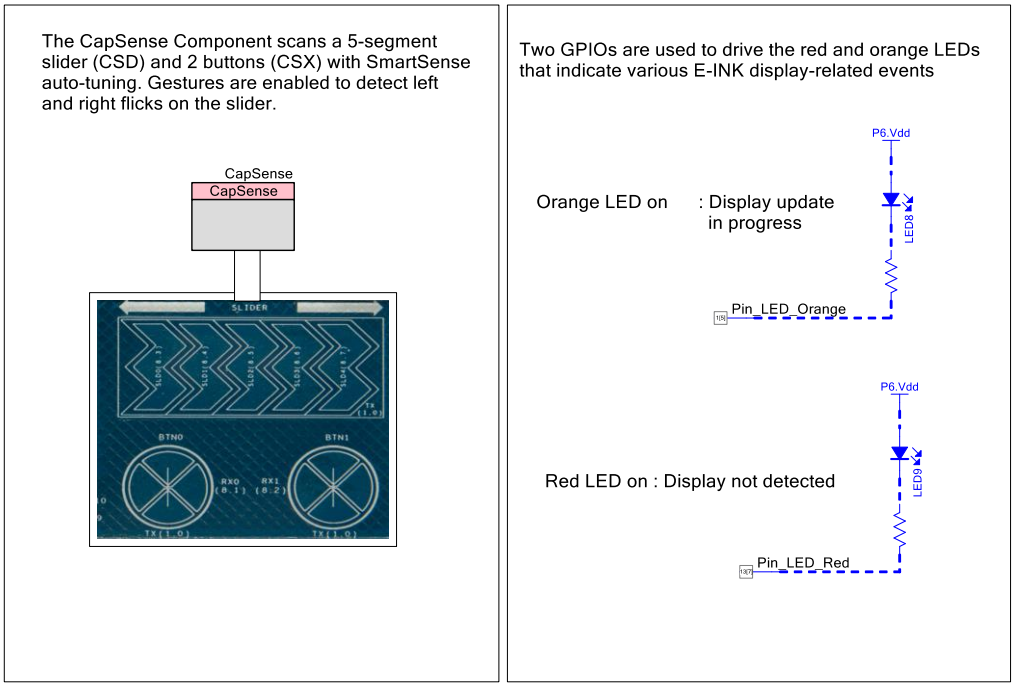



Figure 6. TopDesign Schematic: E-INK Display

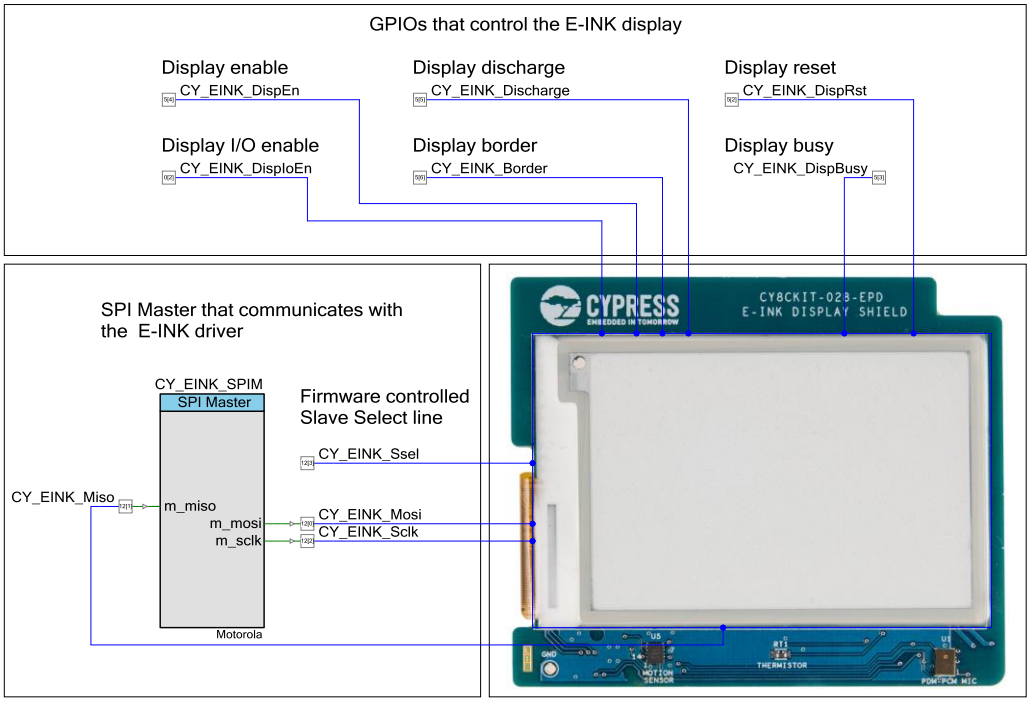
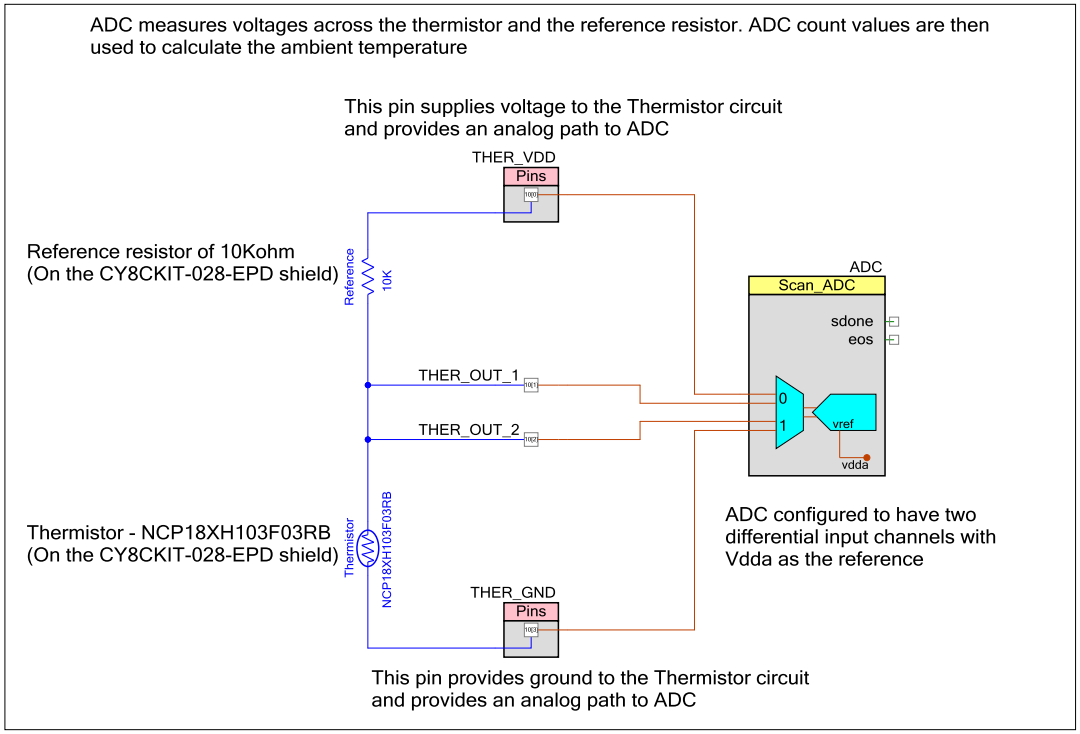


Figure 7. TopDesign Schematic: Temperature Compensation for E-INK Display




The E-INK display in CY8CKIT-028-EPD contains a basic driver IC that interfaces with the PSoC 6 MCU using a custom SPI interface. The driver converts a serial data stream into individual pixel data and generates the voltages required for the E-INK display. PSoC 6 MCU has low-level control of the E-INK display.

This code example contains the required peripheral functions for driving the E-INK display. However, the actual hardware driver functions are not covered in this document. See the E-INK display driver document for more details.

The PSoC 6 MCU controls the E-INK display's reset, enable, discharge and border pins. PSoC 6 MCU also reads the status of the display to determine whether the display is busy with a previous operation. A load switch on CY8CKIT-028-EPD, which is controlled by the PSoC 6 MCU device, can be used to turn the display ON/OFF. A voltage level translator is connected between the E-INK display and PSoC 6 MCU GPIOs so that PSoC 6 MCU can operate with variable $V_{DD}$. The enable input of the voltage level translator is also connected to a PSoC 6 MCU GPIO so that PSoC 6 MCU can disable the level translator to reduce power consumption when the E-INK display is not used.

In this project, PSoC 6 MCU scans a CapSense slider and two buttons for user input. Based on the user input, the E-INK display is updated to scroll through menu items to change information pages, and to move back and forth between the top-level menu and information pages as Figure 8 shows.

The project consists of the following files:

- *FreeRTOSConfig.h* contains the FreeRTOS settings and configuration. Non-default settings are explained with in-line comments.
- *main_cm4.c* contains the main function, which is the entry point and execution of the firmware application. The main function sets up user tasks and then starts the RTOS scheduler.
- *main_cm0p.c* contains the main function that starts up the CM4. CM0 is not used to run any application code.
- *touch_task.c/h* files contain the task that initializes CapSense touch sensing and read touch and gesture data from buttons and sliders.
- *display_task.c/h* files contain the task that changes the menu displayed on the E-INK display per the gesture input.
- *menu_configuration.h* contains macros and datatypes that define the menu structure.
- *screen_contents.c/.h* files constitute storage files for the images and text displayed on the screen. See Appendix A for a description of the image format.

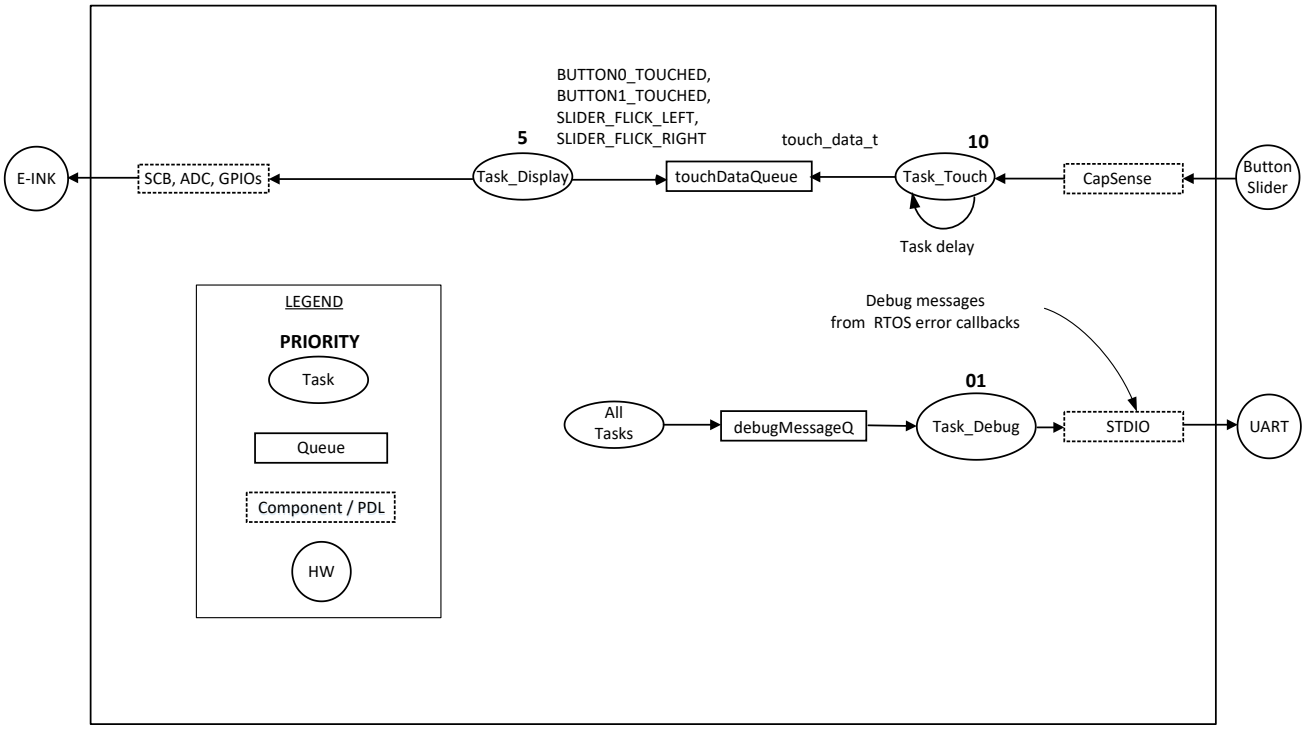
E-INK Peripheral and Driver Files:

- *cy_eink_library.c/.h* files contain the E-INK peripheral functions and macros.
- *cy_eink_fonts.c/h* files contain the font information used for text to pixel data conversion.
- *pervasive_eink_configuration.h* file contains display-vendor-provided definitions of register indexes and hardware parameters of the E-INK display.
- *pervasive_eink_hardware_driver.c/.h* files contain display-vendor-provided low-level display hardware driver functions.
- *cy_eink_psoc_interface.c/.h* files contain the PSoC 6 MCU Component-level interface to the display hardware.

**Note:** Do not edit these files as it may cause an undesirable operation of the E-INK display.

Figure 8 shows the RTOS firmware flow of this project.

Figure 8. RTOS Firmware Flow



## Components

Table 2. List of PSoC Creator Components

| Component | Instance Name | Function |
|---|---|---|
| CapSense | CapSense | The CapSense Component is configured to scan a 5-segment slider (CSD) and 2 buttons (CSX) with SmartSense auto-tuning |
| SPI (SCB) | CY_EINK_SPIM | The SPI Component is configured as a SPI master that communicates with the E-INK display driver. |
| Timer Counter (TCPWM) | CY_EINK_Timer | The Timer Counter is configured to have 1LSB = 1 ms. The count value is used for E-INK display timing. |
| Digital Output Pin | CY_EINK_Ssel<br>CY_EINK_DispRst<br>CY_EINK_DispEn<br>CY_EINK_Discharge<br>CY_EINK _Border<br>CY_EINK _DispIoEn | These GPIOs are configured as firmware-controlled output pins that are used to provide control signals to the E-INK display. |
| | LED_Red<br>LED_Orange | These GPIOs are configured as firmware-controlled output pin that control red (**LED9**) and orange (**LED8**) status LEDs. |
| Digital Input Pin | CY_EINK _DispBusy | This GPIO is a digital input without any hardware connection. It is used to read the status of E-INK display. |

See the PSoC Creator project for more details on PSoC Component configurations and design-wide resource settings.

## Related Documents

| | |
|---|---|
| **Application Notes** | |
| AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity | Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project |
| AN85951 – PSoC 4 and PSoC 6 MCU CapSense Design Guide | Describes how to design Capacitive touch sensing applications with PSoC 6 MCU |
| AN215656 – PSoC 6 MCU: Dual-Core CPU system Design | Describes the dual-core CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-core design |
| AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project | Describes how to import the code generated by PSoC Creator into your preferred IDE |
| **PSoC Creator Component Datasheets** | |
| Pins | Supports connection of hardware resources to physical pins |
| Timer Counter (TCPWM) | Supports fixed-function Timer/Counter implementation |
| Clock | Supports local clock generation |
| Interrupt | Supports generating interrupts from hardware signals |
| CapSense | Supports touch sensing |
| **Device Documentation** | |
| PSoC 6 MCU: PSoC 63 with BLE Datasheet | PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual |
| **Development Kit Documentation** | |
| CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit | |
| **Training Videos** | |
| PSoC 6 101: Lesson 1-4 FreeRTOS | |

# A E-INK Display Peripheral Driver Functions

This section describes the functions provided by the E-INK display peripheral driver. These functions are in the *cy_eink_Library.c* file.

```
void Cy_EINK_Start (      int8_t                    temperature
                          cy_eink_delay_function_t  delayFunction
                    )
```

Initializes the E-INK display hardware and PSoC Components

**Parameters**

| | |
|---|---|
| temperature | Ambient temperature in degree Celsius |
| delayFunction | Pointer to a function that accepts delay in milliseconds (uint32_t) and returns void |

**Returns**

None

**Note**

After initialization of the E-INK hardware, this function turns OFF the power to the display.

```
bool Cy_EINK_Power (      bool       powerCtrl
                    )
```

Turns ON/OFF the power to the E-INK display and initializes the E-INK driver

**Parameters**

| | |
|---|---|
| powerCtrl | false – power OFF, true – power ON |

**Returns**

false – driver initialization failed, true – driver initialization was successful

**Note**

Display contents will be retained even after the display power has been turned OFF.

```
void  Cy_EINK_Clear (     bool       background,
                          bool       powerCycle
                     )
```

Clears the E-INK display to either white or black background

**Parameters**

| | |
|---|---|
| background | false – black, true – white |
| powerCycle | false – does not control the power ON/OFF automatically<br>true – automatically controls the power cycle. This function will turn ON the power, clear the display, and then turn the power OFF. |

**Returns**

None

**Note**

If the `powerCycle` value is false, then E-INK display should be powered on (using the `Cy_EINK_Power` function) before calling this function. Otherwise, the display will not be cleared.

```
void    Cy_EINK_ShowFrame (     cy_eink_frame_t*        prevFrame
                                cy_eink_frame_t*        newFrame
                                cy_eink_update_t        updateType
                                bool                    powerCycle
                           )
```

Updates the display with a frame (image or pixel data stored in flash/RAM). See Appendix B for more information on the image and text formats stored as a frame.

**Parameters**

| | |
|---|---|
| prevFrame | Pointer to the frame that is currently displayed on the E-INK display. A frame consists of 5808 bytes (264x176/8) of data in which each bit stores the pixel information of a monochromatic image. |
| newFrame | Pointer to the frame to be displayed |
| updateType | CY_EINK_PARTIAL – updates the display from the previous frame to the new frame without any intermediate stages. This is the fastest type of update (~0.4 seconds), however, may produce ghosting if the new frame differs considerably from the previous frame.<br>CY_EINK_FULL_2STAGE – updates the display from the previous frame to the new frame with an intermediate stage that updates the display with the inverted version of the previous frame. This additional stage reduces ghosting, but increases the update time (~0.8 seconds).<br>CY_EINK_FULL_4STAGE – updates the display from the previous frame to the new frame with three intermediate stages that consists inverted version of the previous frame, white frame, and inverted version of the new frame. This type of refresh produces minimal ghosting at the cost of having the longest update time (~1.6 seconds). |
| powerCycle | false – does not control the power ON/OFF automatically.<br>true – automatically controls the power cycle. This function will turn ON the power, clear the display, and then turn the power OFF. |

**Returns**

None

**Note**

If the powerCycle value is false, then EINK display should be powered ON (using the EINKCy_EINK_Power_Power function) before calling this function. Otherwise, the display will not be updated.

```
void Cy_EINK_TextToFrameBuffer (    cy_eink_frame_t*        frameBuffer
                                    char*                   string
                                    cy_eink_font_t*         fontInfo
                                    uint8_t*                fontCor
                                )
```

Converts a string of text into pixel data and writes to a frame buffer, which can be then displayed using the `Cy_EINK_ShowFrame` function.

**Parameters**

| | |
|---|---|
| `frameBuffer` | Pointer to a frame buffer stored in RAM. A frame consists of 5808 bytes (264x176/8) of pixel data in which each bit stores the pixel information of a monochromatic image. |
| `string` | Pointer to a string |
| `fontInfo` | Pointer to a font information structure. The E-INK display peripheral driver supports two constant-sized fonts: `CY_EINK_FONT_8X12BLACK` and `CY_EINK_FONT_16X16BLACK`. See Appendix B for details of these fonts |
| `fontCor` | Pointer to a two-byte array that stores coordinates starting at which the text needs to be written. Note that this array should point to text coordinates instead of pixel coordinates. See Appendix B for details. |

**Returns**

None

**Note**

This function does not update the E-INK display. After frame buffer update, use the `Cy_EINK_ShowFrame` function to update the display if required.

```
void Cy_EINK_ImageToFrameBuffer(    cy_eink_frame_t*    frameBuffer
                                    cy_eink_frame_t*    image
                                    uint8_t*            imgCoordinates
                                )
```

Crops an image at the specified coordinates and copies it to the same location in the frame buffer. See Appendix B for more information on the image format.

**Parameters**

| | |
|---|---|
| `frameBuffer` | Pointer to a frame buffer stored in RAM. A frame consists of 5808 bytes (264x176/8) of pixel data in which each bit stores the pixel information of a monochromatic image. |
| `image` | Pointer to a monochromatic image stored in flash/RAM as an array of 5808 bytes. Image should have the same size and format as the frame buffer. |
| `imgCoordinates` | Pointer to a four-byte array that stores byte coordinates at which the image is cropped (including the final X and Y coordinates) before copying to the frame buffer. See Appendix B for details. |

**Returns**

None

**Note**

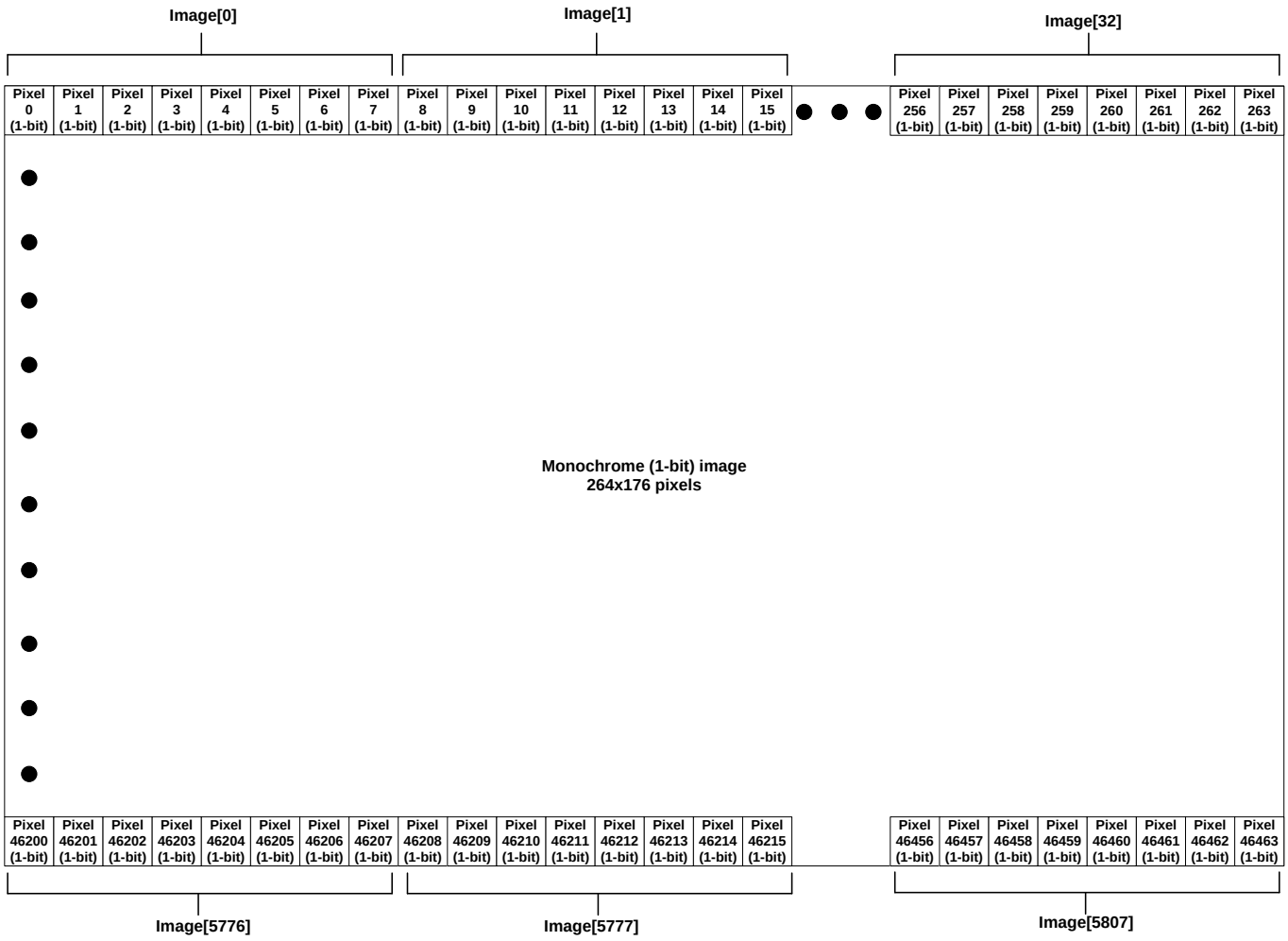
This function does not update the E-INK display. After the frame buffer update, use the `Cy_EINK_ShowFrame` function to update the display, if required.

# B Image and Text Formats

## Image and Frame Buffer Format

The E-INK display has a resolution of 264×176 pixels. The E-INK display peripheral driver supports images and frame buffers stored as a uint8 array of size 5808 (264×176/8). Figure 9 shows how the pixel data is stored in an array image[5808].

Figure 9. Image and Frame Buffer Format




You can use the PDi Apps from the display manufacturer to create a variable array from a bitmap image.

## B.1 Supported Fonts

The E-INK display peripheral driver supports two constant-sized fonts: CY_EINK_FONT_8X12BLACK and CY_EINK_FONT_16X16BLACK. Figure 10 and Figure 11and show the format of these fonts.

Figure 10. CY_EINK_FONT_8X12BLACK Format

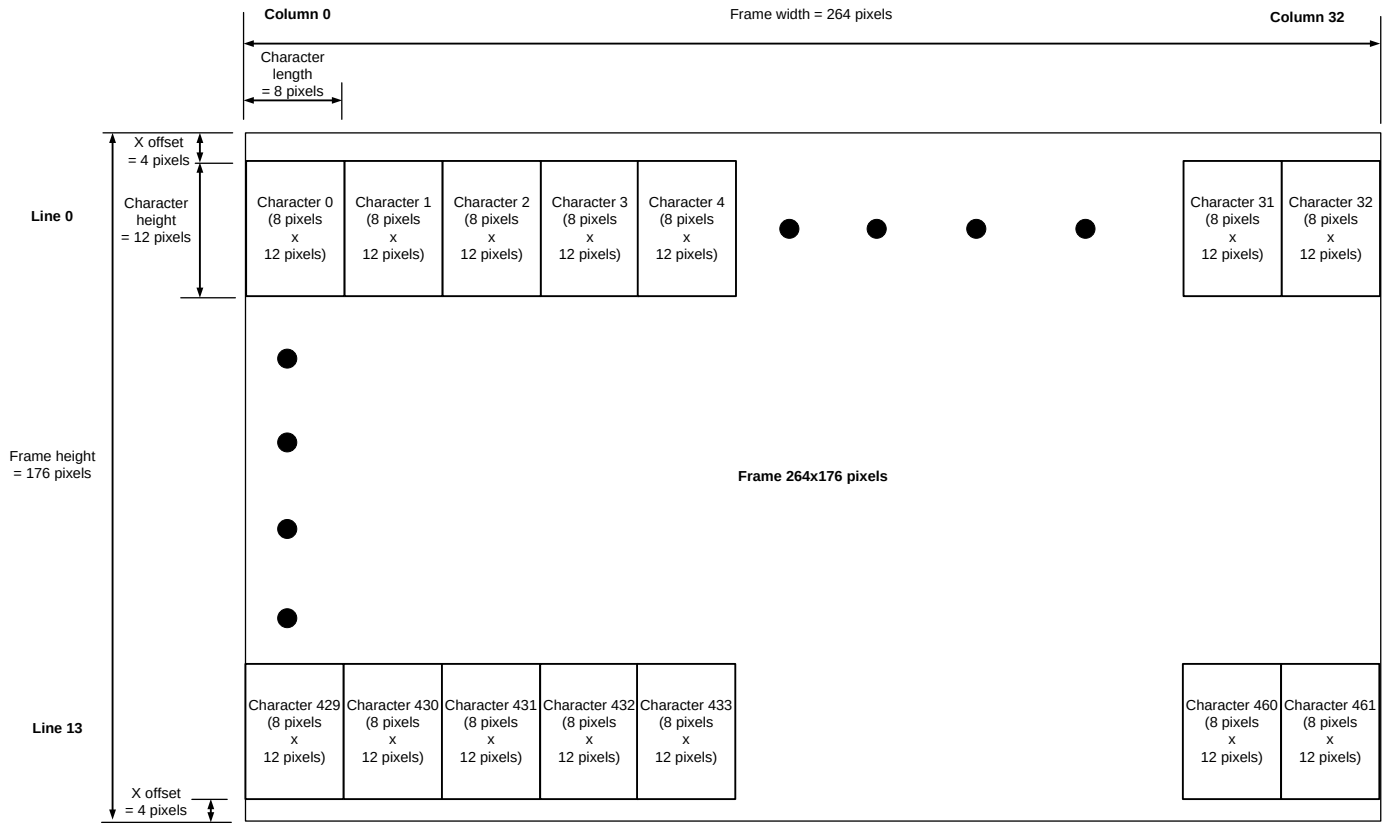
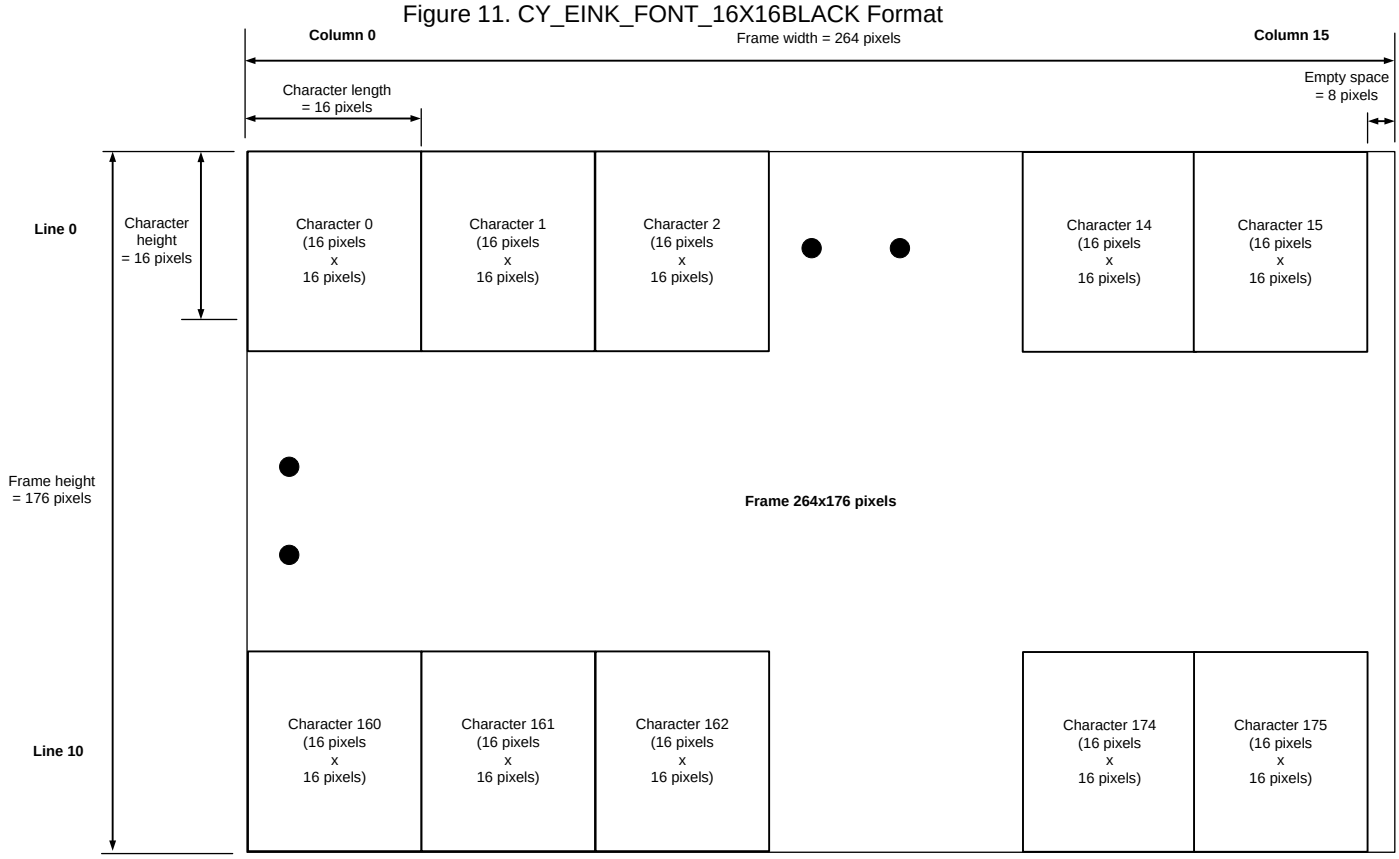


Figure 11. CY_EINK_FONT_16X16BLACK Format



# Document History

Document Title: CE218136 – PSoC 6 MCU E-INK Display with CapSense (RTOS)

Document Number: 002-18136

| Revision | ECN | Orig. of Change | Submission Date | Description of Change |
|---|---|---|---|---|
| ** | 6096839 | NIDH | 04/30/2018 | New code example |

## Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

## Products

| | |
|---|---|
| Arm® Cortex® Microcontrollers | cypress.com/arm |
| Automotive | cypress.com/automotive |
| Clocks & Buffers | cypress.com/clocks |
| Interface | cypress.com/interface |
| Internet of Things | cypress.com/iot |
| Memory | cypress.com/memory |
| Microcontrollers | cypress.com/mcu |
| PSoC | cypress.com/psoc |
| Power Management ICs | cypress.com/pmic |
| Touch Sensing | cypress.com/touch |
| USB Controllers | cypress.com/usb |
| Wireless Connectivity | cypress.com/wireless |

## PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

## Cypress Developer Community

Community Forums | Projects | Videos | Blogs | Training | Components

## Technical Support

cypress.com/support



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709