TI Designs

Access Control Panel With Bluetooth® low energy, Capacitive Touch, and Software Integration Reference Design

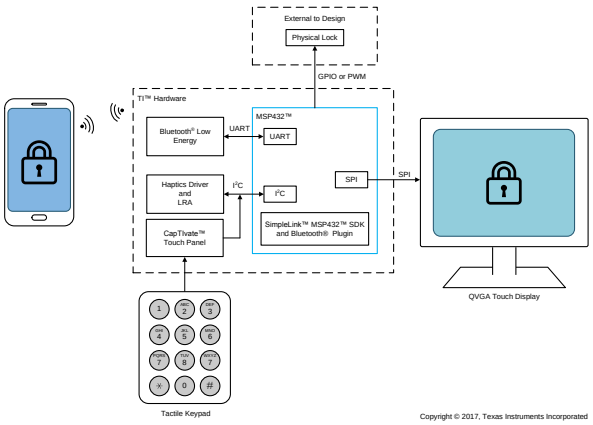
Description

This TI Design provides a modular software package based on the SimpleLink™ Software Development Kit (SDK), which provides portability and extensibility for customized functionality. The design shows one of many possible configurations based on readily available hardware as a power-optimized, highly integrated access panel. The highlighted configuration features SimpleLink microcontrollers (MCU) such as the MSP432P401R as a multifunction host MCU that controls the SimpleLink CC2650 in a network processor configuration, drives a color QVGA LCD, interacts with the MSP430FR2633 integrating CapTIvate™ Touch Technology, and controls the tactile feedback through the DRV2605L Haptics driver.

Resources

TIDM-1004	Design Folder
MSP432P401R	Product Folder
MSP430FR2633	Product Folder
CC2650	Product Folder
DRV2605L	Product Folder
BOOSTXL-K350QVG-S1	Tools Folder
SIMPLELINK-MSP432-SDK	Tools Folder

TI E2E™ Community — ASK Our E2E Experts



Features

- Modular, Configurable, and Reusable Software Package Provides Portability and Extensibility for Customized Functionality Based on SimpleLink SDK
- Optimized for Ultra-Low-Power Operation, <20-µA Overall System (Average) Current Consumption While in Sleep Mode
- Compatible With Existing SimpleLink SDK Plugins for Supporting Various Wireless Protocols
- Features SimpleLink MCUs: SimpleLink MSP432™, Pre-Certified BLE SimpleLink CC2650MODA
- Advanced HMI With 12-Button Keypad and One Proximity Sensor Controlled by Ultra-Low-Power MSP430™ MCU With CapTIvate Touch Technology and FRAM, and Tactile Feedback Using DRV2605L Haptics Driver
- SimpleLink MSP432 MCU Supports Extensible Functionality for Sensing and Control
- Leverages LaunchPad™ for Easy Out-of-Box Experience

Applications

- Industrial Panels
- Security Systems
- Indoor and Outdoor Locks
- Thermostats and HVAC Control

1 System Overview

1.1 System Description

This TI Design for a highly integrated access panel consists of several MCUs supervising certain parts of the system and reconfigurable software designed for TI hardware. The SimpleLink MSP432P401R MCU is the center of this TI Design, integrating the power of the industry-standard ARM Cortex-M4F processor, with the efficiency and ultra-low-power features of the MSP product line from TI. In addition, the LaunchPad™ ecosystem allows interfacing with many different products through the BoosterPack™ module footprint. Here, the design uses existing BoosterPack modules to quickly construct a viable human-machine interface (HMI) demonstration leveraging readily available hardware.

This TI Design is optimal for portable entry systems, locks, interface panels, and thermostat controls because it leverages a smooth, tactile touch panel with out-of-box software, incorporates wireless connectivity with an integrated SDK between the host and the wireless interface, and then packages that into an efficient, modular solution that extends battery life and can be reconfigured for different types of advanced HMI and wireless interfaces.

Some application examples may require replacing the Bluetooth® low energy interface with Wi-Fi®, ZigBee®, Sub-1 GHz, or other wireless standards. Other examples may require changing the advanced HMI to use different technologies such as touch screens, mechanical buttons, LEDs, or membrane switches, and support various sensors. Based on the SimpleLink SDK, this TI Design helps to shorten and simplify complex software development by allowing users to focus on adding different functionality.

In addition, the SimpleLink MSP432 SDK brings in a unified software platform to easily add wireless connectivity through the CC2650 wireless MCU, an ultra-low-power 2.4-GHz multi-standard radio. Furthermore, CapTIvate technology replaces a standard mechanical keypad and adds proximity sensing, which is used as an external interrupt to wake up the system from deep sleep and thus drive down global current consumption.

The MSP432 MCU supervises the entire system. The MSP432 provides enough memory to support RTOS-based applications while providing space for extra functionality and video or multimedia output to the user, and employs various power-saving schemes to preserve battery life. The MSP432 MCU initializes the LCD screen right after start-up, proceeds to initialize the different operating system tasks, and then starts running the operating system. The MSP432 MCU responds to I^2C data from the MSP430FR2633 MCU running the CapTIvate capacitive touch panel, or information to and from mobile devices through the CC2650 BoosterPack. Depending on the action, the MSP432 device may store a keypress, lock or unlock the device from the touch panel data, or lock and unlock the device wirelessly, firing off a corresponding buzz pattern by sending a *fire* message to the DRV2605L haptics driver over I^2C. If there is no activity during a certain time-out period, the period elapses and the MSP432 MCU goes to sleep, shutting down the LCD screen and the CC2650 MCU in the process.

A dedicated capacitive touch HMI reads useable information from capacitive touch sensors, and usually requires an interface to a host processor. This system uses the MSP430FR2633 MCU to drive the CapTIvate IP Phone panel using GPIOs directly wired to the onboard CapTIvate peripheral. Each button press is detected by a change in capacitance by a series of traces on the touch panel. The CapTIvate peripheral then scans these traces as sensors in parallel and interprets the changes in capacitance as possible button presses. Using a dedicated capacitive touch HMI with the MSP432 host processor lets developers integrate multiple technologies into a single product design, while expanding system capabilities and reducing power consumption.

Adding a dedicated wireless MCU delivers connectivity to mobile devices and allows the application to connect to the IoT. In this system, the SimpleLink CC2650 2.4-GHz wireless MCU drives all wireless communication with external devices, such as mobile phones or tablets, over Bluetooth low energy. Here, the SimpleLink Bluetooth low energy CC2650 BoosterPack plug-in module from TI incorporates the CC2650MODA MCU, preprogrammed with network processor firmware.

This MCU is a pre-certified, 2.4-GHz, multi-standard wireless MCU with an integrated antenna. The CC2650 then acts as a gateway that advertises, connects to external devices, and exchanges specific characteristics to relay lock state information and user passcode entry back and forth between the mobile user and the system. This functionality complements the capacitive touch HMI, in the case the end user cannot use the touch panel.

Lastly, a haptics driver with a linear resonant actuator (LRA) provides tactile feedback from the touch panel, providing a confirmation to the user that a valid keypress was detected and processed accordingly. The DRV2605L haptics driver triggers a Samsung DMJBRN1030-series LRA to vibrate upon any key entry, vibrate twice for a successful unlock, or vibrate for a longer period for an unsuccessful PIN entry.

This TI Design provides a software solution to quickly prototype HMI systems and demonstrates how to interface an MSP432 MCU with an MSP430 MCU featuring CapTIvate touch technology, along with Bluetooth low energy communication, and an LCD screen. The design integrates a capacitive touch HMI with haptics, a low-power and high-performance 32-bit ARM Cortex-M4F MCU, a wireless Bluetooth low energy MCU, and an LCD screen.

1.2 Key System Specifications

Table 1 lists the key specifications for the access control panel. All memory footprints are calculated in Code Composer Studio™ 7 (CCS7), with the TI v16.9.0.LTS compiler, with optimization disabled.

Table 1. Access Control Panel System Requirements and Specifications

FEATURES	SPECIFICATIONS	ADDITIONAL DETAILS
Memory footprint (MSP432P401R MCU)	256KB of flash (main memory) 16KB of RAM (SRAM)	Includes CC2650 SNP Hex image (128kB), calculated using CCS7
Memory footprint (MSP430FR2633 MCU)	8KB of FRAM 1.4KB of RAM	Calculated from CCS
Memory footprint (CC2650MODA)	128KB	From SimpleLink MSP432 SDK Bluetooth Plugin User's Guide
Interface composition	12 buttons, 1 wheel button, 1 proximity or guard channel	—
Serial interfaces	UART, I^2C, SPI	—
Measurement time	20 to 40 ms (approximately 1 or 2 scan periods)	20-ms active scan period
Average power consumption	18 µA	Across the system, sleep mode, measured with all LED and XDS110 jumpers off.

1.3 Block Diagram

Figure 1 and Figure 2 show the system block diagram.

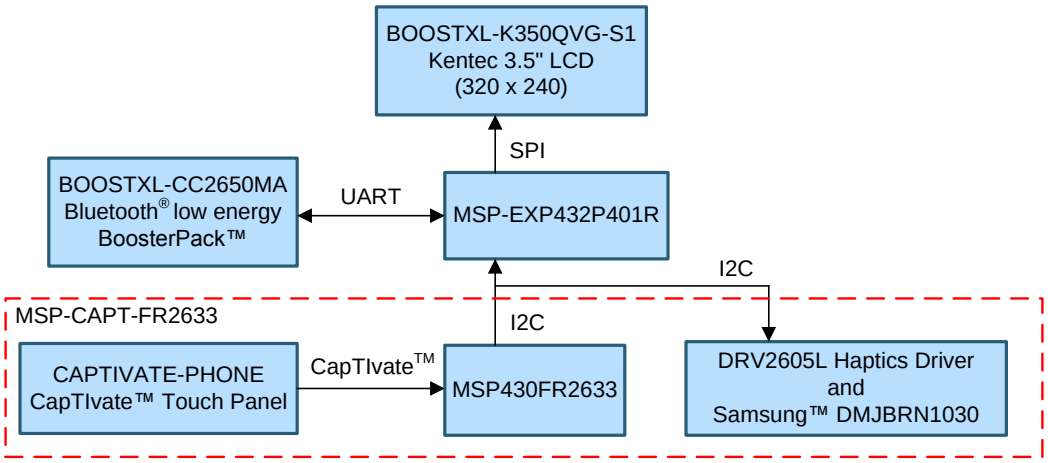


Figure 1. TIDM-1004 Block Diagram

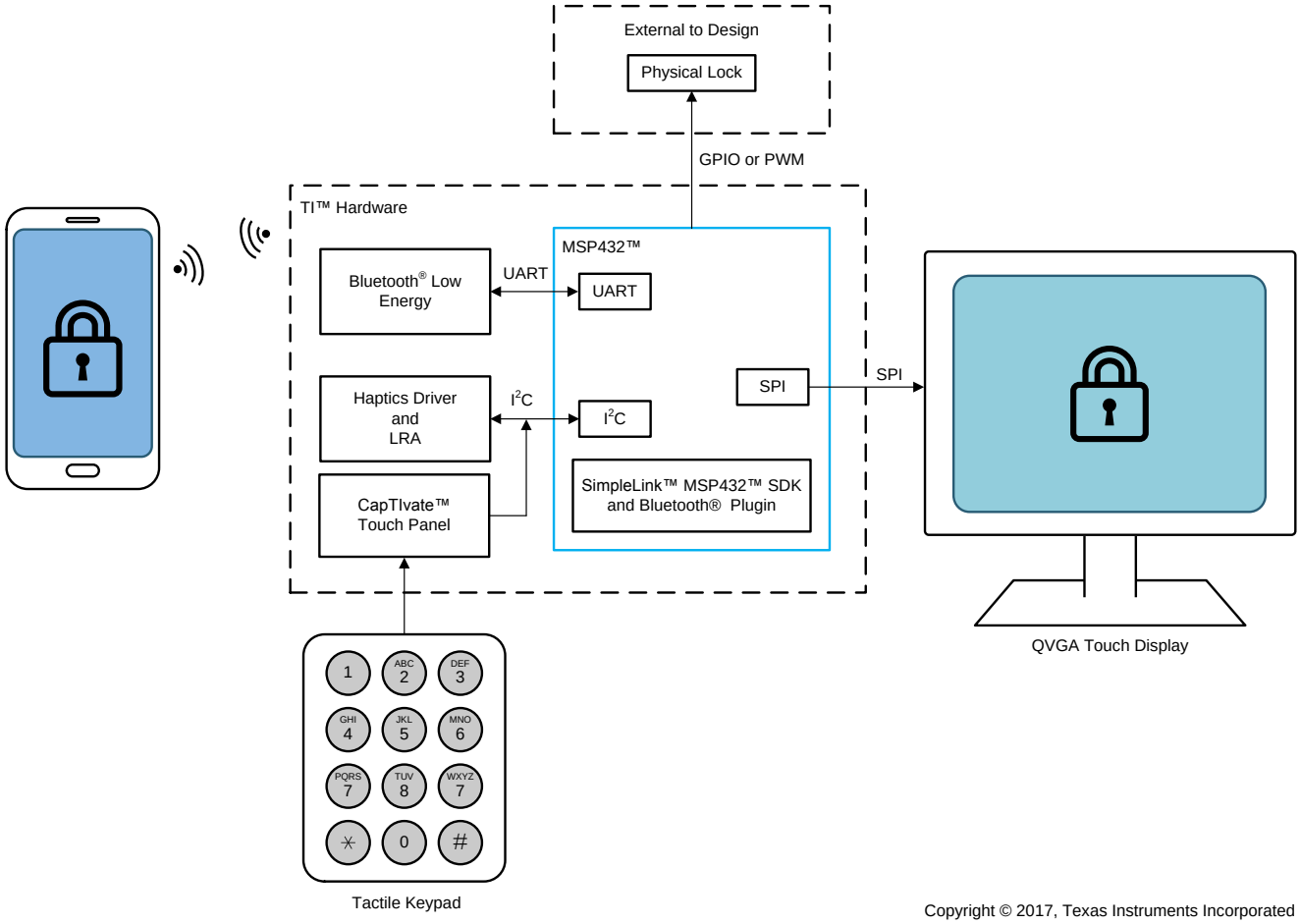


Figure 2. System Block Diagram (Enlarged)

1.4 Highlighted Products

1.4.1 SimpleLink MSP432P401R MCU

The SimpleLink MSP432P401R device is an ultra-low-power, 32-bit, ARM Cortex-M4F-based MCU. The MSP432P401R MCU features a 14-bit, analog-to-digital converter (ADC), and a wide range of analog, timing, and communication peripherals. The MSP432P401R device is optimized to be the wireless host MCU for a wide range of connected applications. The device includes 256KB of flash main memory and 64KB of SRAM. With a clock system of up to 48 MHz, the large amount of memory available makes the MSP432P401R MCU ideal for applications that require efficient data processing with enhanced low-power operation. Figure 3 shows the functional block diagram of the MSP432P401R MCU.

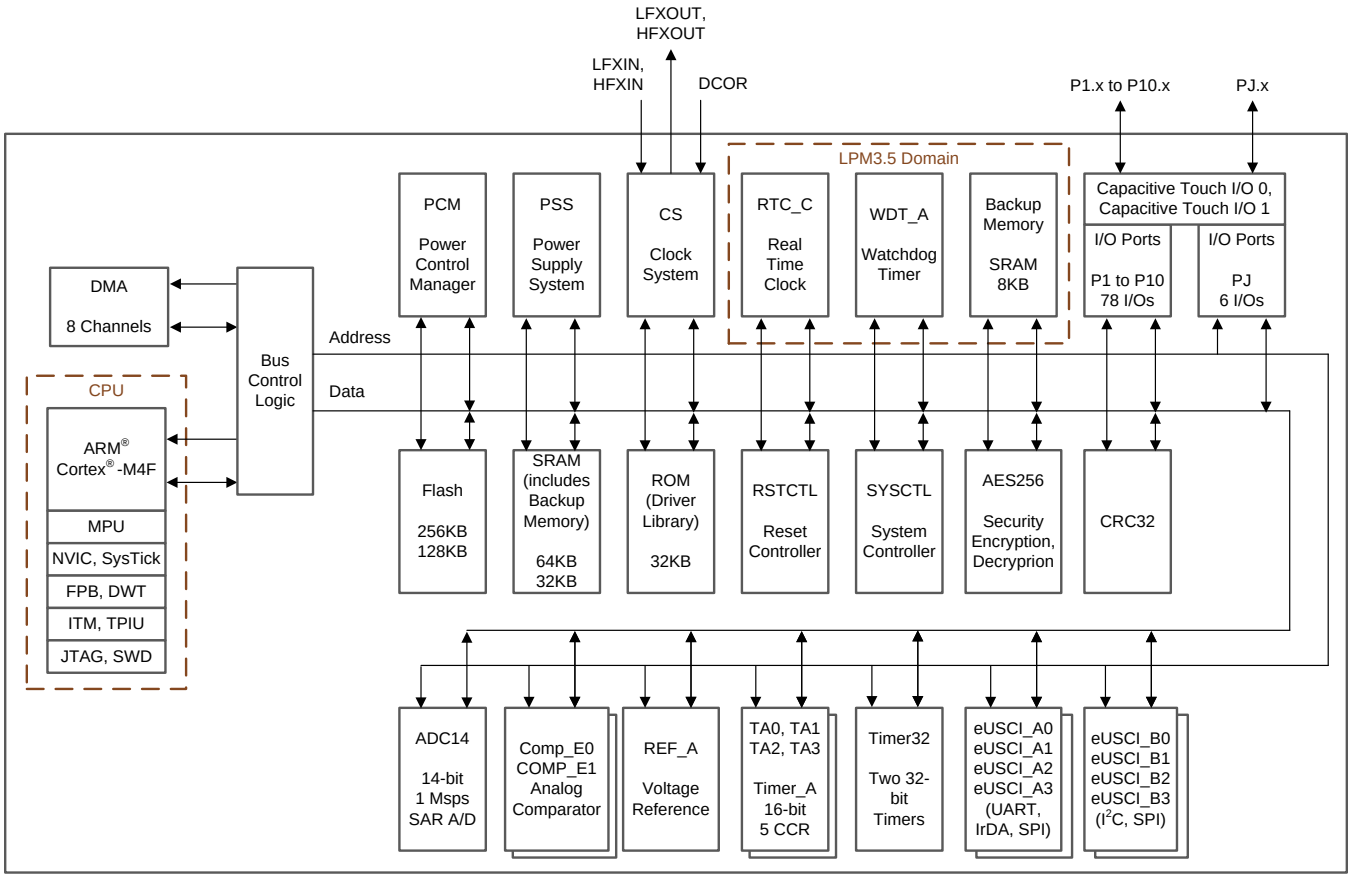


Figure 3. SimpleLink MSP432P401R MCU Block Diagram

SimpleLink MSP432P401R MCU features:

- 32-bit ARM Cortex-M4F CPU with a floating point unit and a memory protection unit
- Flexible clocking with frequency up to 48 MHz
- 256KB of flash main memory, 16KB of flash information memory, 64KB of SRAM, and 32KB of ROM programmed with SimpleLink MSP432 SDK driver libraries
- Eight-channel direct memory access (DMA)
- Four 16-bit timers, two 32-bit timers, and a real time clock (RTC) with calendar and alarm functions
- Four eUSCI-A modules for UART, IrDA, or SPI
- Four eUSCI-B modules for I^2C with multiple-slave addressing or SPI
- 48 I/Os with interrupt and wakeup capability
- 14-bit, 1-msps SAR ADC
- 128-, 192-, and 256-bit AES encryption and decryption accelerator with a 32-bit hardware CRC engine

1.4.2 MSP430FR2633 MCU

The MSP430FR2633 device is an ultra-low-power, FRAM-based MSP430 MCU equipped with CapTIvate touch technology. The MSP430FR2633 MCU includes 15.5KB of FRAM and 4KB of RAM, which makes it capable of supporting complex capacitive touch applications. Integrating CapTIvate touch technology with the strong MSP430 peripheral set and a large memory footprint makes the MSP430FR2633 device an ideal MCU for low-power, user-interface development.

Figure 4 shows a block diagram of the MSP430FR2633 MCU.

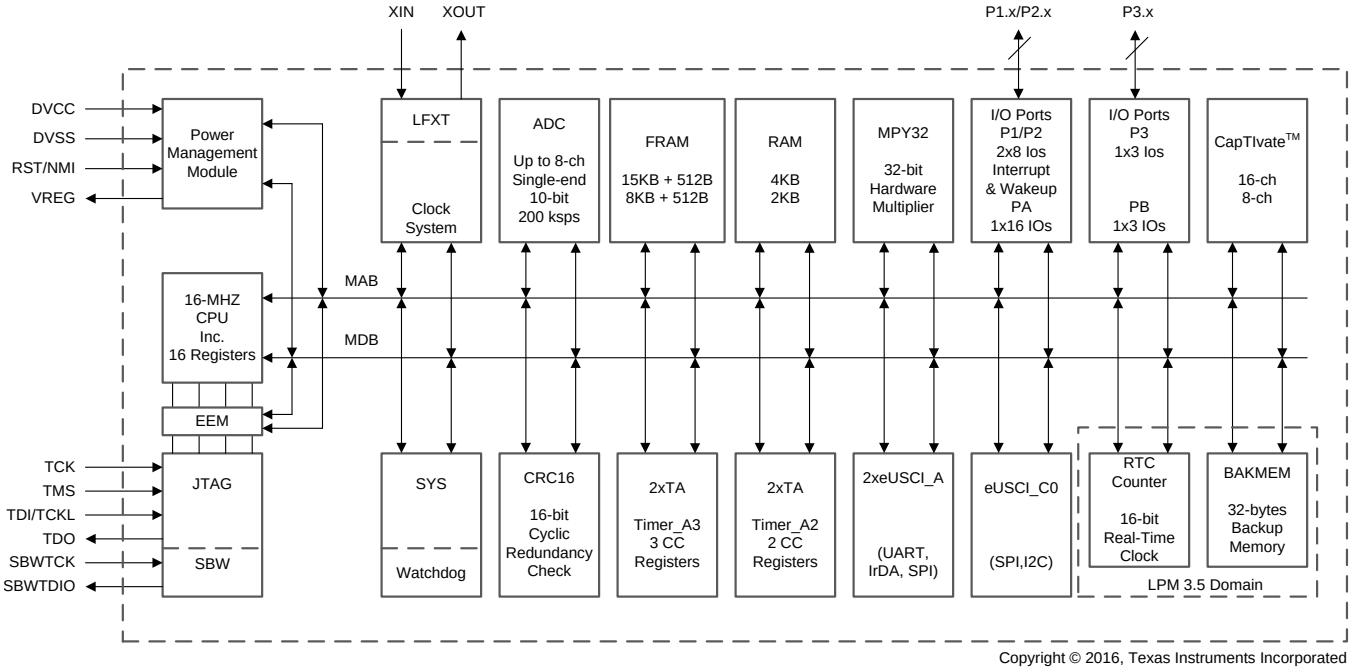


Figure 4. MSP430FR2633 MCU Block Diagram

MSP430FR2633 MCU features:

- 16 CapTIvate technology I/Os that support up to 64 electrodes in mutual-capacitance mode
- Parallel scanning of up to four electrodes at a time
- CapTIvate Software Library included in a preprogrammed 12KB of ROM
- Four 16-bit timers and a 16-bit counter-only RTC
- Three enhanced serial communications peripherals for UART, IrDA, SPI, and I^2C
- 19 I/Os with 16 interrupt pins for wake-up from low-power modes
- High-performance, 8-channel, 10-bit ADC
- Clock system with operation up to 16 MHz

1.4.2.1 CapTIvate Technology

CapTIvate technology enables capacitive sensing on this reference design. CapTIvate technology is a peripheral on the MSP430FR253x or MSP430FR263x MCU dedicated to providing robust capacitive-sensing measurements.

CapTIvate technology provides a set of hardware and software tools, which accommodate a wide range of external capacitances. Each CapTIvate MCU includes four instances of the CapTIvate technology measurement block to enable sensors to scan in parallel. For more information about MSP430 MCUs featuring CapTIvate technology, see the *CapTIvate Technology Guide*.

Figure 5 shows a block diagram of the CapTIvate technology module.

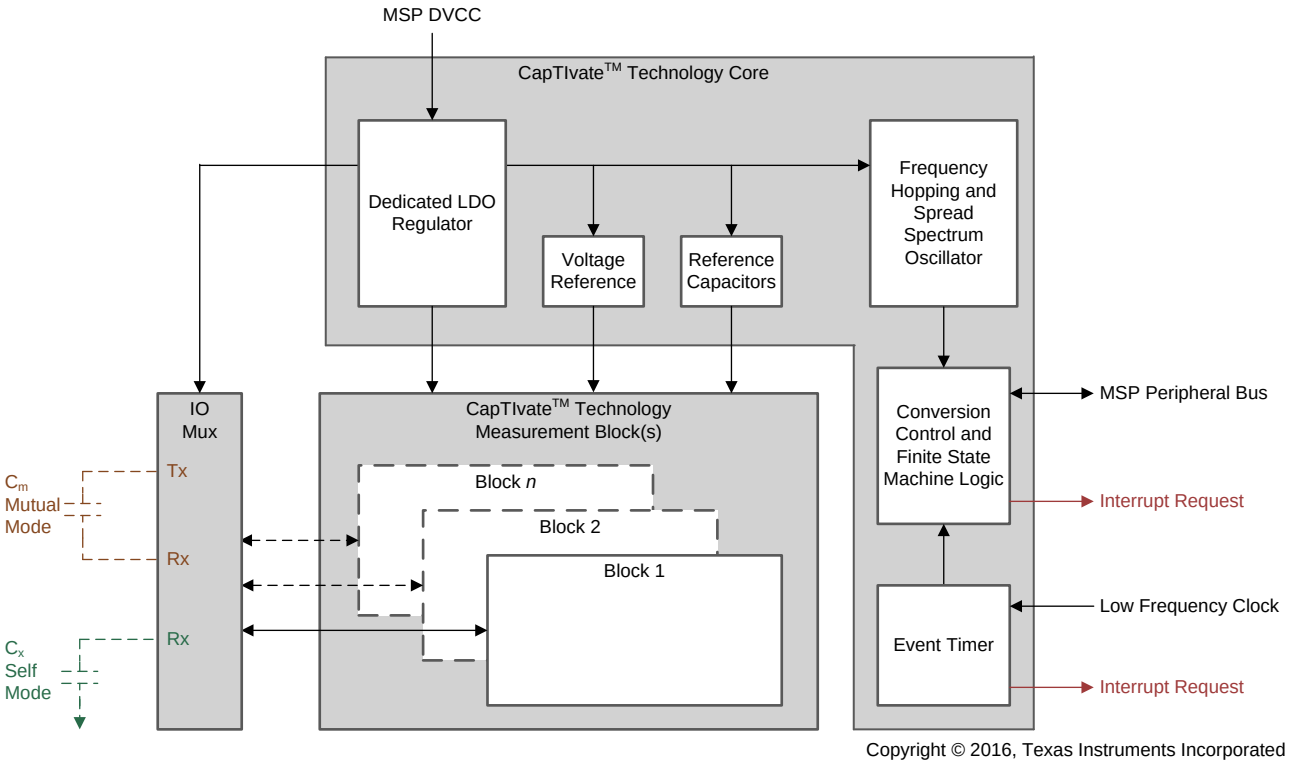


Figure 5. CapTIvate Technology Block Diagram

1.4.3 SimpleLink CC2650 2.4-GHz Multi-Standard Wireless MCU

The SimpleLink CC2650 2.4-GHz multi-standard MCU is an ARM Cortex-M3 MCU with wireless communication peripherals targeting 2.4-GHz ultra-low-power applications. The device includes 2KB of ultra-low-leakage SRAM, 128KB of system flash memory, an 8KB SRAM cache, and 20KB of ultra-low-leakage SRAM. The architecture features very-low current draw when the RF and MCU domains are active, and a low-power mode that reduces current consumption further to enable long battery life, particularly within applications powered by coin-cell batteries. The memory capacity and the low current draw are designed for battery-powered IoT applications.

The SimpleLink Bluetooth low energy CC2650 BoosterPack plug-in module offers a expedited way to provide an integrated hardware solution quickly, without having to develop a new hardware board, integrate an antenna, and obtain approval from regulatory agencies.

Figure 6 shows the functional block diagram of the CC2650 MCU. The CC2650 MCU functions as a simple network interface to the entire system. The CC2650 MCU has an ARM Cortex-M3 core, which is a sensor core for running peripherals while the CPU is in a low-power mode, and a fully integrated RF front-end radio with an additional Cortex-M0 to control the RF front end. The CC2650MODA device used in the hardware stack has a larger architecture because it supports multiple wireless standards and integrates an antenna.

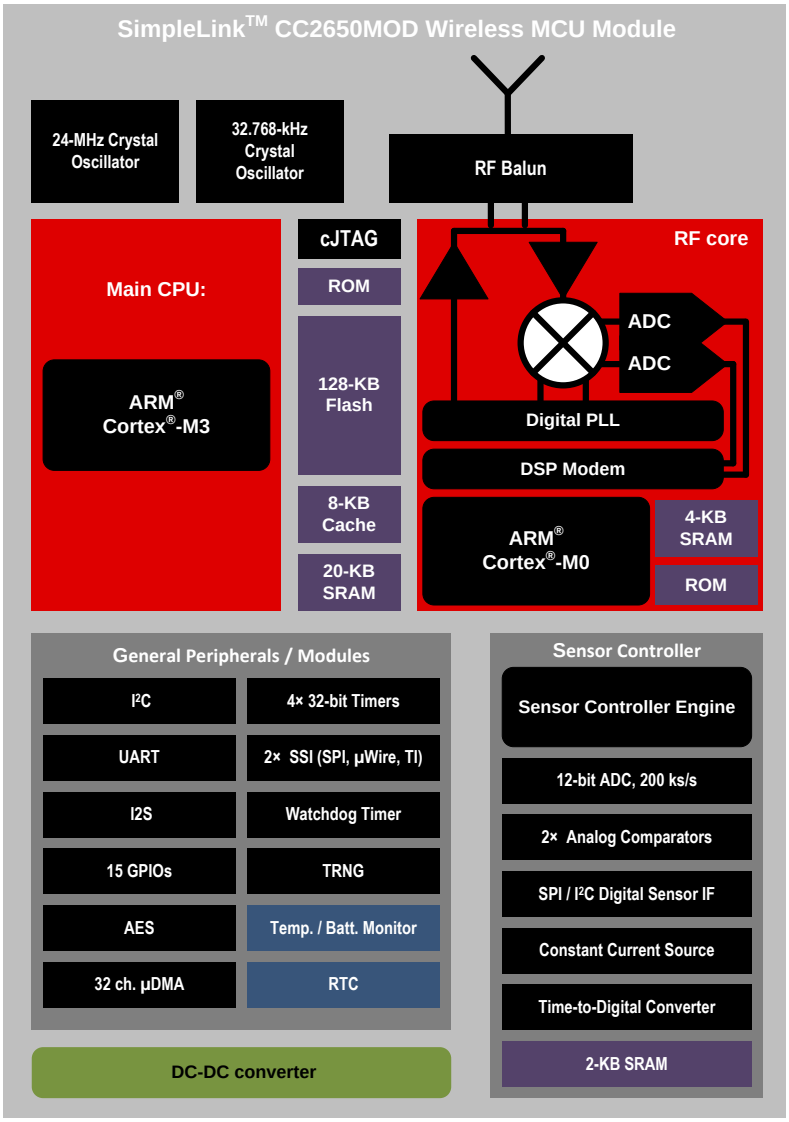


Figure 6. SimpleLink CC2650 MCU Block Diagram

SimpleLink CC2650 MCU features:

- 32-bit ARM Cortex-M3 CPU
- Flexible clocking with frequency up to 48 MHz
- 2KB of ultra-low-leakage program and data SRAM, 128KB of system flash memory, an 8KB SRAM cache, and 20KB of ultra-low-leakage SRAM
- Four 16- and 32-bit configurable general-purpose timer models with PWM outputs
- 12-bit, 8-channel, 200 ksps ADC
- UART, I^2C, I2S
- Two SSI modules (configurable to SPI, Microwire, and TI)
- 31 GPIOs with full pin-multiplexing
- 128-bit AES encryption accelerator with true RNG

1.4.4 DRV2605L Driver

The DRV2605L is a low-voltage haptic driver for eccentric rotating masses (ERMs) and LRAs. The device includes integrated ROM effect libraries and provides a closed-loop actuator control system, with access through a shared I^2C compatible bus or PWM input signal. The DRV2605L device is used with an LRA on the touch panel in this TI Design to provide users with high-quality mechanical feedback, which simulates the click of a button upon interaction.

Figure 7 shows a block diagram of the DRV2605L driver.

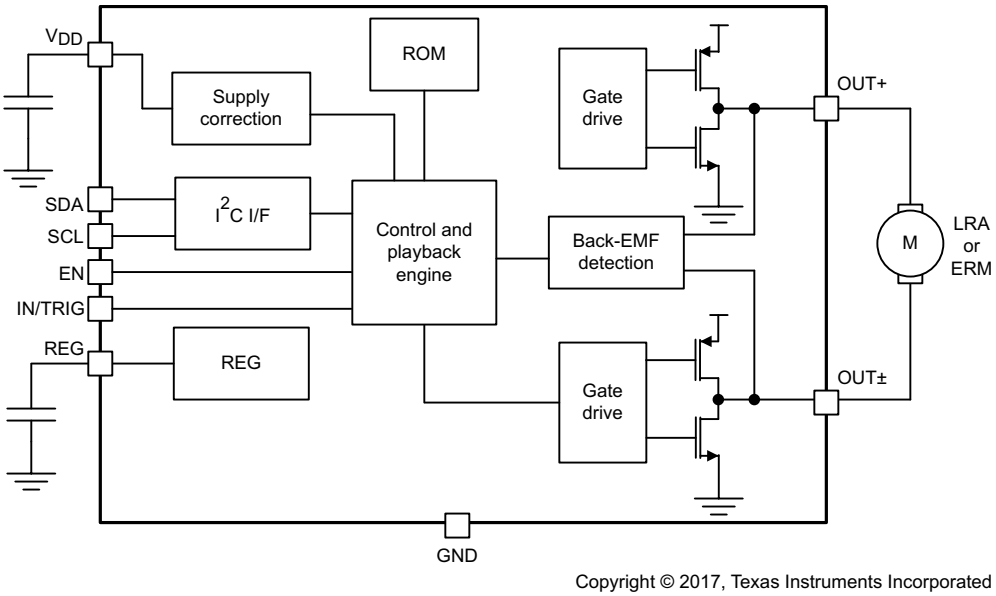


Figure 7. DRV2605L Simplified Schematic

1.4.4.1 Haptics With The DRV2605L Driver

Haptics are implemented with the DRV2605L haptics driver, used to drive linear resonant actuators. This driver IC accepts commands over the I^2C bus. The DRV2605L haptics driver contains a licensed version of the TouchSense® 2200 software.

2 System Design Theory

This TI Design implements an access panel by using the SimpleLink MSP432 MCU as a host to interface to the CC2650 MCU with communications software for the CC2650 MCU. The design also takes advantage of the communications module included in the CapTIvate Software Library to create a simple interface between the MSP430 with CapTIvate technology, and then relays that information to the MSP432 MCU. Finally, the MSP432 MCU drives the information about the state of the system out to the display using driver software (MSP Graphics Library) coupled with the Kentec QVGA BoosterPack.

2.1 CapTIvate Software Library Communications Module

The CapTIvate Software Library communications module is a layered set of firmware that provides a simple top-level application program interface (API) for connecting a CapTIvate MCU to a PC or to a host processor through a standard, common serial interface. The communications module is comprised of the following four layers, listed in order of decreasing abstraction:

- Interface layer: implements top-level API
- Protocol layer: implements CapTIvate protocol packet generation and interpretation
- Serial driver layer: several interchangeable drivers for serial interfaces
- Data structure layer: basic abstract data types, such as a FIFO queue and ping-pong buffer

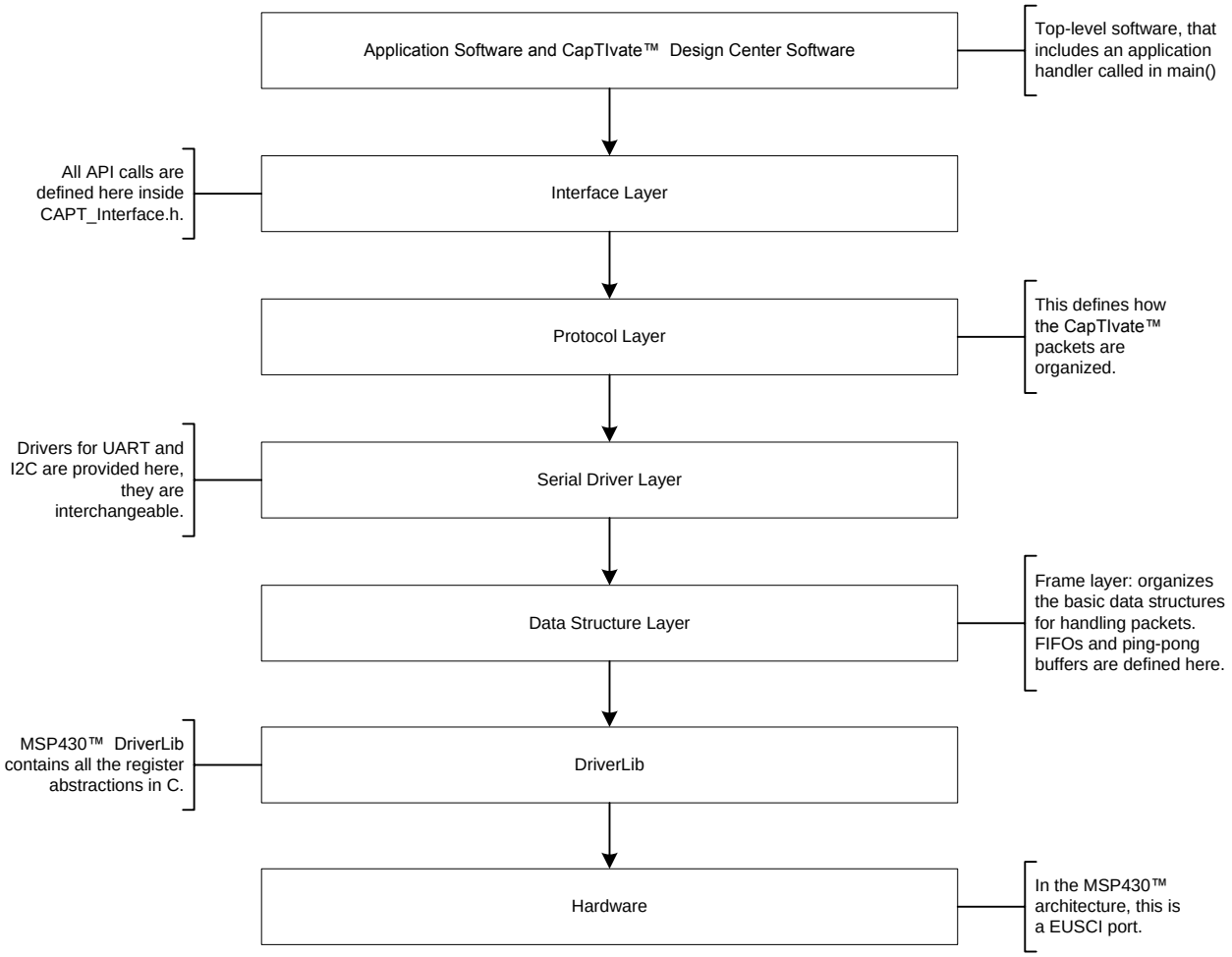


Figure 8. CapTIvate Software Library Abstraction Model

UART and I^2C drivers provided with the communications module speed up the development of applications using MSP430 MCU with CapTIvate technology with a host processor. For more information on the communications module, see the *CapTIvate Technology Guide*.

2.1.1 CapTIvate Communication Abstraction Layer

The communication abstraction layer includes high-level APIs for reading sensor, cycle, and element data from a CapTIvate-technology HMI. Received data is either in a CapTIvate sensor packet or a CapTIvate cycle packet.

Sensor packets are fixed-length and include sensor-status information, as well as dominant-element and previous dominant-element information, slider position, or wheel position. Cycle packets vary in length based on the number of elements on the cycle. Cycle packets include the proximity state, touch state, count value, and lost time accident long-term average (LTA) of each element in the cycle.

The communication abstraction layer services a request to read a sensor or cycle packet by returning a pointer to the most recent version of a received packet to the application. This packet can be used to evaluate the HMI state and respond to user interaction.

2.1.2 Protocol Layer

The protocol layer performs the following functions to ensure complete data is received and stored correctly during a transmission.

The protocol layer facilitates the interpretation of received data by providing definitions for structures that describe element data, cycle data, and sensor data. The protocol layer also includes many constant definitions that describe the CapTIvate data packet structure and protocol-control bytes such as the default I^2C slave address and the UART HID header bytes. This TI Design specifically builds upon the I^2C structures. For more information on the CapTIvate protocol, see the Communications Module under the Software Library Section of the *CapTIvate Technology Guide*.

2.1.3 Data Structures

The example host communications module implementation relies on the creation of a few key data structures. Depending on the use of UART or I^2C communication, the data structures which the host uses to create the interface are different. The following sections provide a brief summary of the functionality and implementation of these data structures.

2.1.3.1 UART Data Structures

While the CapTIvate protocol provides data structures for UART and I^2C communication, a unique I^2C data structure and interface is used in this design. In addition, the UART bus is reserved for the SimpleLink CC2650 wireless MCU, thus rendering it unusable for CapTIvate data transfers to and from the CapTIvate phone touch panel in this reference design. For details on the traditional UART data structure, see Section 4.2.4.1 in the MSP432 CapTIvate™ TI Design User's Guide.

2.1.3.2 I^2C and I^2C Data Structures

The I^2C interface implements the register mode with a 400-kbps data rate for the fastest possible data transfer between the touch panel and the SimpleLink MSP432 MCU. In register mode, the host must request specific data packets from the CapTIvate HMI each time the host requires new data. For information on the traditional I^2C interfaces, see Section 4.2.4.2 in the MSP432 CapTIvate™ TI Design User's Guide.

However, to further encapsulate the touch panel processing and management to the relevant MCUs, a new data transfer scheme was implemented on top of the existing CapTIvate ping-pong buffers. Here, the touch panel controller manages all Boolean flags and CapTIvate data internally, as well as artificially packs a new buffer, txBuffer[], with integers that indicate to the MSP432 MCU which key was pressed on the touch panel. This addition further simplifies software development on the MSP432 MCU because the MSP432 MCU does not have to micromanage all the CapTIvate flags.

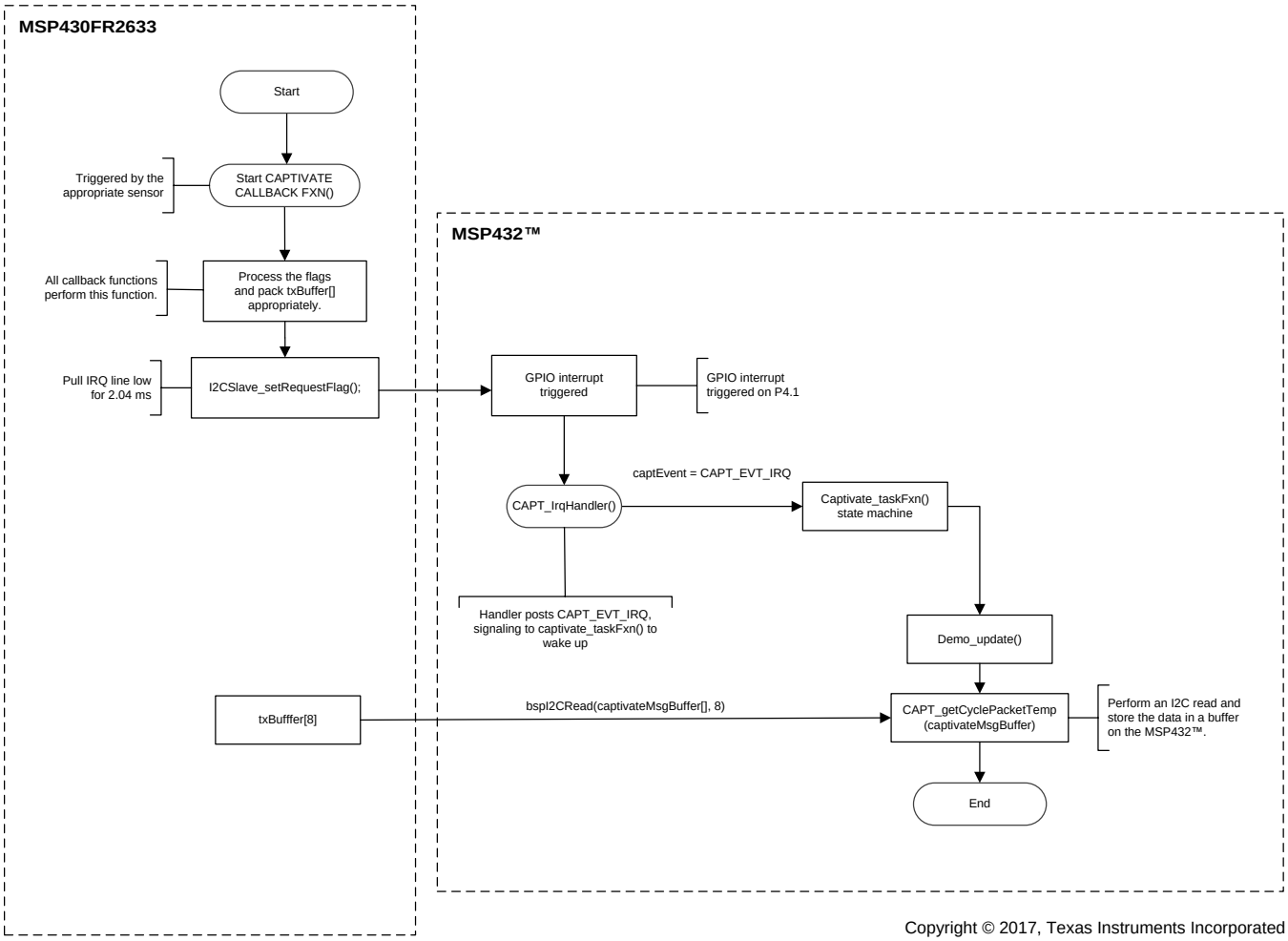


Figure 9. SimpleLink MSP430FR2633 to MSP432 Bridge Interface Flow Chart

First, whenever there is activity on the touch panel, the MSP430 device processes data from the touch panel, packs the buffer appropriately, and then sends a *send request* signal to the MSP432 MCU, by pulling the GPIO interrupt request (IRQ) line low using I2CSlave_sendRequestFlag(). Next, the MSP432 MCU calls three functions in the active state within captivate_tasFxn() to gather touch panel data from the MSP430 device using CAPT_getCyclePacketTemp().

For more information on the txBuffer[] data structure and interface, see Section 3.2.4. For information about captivate_taskFxn(), see Section 3.2.3.3.

2.2 SimpleLink MSP432 SDK

The simplest way to interface a host processor with external processors and devices is to develop firmware that complements related software development packages, such as the SimpleLink MSP432 SDK, the SimpleLink MSP432 SDK Bluetooth low energy plugin.

2.2.1 Overview of SimpleLink MSP432 SDK

The SimpleLink MSP432 SDK is a layered set of firmware that provides multiple APIs, driver libraries, cross-platform plugins, POSIX, and TI-RTOS support for the SimpleLink MSP432 MCU. The SimpleLink MCU portfolio offers a single development environment that delivers flexible hardware, software and tool options for customers developing wired and wireless applications. With 100 percent code reuse across host MCUs, Wi-Fi, Bluetooth low energy, Sub 1-GHz devices and more, choose the MCU or connectivity standard that fits your design. A one-time investment with the SimpleLink software development kit allows you to reuse often, opening the door to create unlimited applications. For more information, visit www.ti.com/simplelink.

Figure 10 shows an overview of the organization of the ecosystem.

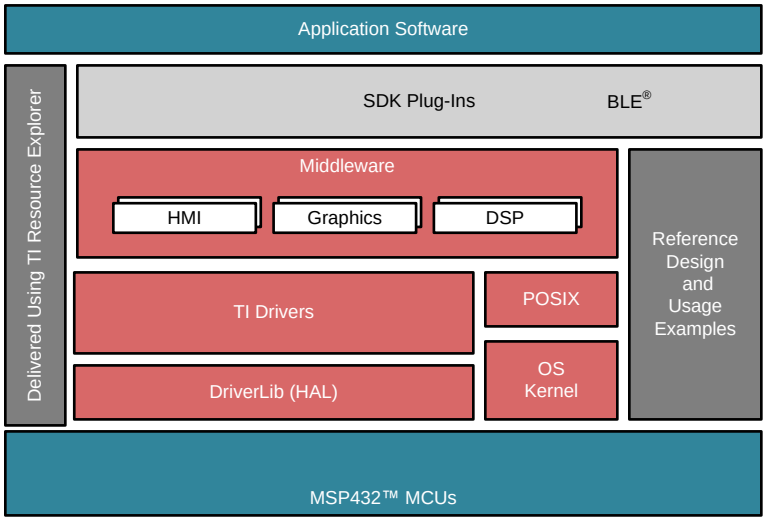


Figure 10. MSP432 MCU Host Application and Software Ecosystem Organization

The interaction between each layer in the firmware stays constant between MCU platforms and wireless product platforms. There is a built-in interchangeability between the choice of RTOS, both TI-RTOS and FreeRTOS. See the SimpleLink SDK release notes in the /docs folder within the installation directories for the SDK and any SDK plugin.

2.2.2 Serial Communications Drivers

The SimpleLink MSP432 SDK provides UART, SPI, and I^2C master drivers, as shown in Figure 10.

2.2.2.1 UART

The UART interface is configured for a 115200 baud rate. This configuration is done to ensure that the simple application processor (SAP) and simple network processor (SNP) can interface with each other, because the SNP hex image on the CC2650 MCU configures the UART port to communicate at a 115200 baud rate. Because the SNP only supports UART and SPI, UART was chosen to interface with the MSP432 MCU.

2.2.2.2 SPI

SPI is used to communicate with the QVGA 320 × 240 display. The SPI bus is clocked at 12 MHz to ensure a qualitatively fast response time. The MSP432 MCU uses the supplied TI SPI driver files, referenced within the LCD driver software, to initialize SPI and the display, and then uses the MSP Graphics Library to draw images and write text on the display.

2.2.2.3 I²C Master

This reference design services incoming data signaled in using a GPIO interrupt from pin 4.1 on the MSP432 MCU connected to pin 2.2 on the MSP430FR2633 MCU, which raises an event flag to direct the host toward servicing the data. See Section 2.1.3.2 for details on the I²C data transfer and servicing. See Figure 11 in Section 2.4 for a diagram of the I²C bus members.

2.3 CapTIvate Phone Touch Panel

The CAPTIVATE-PHONE touch panel uses a combination of mutual-capacitance and self-capacitance technology in a desk-phone application form factor. The touch panel demonstrates how to create a mutual-capacitance matrix to form 17 buttons, 2 sliders, 1 wheel, and 1 proximity sensor, using only 12 CapTIvate I/Os. See Sensor Panel Demonstrators and Capacitive Sensing Basics in the *CapTIvate™ Technology Guide* for more information on the CapTIvate Phone Touch Panel. This panel is used because it provides an out-of-box keypad designed for use with the CapTIvate IP in this system.

2.4 Haptics

In this TI Design, the DRV2605L haptics driver is used to drive a Samsung DMJNRB1030 LRA to create various effects in response to a touch on the HMI. The DRV2605L is driven from the SimpleLink MSP432 MCU over I²C, because the MSP432 MCU is the I²C master in the system. To that end, all the drivers from the CapTIvate phone panel demonstration were plugged directly into the source code of the MSP432 MCU. Figure 11 illustrates the I²C bus organization. See for software implementation details.

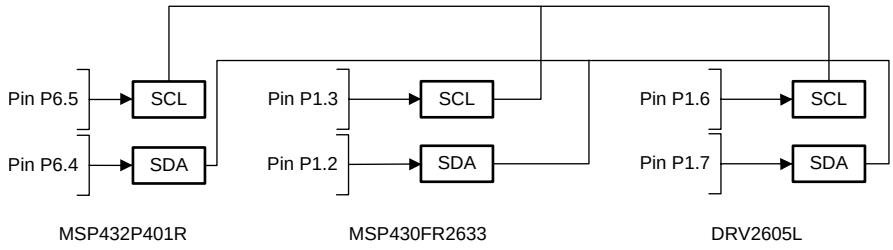


Figure 11. I²C Bus Block Diagram

2.5 LCD and Power Saving

The SimpleLink MSP432 MCU updates the Kentec QVGA Display BoosterPack over SPI. The MSP432 MCU also has the ability to turn the LCD on and off using the driver API found in the /lcd_driver folder within the main project directory. Because the LCD consumes a large amount of current, relative to the rest of the system, and because power consumption is a major consideration in this reference design, the ability to turn the LCD on and off while the system idles or sleeps is vital to extending battery life, extending servicing intervals, and improving end equipment specifications. The trigger to keep the LCD on is set in DEMO_TIMEOUT_taskFxn(). If the LCD shutoff timer elapses then the LCD shuts off. For more information, see Section 3.2.3.4 on the LCD shutoff timer and DEMO_TIMEOUT_taskFxn().

3 Getting Started Hardware and Software

The TI Store sells each component of the TIDM-1004 design separately.

3.1 Getting Started Hardware

3.1.1 SimpleLink MSP-EXP432P401R LaunchPad Development Kit

The SimpleLink MSP432P401R LaunchPad Development Kit, available at the TI Store, enables the development of high-performance applications benefitting from low-power operation. The kit features the MSP432P401R, which includes the following:

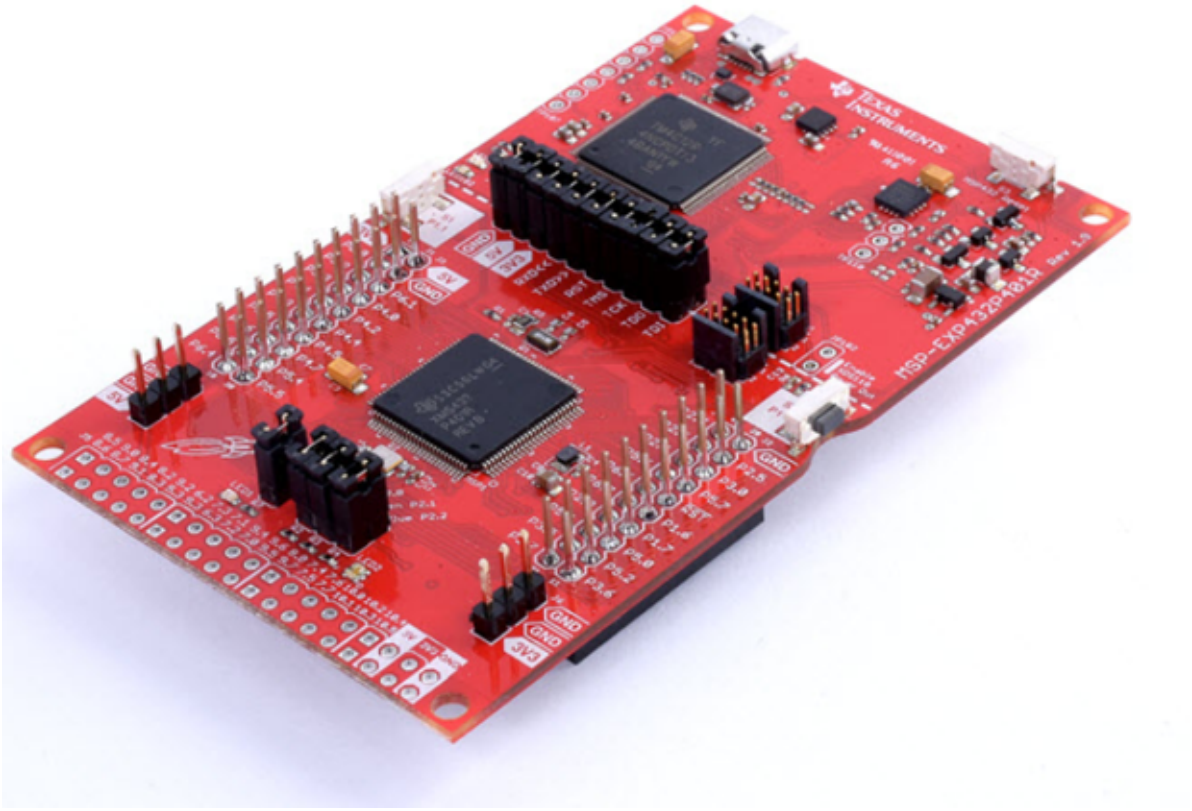


Figure 12. MSP432 LaunchPad

- 48-MHz ARM Cortex-M4F
- 95-µA/MHz active power
- 850-nA RTC standby operation
- 24-channel, 14-bit differential, 1-msps differential SAR ADC
- Advanced encryption standard (AES256) accelerator

This LaunchPad includes an onboard emulator with EnergyTrace+ Technology, meaning the LaunchPad can program and debug projects without needing additional tools, while also measuring total system energy consumption.

3.1.2 MSP-CAPT-FR2633 MCU Development Kit

The MSP-CAPT-FR2633 MCU Development Kit (see Figure 13), available for purchase at the TI Store, is used to evaluate the CapTIvate technology in a wide range of capacitive touch configurations.

Figure 13. MSP-CAPT-FR2633 MCU Development Kit

The kit includes:

- CAPTIVATE-FR2633 Target MCU Module
- Self-capacitance touch panel
- Mutual-capacitance touch panel, referred to in this guide as the CapTIvate Phone touch panel
- Proximity-sensing panel

These touch panels interface directly into the CAPTIVATE-FR2633 board and come with premade demonstration projects for easy plug-and-play use.

3.1.3 SimpleLink Bluetooth Low Energy CC2650 BoosterPack Plug-In Module

The BOOSTXL-CC2650MA SimpleLink Bluetooth low energy BoosterPack, available at the TI Store, is a plug-in module for adding Bluetooth low energy communications to your LaunchPad design. This BoosterPack comes pre-certified with the FCC, and also includes pre-loaded software, as well as a JTAG cable to connect to the XDS110 JTAG emulator on the MSP-EXP432P401R for programming and debugging the module.

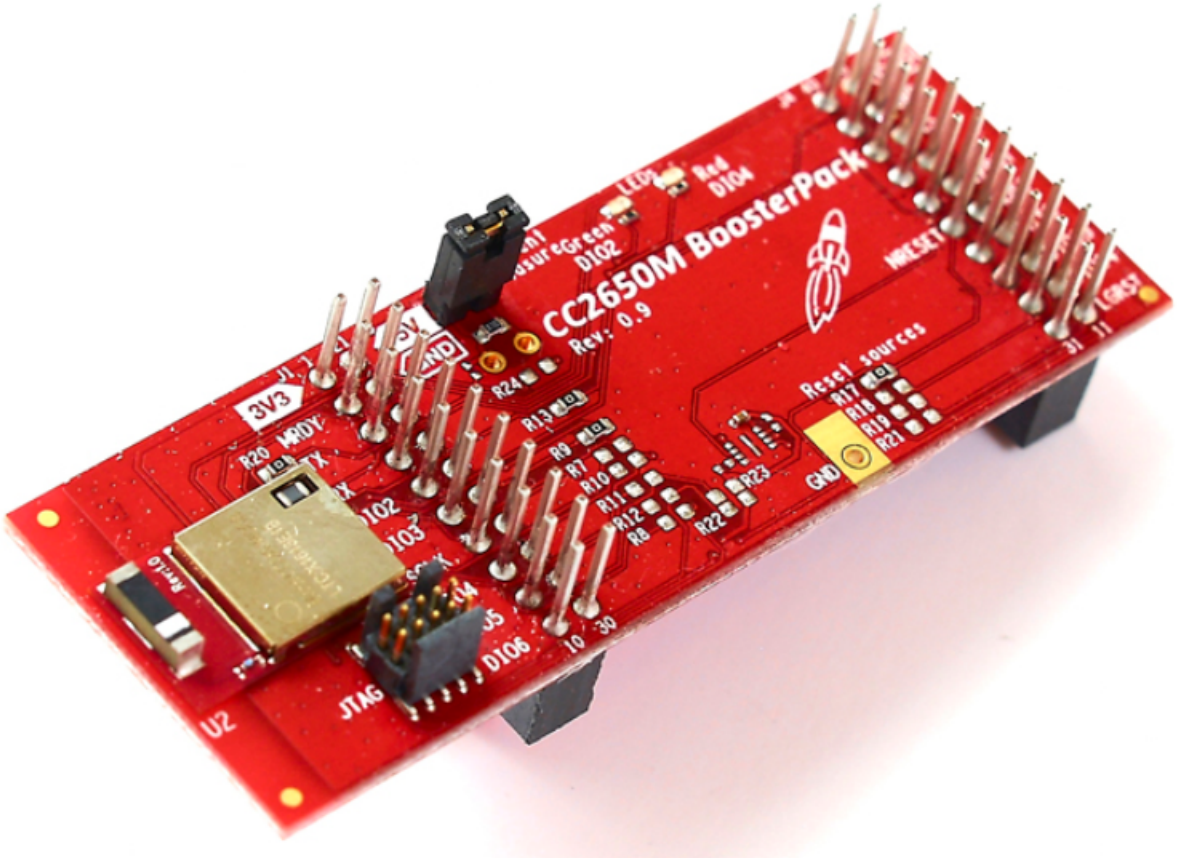


Figure 14. CC2650 BoosterPack

3.1.4 BOOSTXL-K350QVG-S1 Kentec QVGA BoosterPack

The BOOSTXL-K350QVG-S1 Kentec QVGA Display BoosterPack, available at the TI Store, is an easy-to-use plug-in module for adding a touchscreen color display to the LaunchPad design. MCU LaunchPad developers can use this BoosterPack to start developing applications using the 320×240 pixel, SPI-controlled, TFT QVGA display with resistive touchscreen.

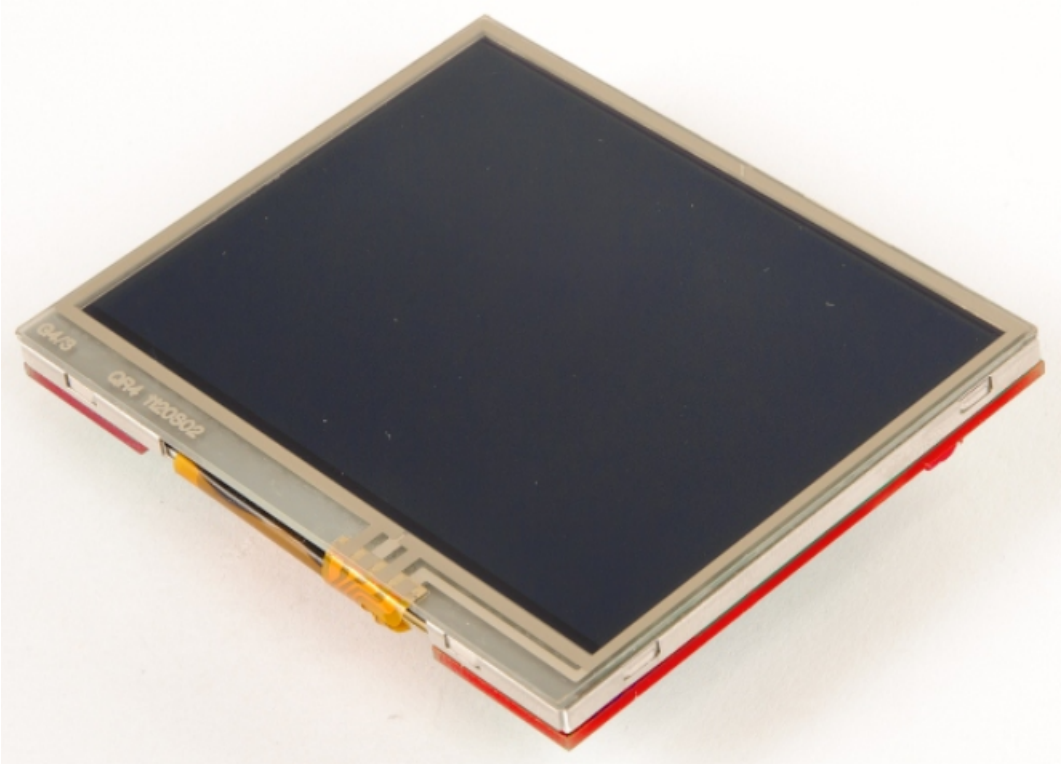


Figure 15. Kentec QVGA BoosterPack

3.1.5 How To Assemble The Reference Design Hardware Stack

The hardware stack of the access panel is assembled by attaching the BoosterPacks to the SimpleLink MSP432 LaunchPad. The MSP432 LaunchPad can attach to the BoosterPacks by using either the headers on the top of the board or the header sockets on the bottom of the board.

TI recommends connecting the MSP432 LaunchPad on top of the CapTIvate MSP430FR2633 controller board using the LaunchPad header pins. From there, stack the CC2650MODA BoosterPack on top of the MSP432 LaunchPad, and then finally attach the LCD on top of the CC2650 BoosterPack. TI recommends this setup so the user has easier access to the JTAG ports and JTAG jumper pins of the MSP432 MCU, and the current measurement jumper pins of the CC2650 MCU. See Figure 16 for more detail.

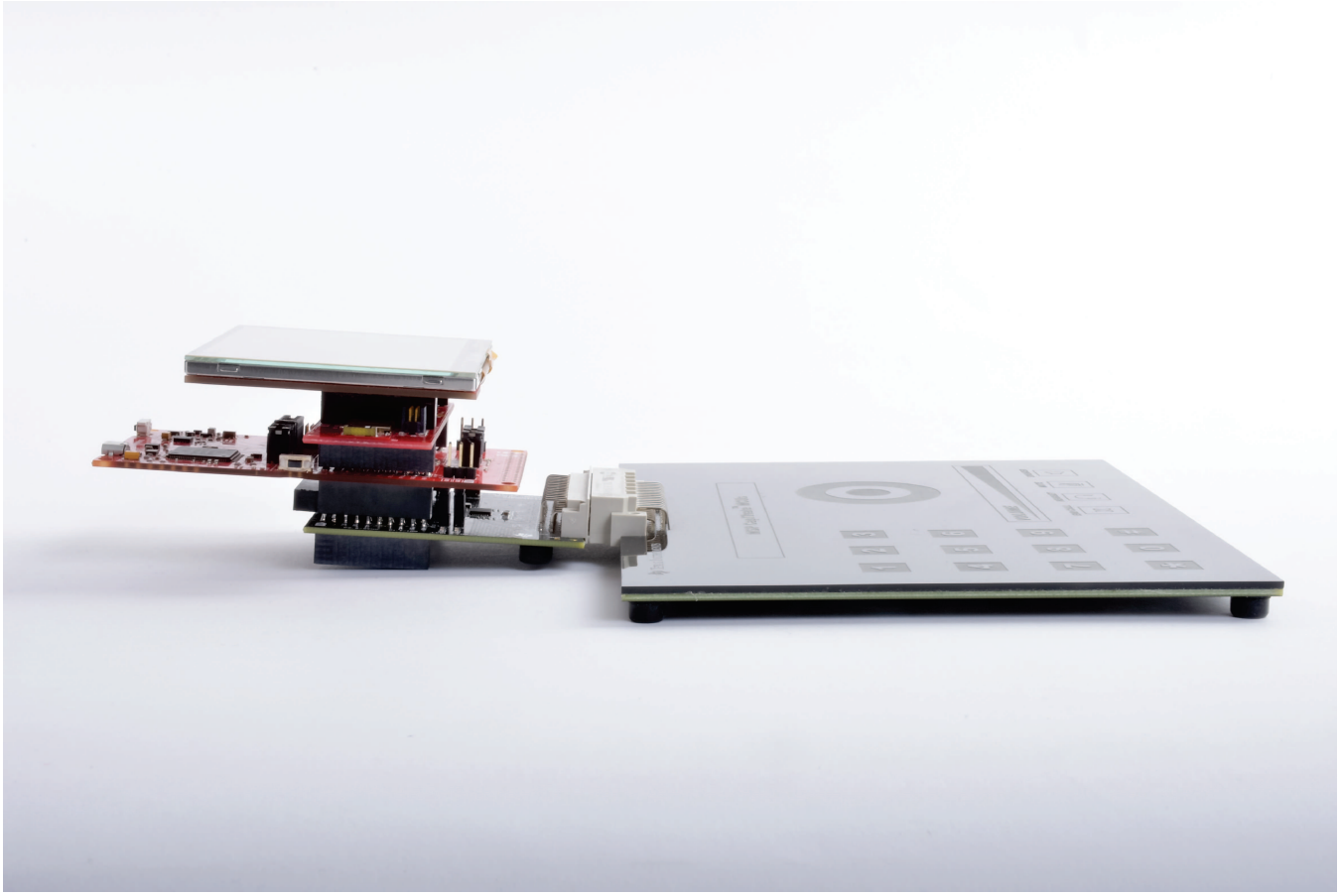


Figure 16. TIDM-1004 Hardware Stack, Side Profile

For more information, see the LaunchPad Ecosystem Page.

3.1.6 TIDM-1004 Pin Configurations

This reference design uses a LaunchPad and multiple BoosterPacks with interlocking pin configurations based on the ports available for use through the LaunchPad footprint. Because BoosterPacks may have conflicting pin configurations, Table 2 has been provided here to show the factory pin configuration across the entire system. Cells shaded in green are inputs to a LaunchPad or a BoosterPack. Cells shaded in yellow are outputs from a LaunchPad or a BoosterPack. Cells shaded in cyan are left as floating pins. Cells shaded in red are not connected (NC) to the LaunchPad footprint. Cells left white are open-drain or are unused in this TI Design.

Table 2. TIDM-1004 Pin Configuration

PIN	DEMO FUNCTION	CAPTIVATE-FR2633	MSP-EXP432P401R	BOOSTXL-CC2650MA	KENTEC LCD BP
1	3.3-V supply	3.3-V supply	3.3 V supply	3.3-V supply	3.3-V supply
2	—	NC	MRDY to CC2650 [OUT]	MRDY [IN]	NC
3	UART from CC2650	[IN]	UCA2RX [IN]	TX [OUT]	NC
4	UART from MSP432	[IN]	UCA2TX[OUT]	RX [IN]	NC
5	IRQ CapTIvate to 432	P2.2 IRQ [OUT]	IRQ [IN]	NC	NC
6	—	NC	LOW [OUT]	NC	NC
7	SPI clock to LCD	NC	UCB0CLK [OUT]	SPI CLK [IN]	LCD_SCL [IN]
8	LCD CMD CTRL	[IN]	CTRL to LCD [OUT]	DIO21	LCD_SDC
9	I2C Bus SCL	UCB0SCL [OPEN DRAIN]	UCB1SCL [OPEN DRAIN]	NC	NC
10	I2C Bus SDA	UCB0SDA [OPEN DRAIN]	UCB1SDA [OPEN DRAIN]	NC	NC
11	Proximity LED	LOW [OUT]	Floating	NC	Y–
12	Touch LED	LOW [OUT]	Floating	NC	NC
13	LCD CS and DRV ENABLE	[IN]	LCD and DRV CS [OUT]	DIO13	LCD_SCS [IN]
14	—	NC	LOW [OUT]	MISO [OUT]	NC
15	LCD data IN	NC	UCB0MOSI [OUT]	MOSI [IN]	LCD_SDI [IN]
16	—	NC	RST	NC	NC
17	—	NC	LOW [OUT]	NC	NC
18	—	NC	LOW [OUT]	SPI CS [IN]	NC
19	—	NC	SRDY [IN]	SRDY [OUT]	NC
20	GND	GND	GND	GND	GND
21	5-V supply	5-V supply	5-V supply	5-V supply	5-V supply
22	GND	GND	GND	GND	GND
23	—	NC	LOW [OUT]	NC	Y+
24	—	NC	LOW [OUT]	NC	X+
25	—	NC	LOW [OUT]	NC	NC
26	—	NC	LOW [OUT]	NC	NC
27	—	NC	LOW [OUT]	NC	NC
28	—	NC	LOW [OUT]	NC	NC
29	—	NC	LOW [OUT]	NC	NC
30	—	NC	LOW [OUT]	NC	NC
31	—	NC	LOW [OUT]	NC	X–
32	—	NC	LCD reset [OUT]	NC	LCD_RESET
33	—	NC	LOW [OUT]	NC	NC
34	—	NC	LOW [OUT]	NC	NC
35	—	NC	LOW [OUT]	nRESET	NC
36	—	NC	LOW [OUT]	NC	NC
37	—	NC	LOW [OUT]	NC	NC
38	—	NC	LOW [OUT]	NC	NC
39	—	NC	LOW [OUT]	NC	NC
40	—	NC	LOW [OUT]	NC	PWM

3.2 Getting Started Software

The example software was developed using CCS7 and above, as well as the TI MSP432 Compiler version 16.9.0.LTS. Download the latest version of CCS to evaluate the example software to import the projects into a CCS workspace from {TI Software Install Root}/*.

The software package that contains all the specific source files and libraries for this reference design is in the main project, ble_lock_dev_MSP_EXP432P401R_tirtos_ccs. This project uses the SimpleLink MSP432 SDK and the SimpleLink MSP432 SDK Bluetooth Plugin. The example project for the MSP430FR2633 MCU includes associated CapTIvate Design Center project files. Download the CapTIvate Design Center to view detailed sensor data, configure and tune sensor performance, and perform signal-to-noise ratio (SNR) measurements for the design in real time. CapTIvate Design Center project files can open from the same directory as their respective CCS project. See Section 3.2.4 for a description of the MSP430FR2633 software.

The example project, Simple Application Processor, was used as a reference for this TI Design. For more information, see the SimpleLink MSP432 SDK Bluetooth low energy Plugin examples folder.

The software used to command the DRV2605L haptics driver is included within the CapTIvate Software examples. In this TI Design, the software is driven from the SimpleLink MSP432, the I^2C master in the system. See Section 3.2.2 for details.

3.2.1 Software Architecture

For this reference design, the software is divided into two parts: host side (SimpleLink MSP432 MCU) and subordinate side (MSP430FR2633 MCU). There is one projects for the host side and one example project for the HMI side. To import the projects into CCS, ensure to install the SimpleLink MSP432 SDK and the SimpleLink MSP432 SDK Bluetooth Plugin first. Then, import the projects by using the CCS Import Function found in the File drop-down menu. The projectspec files for both the MSP430 and the MSP432 code are located in the development_projects folder, within the directory of the reference design. Ensure to import only the release projects and not the development projects.

The first project, blelock_MSP_EXP432P401R_tirtos_ccs, is the reference design's source for managing and executing all software running on the MSP432 MCU. The source runs on top of the SimpleLink MSP432 SDK and the SimpleLink MSP432 Bluetooth low energy plugin, with an internal TI-RTOS (SYS/BIOS) kernel packaged within the SDK. This project also interfaces to a CapTIvate BoosterPack, a KentecLCD BoosterPack for a real-time display, and the CC2650 Bluetooth low energy BoosterPack to enable communications with mobile devices over Bluetooth low energy. The software is described in detail in Figure 52, with haptics described in Section 3.2.2. Figure 17 shows the architecture diagram for the TIDM-1004.

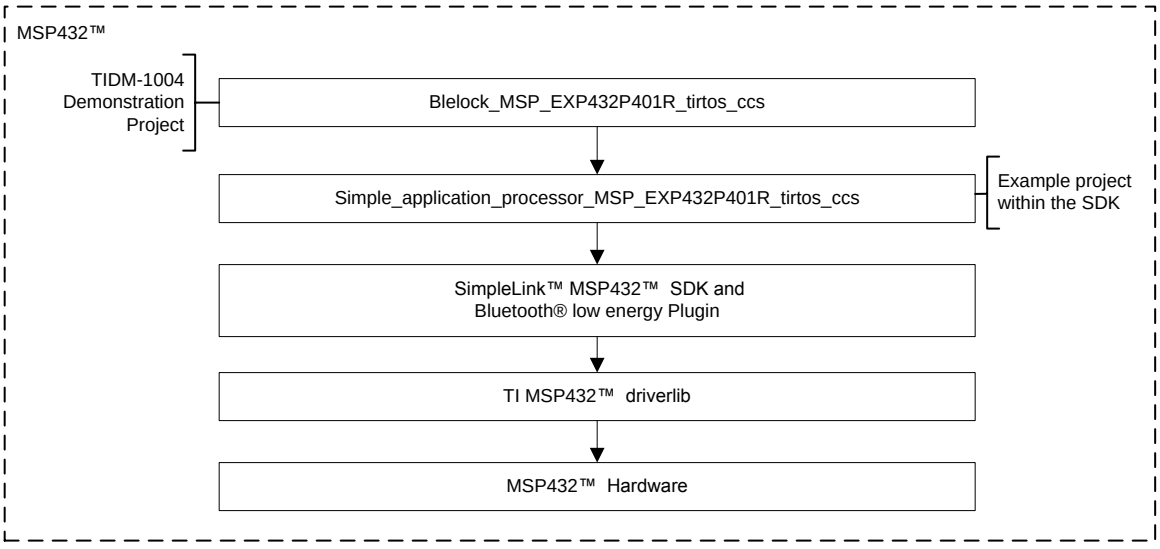


Figure 17. TIDM-1004 Software Architecture Diagram

This project is built upon the SAP, which demonstrates using the BOOSTXL-CC2650MA BLE BoosterPack to enable Bluetooth low energy communications using a simple UART interface between the MS432 MCU and the CC2650 device. This project is the base example on which the MSP432 software was written. The project is in the SimpleLink MSP432 BLE Plugin directory, under the example folder: {TI install directory} \simplelink_msp432_sdk_bluetooth_plugin_{version}\examples\rtos \MSP_EXP432P401R\bluetooth\simple_application_processor.

The SAP is a simple code example that showcases basic functionality of the SimpleLink MSP432 SDK Bluetooth Plugin. Simple Bluetooth low energy concepts such as characteristic manipulation and property notifications are examined. This code example is intended to be a foundational implementation that can be used as a basis for more advanced applications. The code example specifically controls an LED based upon data received over Bluetooth from a mobile device running any generic Bluetooth low energy testing applications. The SAP software of the MSP432 MCU described in Section 3.2.3.1 is based off of this example.

The project, blelock_captivate, runs on the MSP430FR2633 MCU. The project communicates with a host through I^2C.

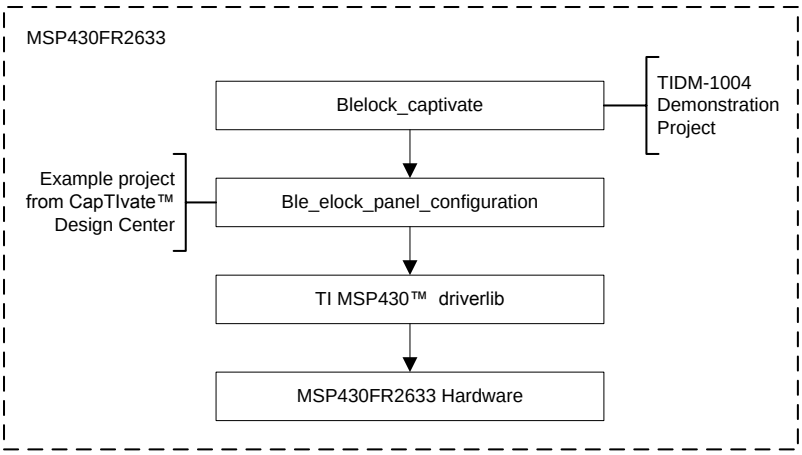


Figure 18. TIDM-1004 MSP430 Software Architecture Diagram

As mentioned previously, the MSP430FR2633 software is built upon the CapTIvate Software Library. This software must be loaded on to the MSP430FR2633 MCU through the provided CapTIvate programmer from the CapTIvate MCU development kit. For details on the kit, see Section 3.1.1. The blelock_captivate project is based upon existing code automatically generated by the CapTIvate design center, with a custom I^2C interface. See Section 3.2.4 for details on the blelock_captive project.

The example software from the CapTIvate Design Center initializes the MSP430FR2633 MCU with the information for the keypad, wheel button, and the proximity channel sensors. The keypad is a mutual capacitance sensor, which uses 3 transmission strips and 4 reception strips to make a grid of 12 intersections, with each intersection representing a keypad. The wheel button is one mutual capacitance sensor with one transmission and one reception sensor. The proximity sensor is a self-capacitance sensor with just one reception line to the MSP430FR2633 MCU.

There are five sequential sampling cycles, with the first three dedicated to the keypad, aligned with firing one transmission impulse at a time. The fourth sampling cycle is dedicated to the proximity sensor, with just one reception line for the MSP430FR2633 MCU to check. The fifth and final cycle is reserved for the wheel button sensor, with one transmission impulse and one reception impulse to check for activity.

Controller		Sensors				Time Cycles				
Port	Use Mode	Parallel Block	numericKey...	proxAndGu...	wheelButto...	numericKey...	numericKey...	numericKey...	proxAndGu...	wheelButto...
CAP0.0	Unrestricted	B0	RX00			E00	E04	E08		
CAP0.1	Unrestricted	B1	TX00			TX				
CAP0.2	Unrestricted	B2	TX01				TX			
CAP0.3	Unrestricted	B3	TX02					TX		
CAP1.0	Unrestricted	B0	RX01			E01	E05	E09		
CAP1.1	Unrestricted	B1								
CAP1.2	Unrestricted	B2			TX00					TX
CAP1.3	Unrestricted	B3								
CAP2.0	Unrestricted	B0	RX02		RX00	E02	E06	E10		E00
CAP2.1	Unrestricted	B1								
CAP2.2	Unrestricted	B2								
CAP2.3	Unrestricted	B3								
CAP3.0	Unrestricted	B0	RX03			E03	E07	E11		
CAP3.1	Unrestricted	B1								
CAP3.2	Unrestricted	B2								
CAP3.3	Unrestricted	B3		RX00					E00	

Figure 19. CapTIvate Phone Touch Panel Sensor Configuration

3.2.2 Haptic Feedback

This demonstration includes haptics to provide users with mechanical feedback if they touch a key. The most common and cost-effective actuators are ERMs and LRAs. The LRA provides a higher quality vibration feel than the ERM. Because of the higher quality vibration, LRAs are more common with consumer products. However, there are advantages to an ERM. LRAs have a limited lifetime of clicks so they are not as suitable for long life cycle products as an ERM. Therefore, ERMs are better for long-life products such as industrial control panels.

The DRV2605L is the driver IC for this demonstration for the following reasons:

- Integrated ROM effect libraries are pre-licensed from Immersion® Corporation
- Supports both ERM and LRA haptic actuators
- Simple I^2C register interface

The software package of this reference design includes a DRV26x driver, which communicates with the DRV2605L. The DRV2605L is fired from the display task, LCD_taskFxn(), because of hardware constraints; namely the SimpleLink MSP432 is the I^2C master in the system, and pin P5.0 on the MSP432 is shared between the DRV26x and the LCD. For information regarding the state machine of LCD_taskFxn(), see Section 3.2.3.2. Setting up the DRV2605L with these modules is accomplished at the top of the LCD_taskFxn() function in the lcd_task.c file.

The MSP432 device uses the DRV26x drivers to trigger playback of haptic events directly from the library. The following code shows portions of LCD_taskFxn(), where the DRV26x API is called. Effects are fired on a new touch, if *touch* is true and *previous touch* is false. Effects are also fired upon a successful lock, a successful unlock and an unsuccessful unlock.

The DRV26x effects are only fired from certain states within LCD_taskFxn(), namely the following:

- DRAW_WHEEL_BUTTON: 2 short responses are fired upon successful unlock.
- DRAW_NUM_KEYPPAD: 1 short response is fired for every keypad touch.
- DRAW_INVALID_PASSCODE: 1 long pulse is fired for an incorrect PIN entry.
- DRAW_INVALID_BLE_PASSCODE: 1 long pulse is fired for an incorrect PIN entry using Bluetooth.

These effects are fired using the DRV26x API within the DRV26x library, provided in the /msp432/ drv26x folder in the TIDM-1004 software.

3.2.3 SimpleLink MSP432 Software: blelock_MSP_EXP432P401R_tirtos_ccs

The main project, ble_lock_MSP_EXP432P401R_tirtos_ccs, runs on the SimpleLink MSP432 MCU and implements the functionality of this reference design with the aid of the SimpleLink MSP432 SDK and the SimpleLink MSP432 SDK Bluetooth Plugin. The project runs several tasks that control different aspects of the system, namely AP_tasnFxn() which runs the Bluetooth low energy aspect, captivate_taskFxn() which reads the CapTIvate touch panel interface, LCD_taskFxn() which runs the display aspect, and DEMO_TIMEOUT_taskFxn() which runs the system power-saving aspect. See Section 3.2.3.1 for a description of AP_taskFxn(), Section 3.2.3.2 for a description of LCD_taskFxn(), Section 3.2.3.3 for a description of captivate_taskFxn(), and Section 3.2.3.4 for a description of DEMO_TIMEOUT_taskFxn().

The four tasks communicate using semaphores and events. Semaphores are used for some general signaling and handoff procedures, and events are used to signal specific conditions to trigger a transition to a particular state within the task. Table 3 and Table 4 list the semaphores and events used for inter-task communication.

Table 3. Semaphores in MSP432 Software

SEMAPHORE TYPE	SEMAPHORE NAME	TASKS TRANSFERRED FROM AND TO
SYS/BIOS Binary Semaphore	lcdWaitForDisplay	From AP_taskfxn() to LCD_taskFxn()
		From captivate_taskFxn() to LCD_taskFxn()
SYS/BIOS Binary Semaphore	lcdUpdateSemaphore	From LCD_taskFxn() to captivate_taskFxn()
		From captivate_taskFxn() to LCD_taskFxn()
SYS/BIOS Binary Semaphore	demoTimeOutSemaphore	From any task to DEMO_TIMEOUT_taskFxn()
SYS/BIOS Binary Semaphore	demoStartTimeOutSemaphore	From any task to DEMO_TIMEOUT_taskFxn()
SYS/BIOS Binary Semaphore	TaskSemaHandle	From DEMO_TIMEOUT_taskFxn() to any task

Table 4. Events in MSP432 Software

EVENT TYPE	EVENT NAME	LIST OF EVENTS
SYS/BIOS Event	captEvent	CAPT_NONE
		CAPT_EVT_KEYPRESS
		CAPT_EVT_BUTTONPRESS
		CAPT_EVT_IRQ
SYS/BIOS Event	apEvent	AP_NONE (no event)
		AP_EVT_PUI (power-up indication)
		AP_EVT_ADV_ENB (Advertisement enable)
		AP_EVT_ADB_END (advertisement end)
		AP_EVT_CONN_EST (connection established)
		AP_EVT_CONN_TERM (connection ended)
		AP_EVT_OAD_ID_REQ (OAD Write Identify Request)
		AP_EVT_OAD_BLOCK_REQ (OAD Write Block Request)
		AP_EVT_OAD_SNP_IMAGE (OAD SNP image received)
		AP_EVT_BSL_BUTTON (BSL Button- LP S2
		AP_EVT_BUTTON_RESET (reset button)
		AP_EVT_BUTTON_RIGHT (RIGHT button press- LP S1)
		AP_ERROR (error)

Table 5 lists the task priorities.

Table 5. Task Priority Levels

TASK	PRIORITY
AP_taskFxn()	1
Captivate_taskFxn()	1
DEMO_TIMEOUT_taskFxn()	2
LCD_taskFxn()	2

All tasks except for DEMO_TIMEOUT_taskFxn() are disabled at the outset. The demo task sets the priority levels for the other three tasks from –1 (disabled) to the values in Table 5 using the Demo_init() function. LCD_taskFxn() is initialized first, then AP_taskFxn(), and then captivate_taskFxn(). Each task is signaled by the TaskSemaHandle semaphore.

Once all four tasks are initialized, AP_taskFxn() runs until it idles, while captivate_taskFxn() sleeps until it reads in a GPIO interrupt through a GPIO callback. Captivate_taskFxn() then waits on the captEvent event, and once captEvent receives an IRQ event, it switches into an active state or an idle state based on reading the flags indicating which way to go. AP_taskFxn() starts to run only once the S1 or S2 buttons are pressed, at which time the AP_keyHandler() or AP_bslKeyHandler() functions post events to AP_tasnFxn() to get it out of the idle state, AP_IDLE. LCD_taskFxn() runs at a lower priority than AP_tasnFxn() or captivate_taskFxn(), and it is only run when other tasks post to lcdUpdateSemaphore. LCD_taskFxn() executes the designated LCD graphics functions and then returns to the caller task by posting to lcdWaitForDisplay. Finally, DEMO_TIMEOUT_taskFxn() implements the system shutoff timer and starts to run immediately. The other tasks can control the system shutoff timer by posting to demoTimeoutSemaphore and demoStartTimeoutSemaphore. See Figure 20 for a description of the tasks' triggering states and priority levels.

Figure 20 shows the task trigger diagram.

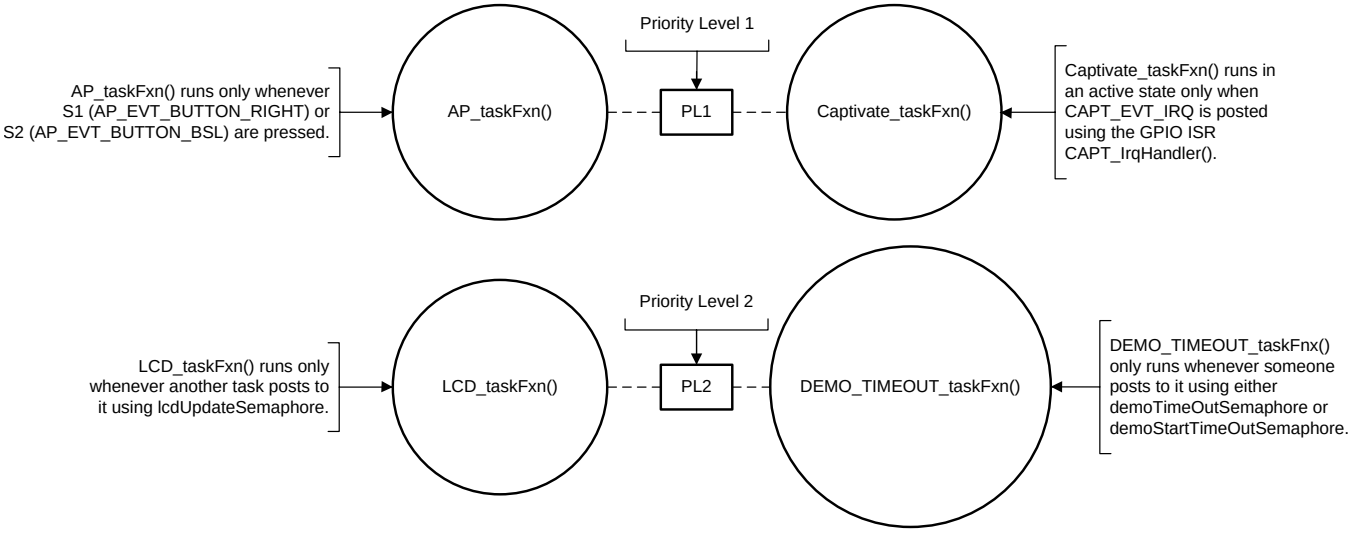


Figure 20. TIDM-1004 Task Trigger Diagram

Lastly, if there is no activity from either the touch panel or the mobile device through Bluetooth, then DEMO_TIMEOUT_taskFxn() runs the timer and shuts off the LCD screen, haptics, and RTC when the timer elapses.

3.2.3.1 Bluetooth Low Energy

The Bluetooth low energy aspect is implemented by the Bluetooth low energy task AP_taskFxn. This task manages communication on the SimpleLink MSP432 MCU when communicating with the CC2650 device, which runs example firmware as a SNP. For reference, this firmware is in a binary file and is referred to as the SNP hex image.

The diagram in Figure 21 details which apEvent events trigger a change of state.

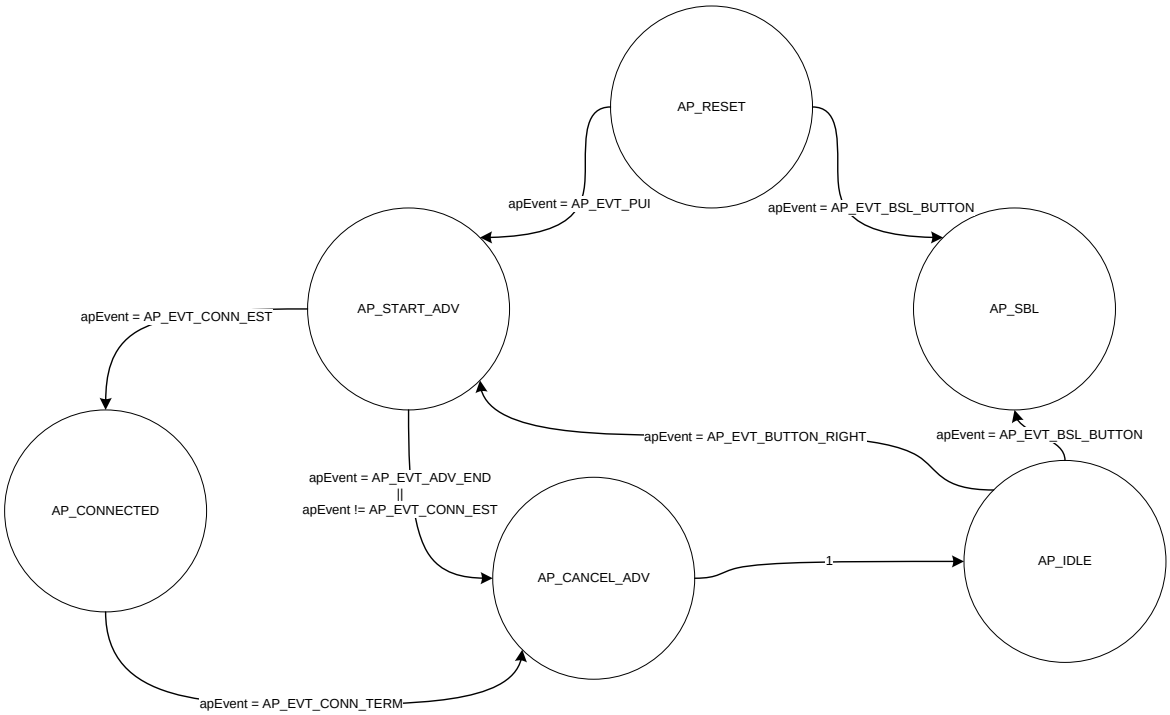


Figure 21. AP_taskFxn() State Machine

There are seven states to AP_taskFxn:

- AP_RESET: the reset state for the Bluetooth low energy task
- AP_IDLE: the state when the task is idling
- AP_START_ADV: the state when the Bluetooth low energy module starts to broadcast to any Bluetooth-enabled devices in near proximity
- AP_CONNECTED: the state when the device is connected to a Bluetooth-enabled mobile device
- AP_CANCEL_ADV: the state that turns off advertising and then returns to AP_IDLE
- AP_OAD: the over-the-air download, not yet implemented in this state machine
- AP_SBL: the boot-loader state for the CC2650 device, the MSP432 MCU flashes the CC2650 device with the SNP binary

1. AP_taskFxn() initializes, and then starts off in AP_RESET.
2. AP_RESET first checks to see that the CC2650 is not in BSL mode. Then it proceeds to open the network processor (the CC2650 device) up, set up the asynchronous callback function, and then wait for either a button press and goes to AP_SBL, or goes to AP_START_ADV.
3. From AP_IDLE, the task waits the user to press the left button on the MSP432 to activate the Bluetooth low energy BoosterPack, or the right button to re-flash the CC2650 with the SNP binary.
4. At that point, the BLE task then goes into AP_START_ADV and waits for a mobile device to connect.
5. Once connected, the task proceeds to AP_CONNECTED. The mobile device and the MSP432 can send info to each other asynchronously.
6. Once the mobile user disconnects from the device, the task proceeds to AP_CANCEL_ADV and stops advertising.
7. Finally, AP_taskFxn() transitions to AP_IDLE after waiting for AP_EVT_ADV_END, which signals that the CC2650 has stopped advertising. At that point, it waits in AP_IDLE until the user presses the left or right buttons on the MSP432 LaunchPad.

In addition to the state machine of AP_taskFxn(), a callback function reads in the input data as the passcode, and stores it appropriately, then updates the display, like captivate_taskFxn(). This callback function is AP_SPWriteCB() of the simple_application_processor.c file.

This function uses a modified version of the GATT Profile from the SAP, shown in Table 6.

Table 6. TIDM-1004 GATT Profile

CHARACTER NUMBER	PURPOSE	UUID	FORMAT	PROPERTIES	PROFILE SOURCE
1	Lock ID	0xFFF1	Integer	Read and write	Simple_gatt_profile.c
2	Lock Notification	0xFFF2	Integer	Notification	Simple_gatt_profile.c
3	Characteristic #3	0xFFF3	Integer	Write	Simple_gatt_profile.c
4	Characteristic #4	0xFFF4	Integer	Notification	Simple_gatt_profile.c

There are two cases for the write-enabled characteristics, as follows:

- SP_CHAR1: the sequential process for processing characteristic 1
- SP_CHAR3: the sequential process for processing characteristic 3

The Bluetooth characteristic callback function, AP_SPWriteCB(uint8_t charID), grabs the incoming passcode from characteristic 1 if there has been a write event on the Bluetooth low energy connection. The function then grabs the entered passcode using SimpleProfile_GetParameter(), and checks to verify if it matches the stored passcode. If the entered passcode matches the stored passcode, it unlocks the device and signals to LCD_taskFxn() to draw the unlocked lock on the screen corresponding to the freshly toggled lock state variable. Otherwise, the passcode is wrong and the function signals to LCD_taskFxn() to flash *Invalid BLE Passcode* to the screen and refresh the screen in the last saved state.

Figure 22 shows the flow diagram of AP_SPWriteCB().

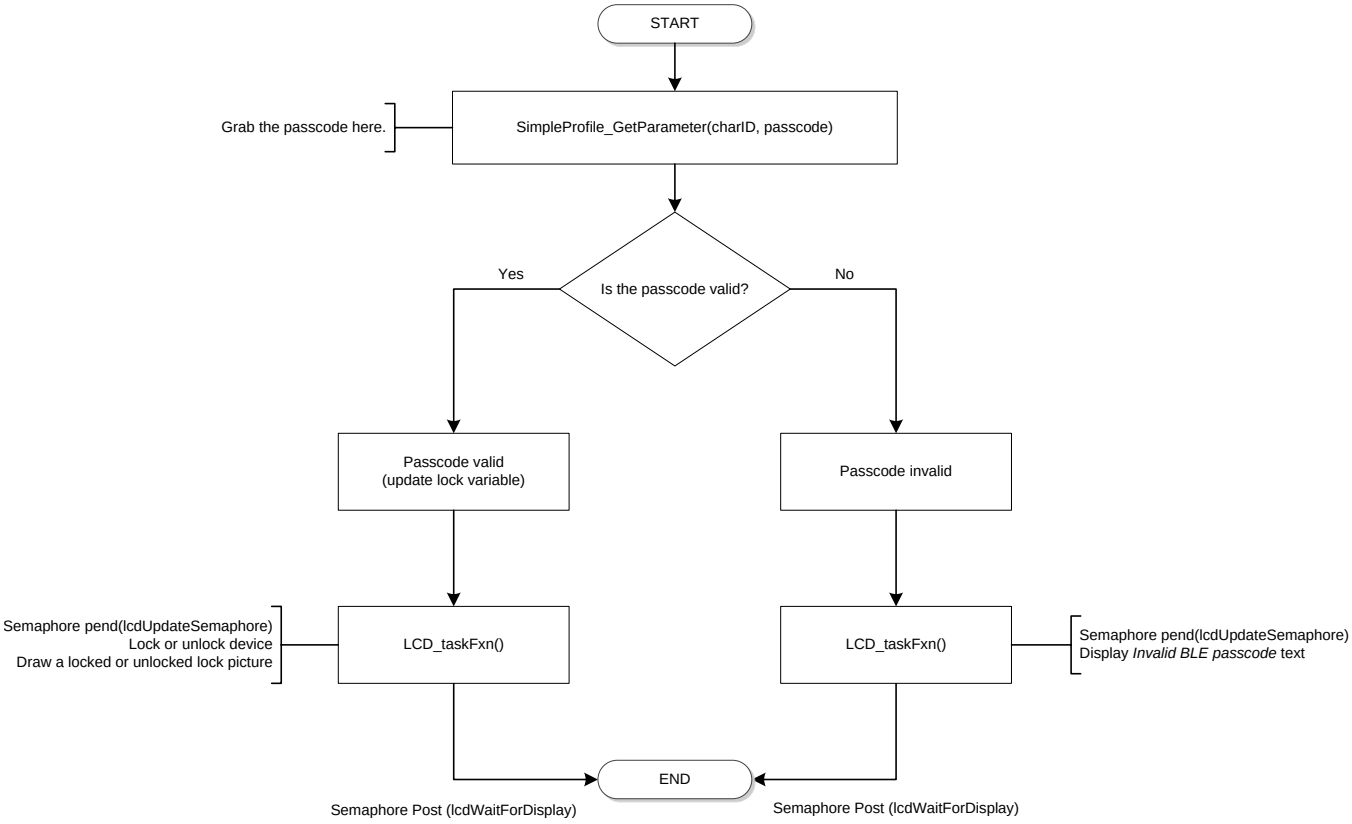


Figure 22. AP_SPWriteCB() Flow Diagram

In Figure 22, states that are inside the LCD_taskFxn() state machine are represented by octagons, whereas states inside AP_SPWriteCB() are represented by circles.

For characteristic 3, the function just uses SimpleProfile_GetParameter() to grab the written value and store it in a buffer in memory, like in the Bluetooth plug-in example, SAP. No processing is done on characteristic 3.

Specialized applications that are developed to complement the SimpleLink MSP432 SDK Bluetooth Plug-In are planned for a follow-on release; however, there are generic Bluetooth applications available that give users a lower-level way to experience the code examples. For these generic applications, TI recommends:

- LightBlue Explorer for iOS
- BLE Scanner for Android

To use the BLE on an Android device do the following:

1. Install the application.
2. Press the S1 button on the MSP-EXP432P401R LaunchPad to start advertising the access panel over Bluetooth.
3. Open the BLE Scanner (see Figure 23).

NOTE: The same procedure works with the LightBlue™ Explorer (see Figure 24).

4. Refresh the scanner. The access panel should show up as *MSP432 SAP* in the window.

5. Press connect to connect to the access panel through the CC2650 MCU (see Figure 23 and Figure 24).

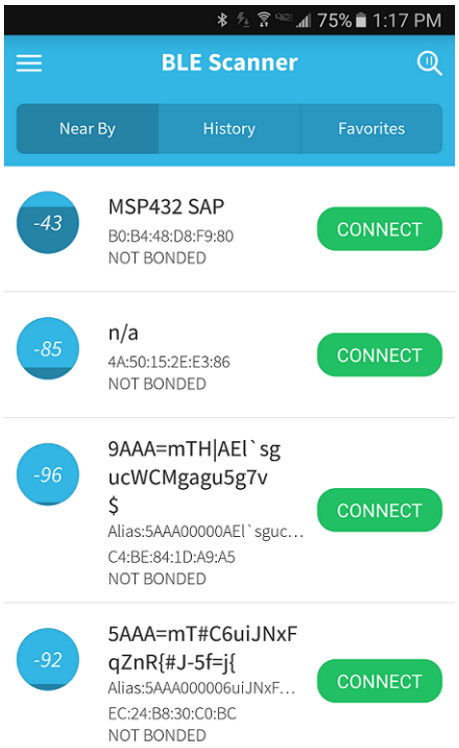


Figure 23. BLE Scanner Scan Page (Android)

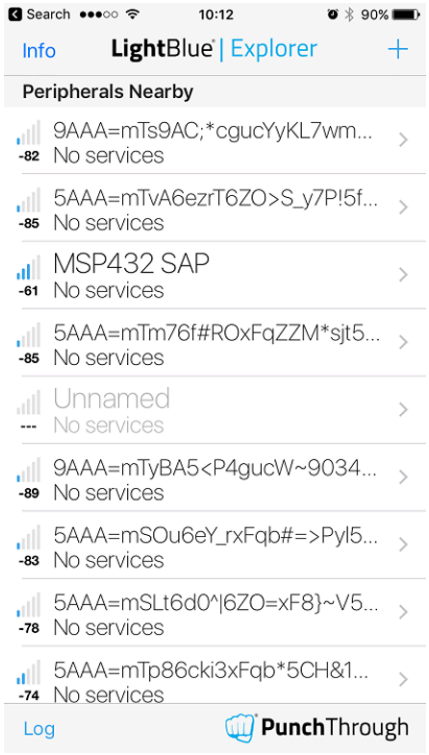


Figure 24. LightBlue Scan Page (Apple)

6. From there, navigate down to the *Custom Service Tab*.
7. Press on the tab to unfold it. There are two characteristics: Lock ID and Notification. Notification provides the lock state, and lock ID is the passcode buffer that the mobile device writes to.

   Figure 25 shows the MSP432 SAP on Android, and Figure 26 shows the MSP432 SAP on iOS. For iOS, the Bluetooth service is already unfolded and is at the bottom of the window.

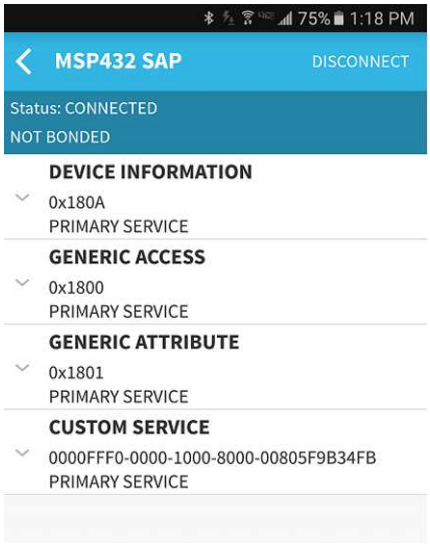


Figure 25. MSP432 SAP Connected (Android)

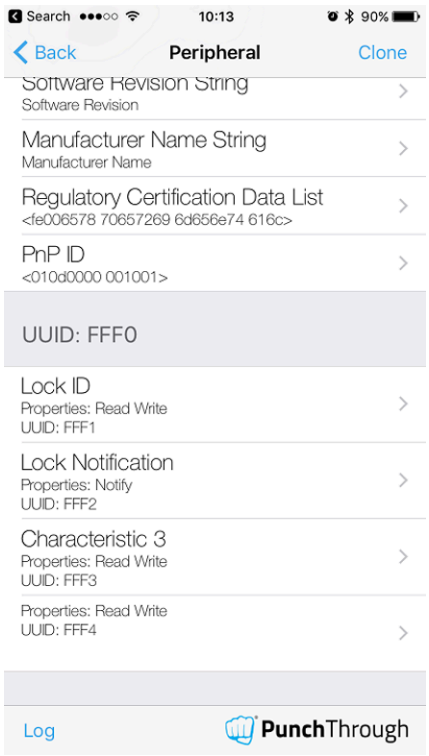


Figure 26. MSP432 SAP Connected (Apple)

8. Press the *N* button in the circle under the Notification characteristic to turn it on (see Figure 27). On LightBlue, click on the *Listen for Notifications* button (see Figure 28).

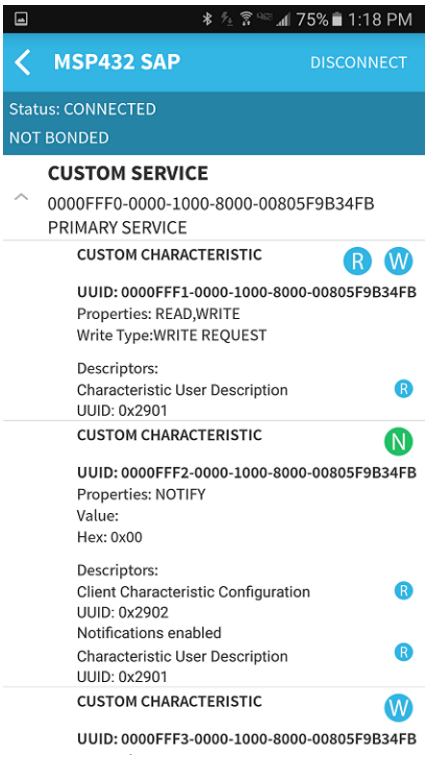


Figure 27. Listening to Notification Characteristic (Android)

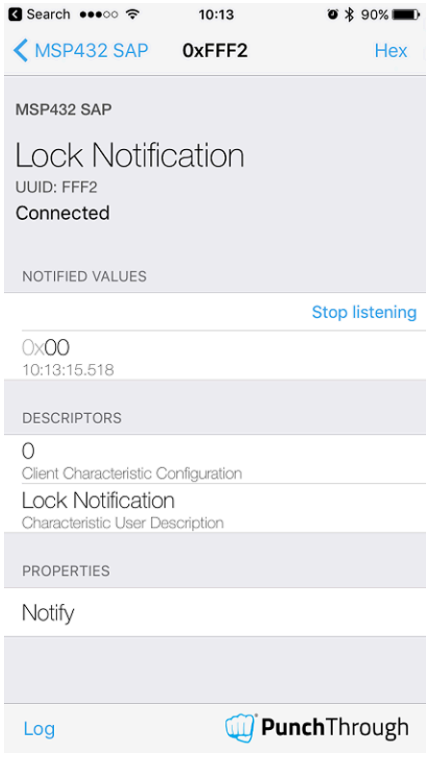


Figure 28. Listening to Notification Characteristic (Apple)

9. Then, press the Lock ID characteristic to bring up a window. From there, select the *byte array* option. Finally, write *1234* to lock and unlock the access panel through Bluetooth (see Figure 29 and Figure 30).

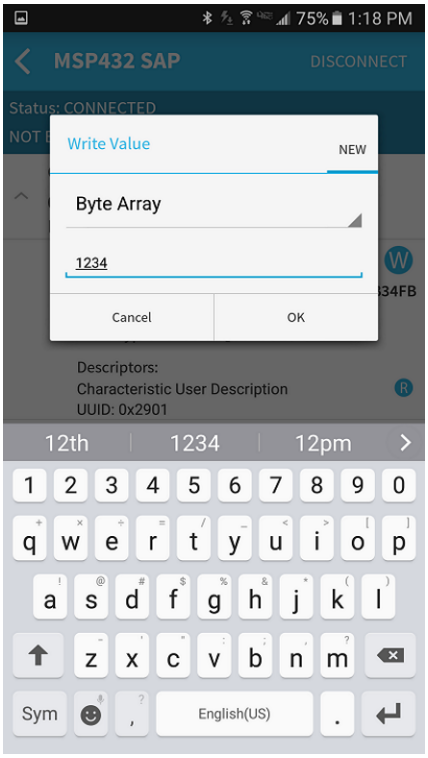


Figure 29. Entering Passcode (Android)

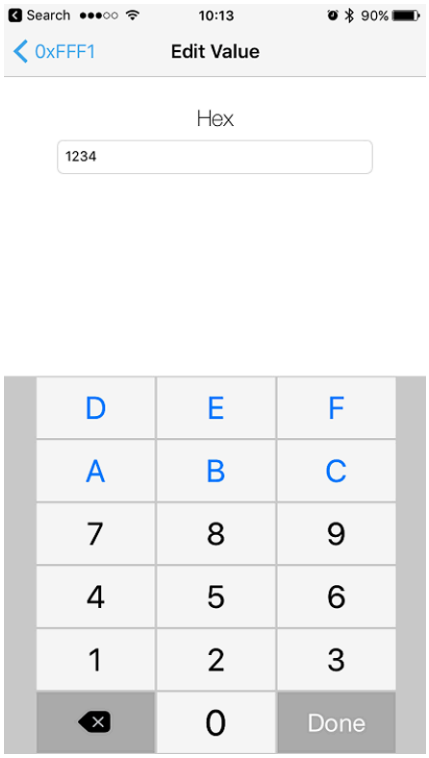


Figure 30. Entering Passcode (Apple)

Figure 31 and Figure 32 show what the final screen should look like, if everything works.

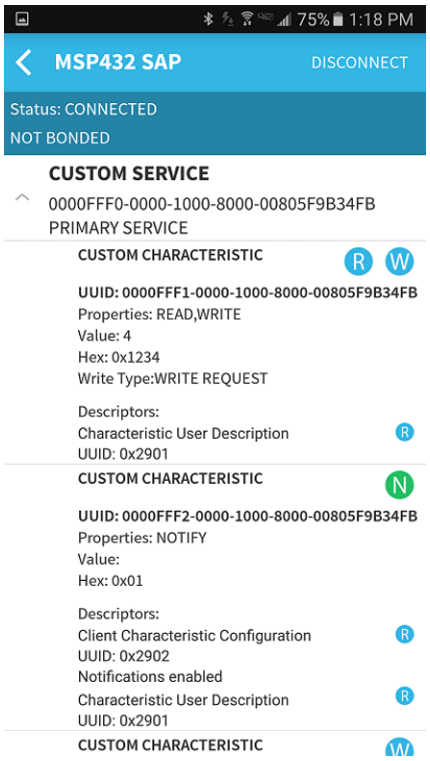


Figure 31. Device Locked Successfully (Android)

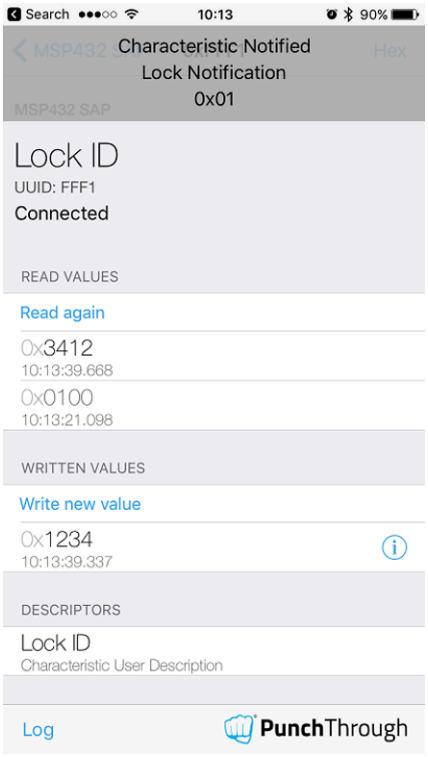


Figure 32. Device Locked Successfully (Apple)

The CC2650 device runs TI-proprietary firmware supplied by the SimpleLink MSP432 SDK Bluetooth Plugin. The firmware contains a stack for Bluetooth communication, and can be treated as a "black box" from the perspective of the MSP432 MCU. This firmware is provided in a hex file from {Plugin install directory}/source/ti/snp and can be re-programmed to the CC2650 from the MSP432 using the onboard XDS110 JTAG emulator on the MSPEXP432P401R device, or stored as an array in C inside the source file simple_np_cc2650bp_pm_merge.c within the CCS project directory. To program from the XDS110 JTAG emulator, the SmartRF Flash Programmer is required. To copy the converted hex image into a C array in simple_np_cc2650bp_pm_merge.c, TI recommends the open source SRecord utility.

For more detailed information on the SAP, see the provided README.html file inside the project directory at: {TI Installation Directory}/simplelink_msp432_sdk_bluetooth_plugin_1_00_00_81/examples/rtos/MSP_EXP432P401R/bluetooth/simple_application_processor/README.html. For more detailed information on reprogramming the CC2650 device, see the *Bluetooth Plugin User's Guide* at {Plugin install directory}/docs/users_guide_simplelink_msp432_sdk_plugin.html.

3.2.3.2 Display

The display aspect is implemented by the display task, LCD_taskFxn(). After getting the semaphore signal from AP_taskFxn, LCD_taskFxn() starts by drawing the initial display, shown in Figure 33.

Figure 33. Access Panel Screen Initialization

Then, the task waits for semaphore signals from captivate_taskFxn() and AP_SPWriteCB(), depending on what images the tasks need to draw.

Figure 34 shows the LCD_taskFxn() states.

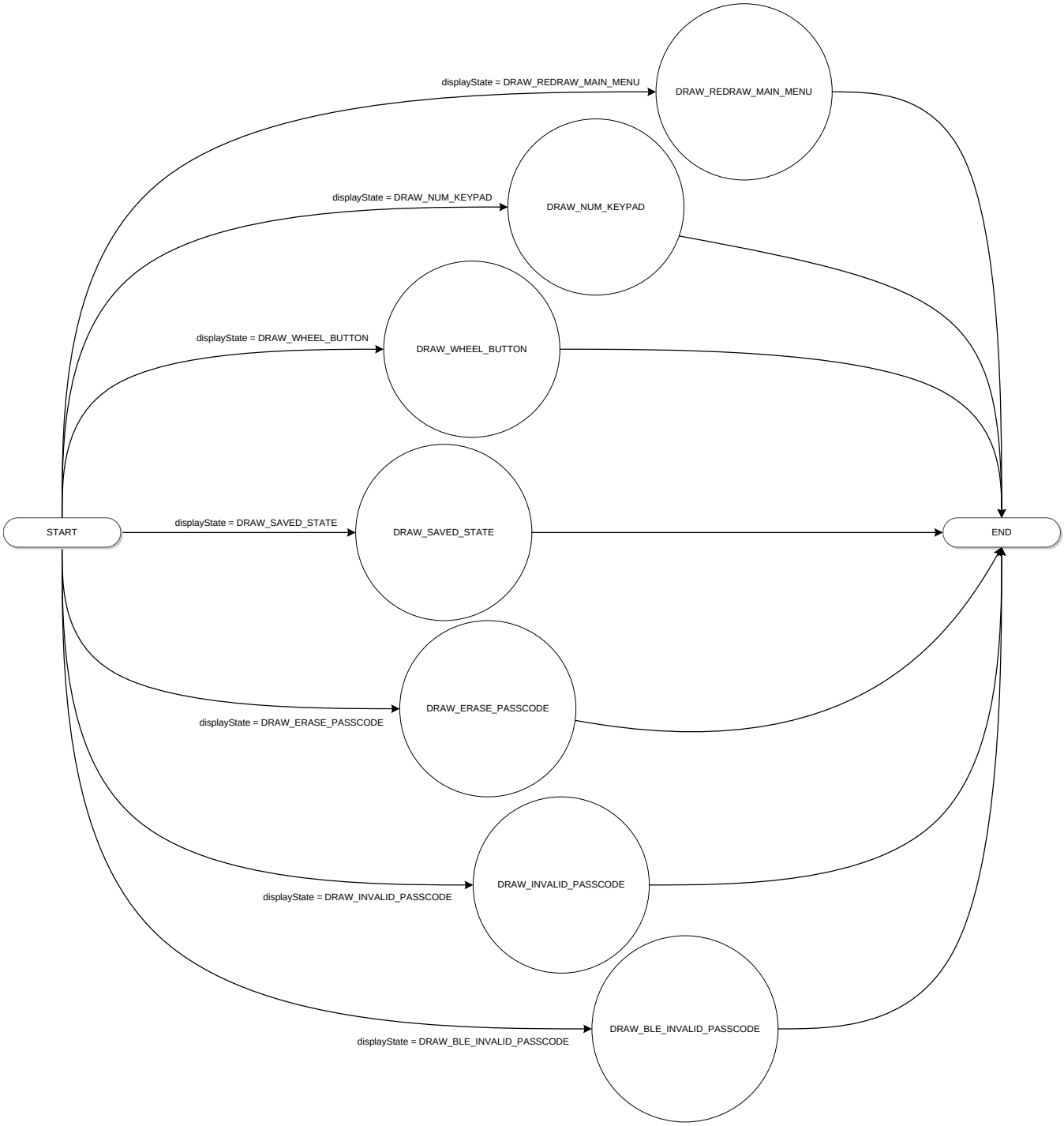


Figure 34. LCD_taskFxn() State Machine

There are seven states:

- DRAW_SAVED_STATE: draws the last saved state of the device. The lock symbol is drawn if the lock state variable is set accordingly.
- DRAW_NUM_KEYPAD: accepts the index of the pressed button from the CapTIvate keypad, decodes it to the proper value, saves it to the passcode buffer, and then draws a star for each key entered.
- DRAW_WHEEL_BUTTON: draws the lock depending on the state of the lock state variable. If the device is locked, DRAW_WHEEL_BUTTON draws a lock; if not, DRAW_WHEEL_BUTTON draws an unlocked lock.
- DRAW_INVALID_PASSCODE: flashes a notification to the center of the display reading *INVALID PASSCODE*.
- DRAW_BLE_INVALID_PASSCODE: flashes a notification to the center of the display reading *INVALID BLE PASSCODE*.
- DRAW_ERASE_PASSCODE: erases the passcode in the passcode box on the left side of the display.
- DRAW_REDRAW_MAIN_MENU: redraws the main menu in the same state it was in before the system entered deep sleep.

The LCD_taskFxn() runs through only one of these states anytime the Bluetooth low energy or master task must display some message to the screen. Then, LCD_taskFxn() updates the lock state variable depending on information passed from the other tasks (setting pins P2.1 and P2.2 on the RGB LED high). At that point, LCD_taskFxn() clears and re-enables interrupts and posts a semaphore to return control to the caller task. This action enables synchronous execution which mitigates communications issues with lost traffic from the CapTIvate HMI or Bluetooth low energy interfaces.

3.2.3.3 Touch Sensing Keypad

The state task of the MSP432, captivate_taskFxn(), which decides when to lock and unlock the system, receives the CapTIvate touch sensing peripheral data through the I^2C traffic from the MSP430FR2633 MCU. The SimpleLink MSP432 MCU runs the main state machine of the task, which has four states:

- CAPT_INIT: waits for a CapTIvate Touch Sensing Peripheral activity interrupt. If there is an interrupt it goes to the CAPT_ACTIVE state, if not it goes to the CAPT_IDLE state.
- CAPT_IDLE: where the device turns off the LCD screen and sleeps until the IP Phone panel wakes up the device using an interrupt.
- CAPT_ACTIVE: where the device reads data relevant to the keypad and buttons and updates the touch flags for both keypad and button accordingly, then goes to the CAPT_PASSCODE_ENTRY state if there is keypad input or else returns to the CAPT_IDLE state.
- CAPT_PASSCODE_ENTRY: where the device reads in the passcode from the passcode buffer, and returns to the CAPT_IDLE state until four numbers are entered. When a full code is entered, the device checks the entire code against the correct passcode. If there is a mismatch, the device erases the passcode off the screen using DRAW_ERASE_PASSCODE from the display task, clears out the buffer, prints the INVALID_PASSCODE message, and then draws the last valid state of the device using DRAW_SAVED_STATE from the display task state diagram. If the code matches the passcode in memory, then the device clears out the buffer, updates the lock state variable, signals to the display task to draw the unlocked lock based upon the lock state variable, and then erases the passcode with DRAW_ERASE_PASSCODE from the display task state machine. At that point, captivate_taskFxn() returns to the CAPT_IDLE state to await further input from the IP Phone panel.

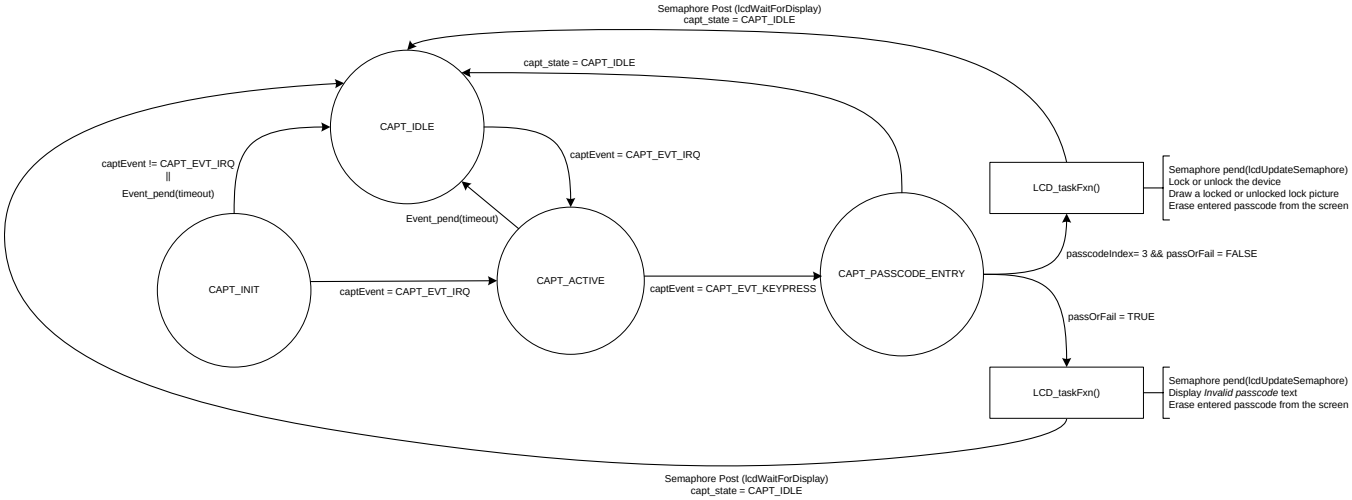


Figure 35. captivate_taskFxn() State Machine

In Figure 35, states that are inside the LCD_taskFxn() state machine are represented by octagons, whereas states inside captivate_taskFxn() are represented by circles.

3.2.3.4 Power Saving Functionality

The system time-out and shutoff task, DEMO_TIMEOUT_taskFxn(), manages the time-out period of the access panel. The task takes the role of a system shutoff timer that automatically counts to a certain period, elapses, and then turns off the LCD screen and haptics, and modifies the LPM4 transition constraint to conserve power. Furthermore, DEMO_TIMEOUT_taskFxn() sets the task priorities of the other three tasks (captivate_taskFxn(), LCD_taskFxn(), and AP_taskFxn()). This setting allows the tasks to run their initialization functions in a deterministic manner: LCD_taskFxn(), AP_taskFxn(), and captivate_taskFxn() initializing one after the other.

The timer implemented by the demo task is only reset every time a function or another task posts to demoTimeOutSemaphore. The timer is started whenever the software posts to demoStartTimeOutSemaphore, after the initial time-out and screen shutoff, as the startCounter Boolean flag is initialized to true.

DEMO_TIMEOUT_taskFxn() does not have a state machine. Figure 36 shows a flow diagram.

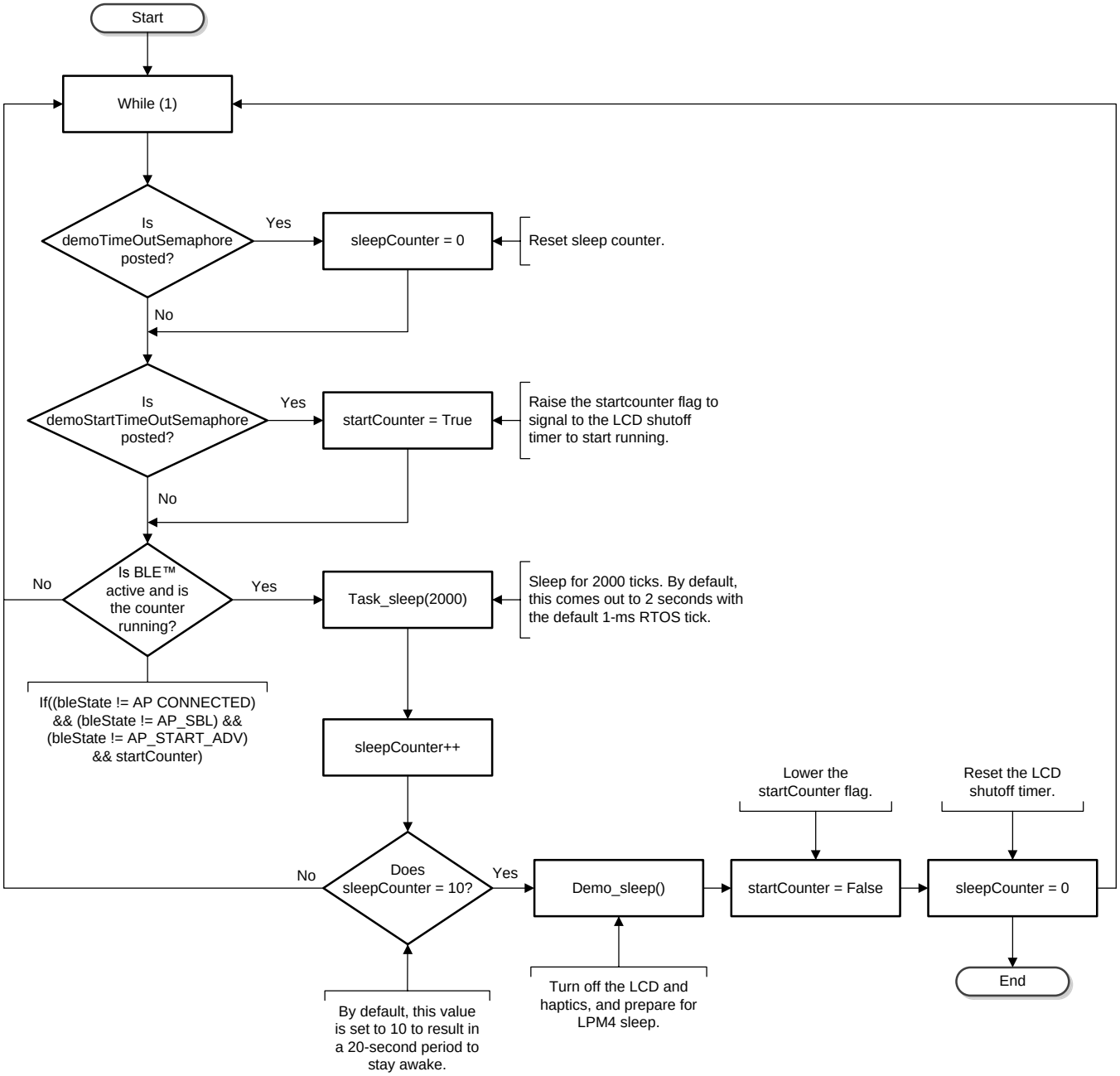


Figure 36. DEMO_TIMEOUT_taskFxn() Execution Flow Diagram

The task starts out by initializing a counter variable, sleepCounter, to 0, and a Boolean flag, startCounter, to true. From there, the demo task starts to execute. The task first checks for the demoTimeOutSemaphore but does not wait on the semaphore to signal execution. If demoTimeOutSemaphore is posted from somewhere else in the software, then the demo task resets the counter variable, sleepCounter, to 0. The demo task then goes to the next if() statement and checks for the demoStartTimeOutSemaphore without waiting on it to be posted. If demoStartTimeOutSemaphore is posted, it then sets the flag startCounter to true. After checking for those two semaphores, the demo task then sleeps for 2000 ticks.

After waking up, the task checks to see if the startCounter flag is true, and if the Bluetooth low energy task, AP_taskFxn(), is in the AP_START_ADV state (the CC2650 is advertising), the AP_CONNECTED state (an active Bluetooth low energy connection exists), or the AP_SBL state (the MSP432 MCU is reprogramming the CC2650 MCU with the SNP hex image). If all conditions are true, then the task increments startCounter. If the counter passes a threshold, the task turns off the whole system using Demo_sleep(), lowers the startCounter flag, and resets startCounter.

To change the timeout period, adjust the *number of ticks* argument in the Task_sleep() call, at line 165 in demo_task.c. This argument represents the number of ticks that the task sleeps for, with each tick being configured to represent 1 ms in time. Then, adjust the sleepCounter threshold in if(sleepCounter > 15) at line 173. The total period that the system waits until it shuts down the task is represented by Equation 1.

$$\text{Total time (seconds)} = \text{sleepCounterthreshold} \times \text{ticks} / 1000 \quad (1)$$

Several functions to note follow:

- DEMO_prepareForDeepSleep(): This function turns off the real time clock (RTC), releases the LPM4 transition constraint, and sets the *deepSleep* Boolean flag.

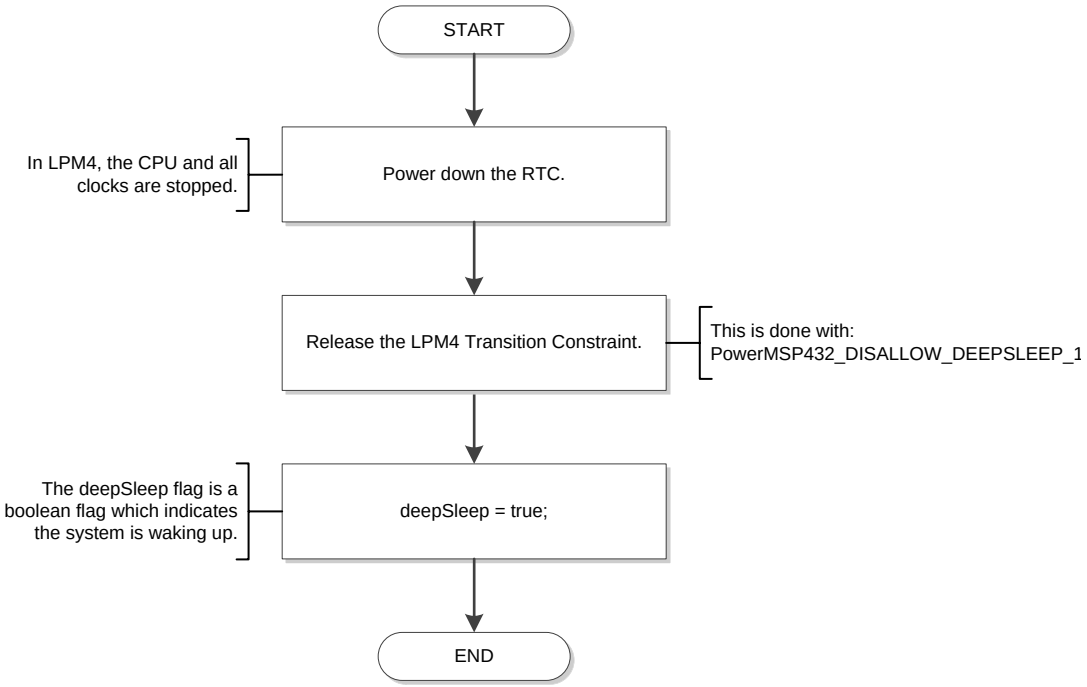


Figure 37. Demo_prepareForDeepSleep() Flow Diagram

- DEMO_comeOutFromDeepSleep(): This function checks for the deepSleep flag. If the flag is raised, the function first lowers the flag, then it re-enables the LPM4 transition constraint, and restarts the RTC.

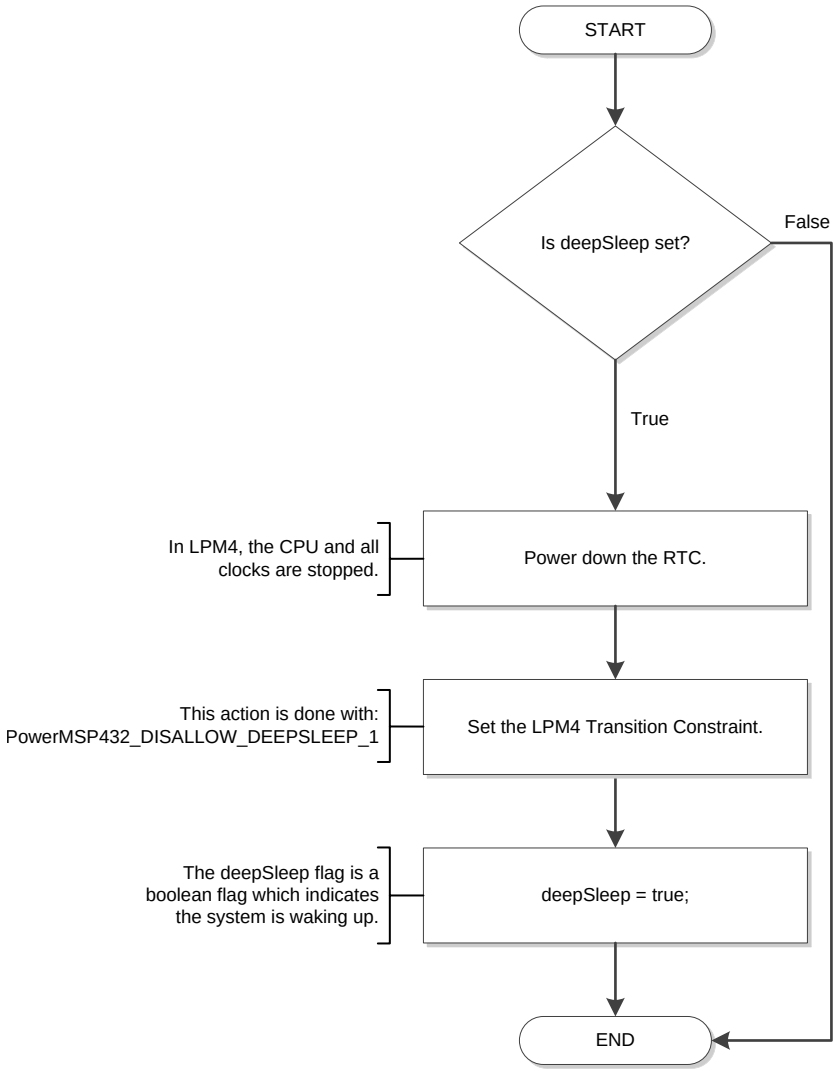


Figure 38. Demo_comeOutFromDeepSleep() Flow Diagram

- Demo_sleep(): This function turns off the backlight and the LCD, puts the DRV26x device into standby using the DRV26x API, and then calls DEMO_prepareForDeepSleep().

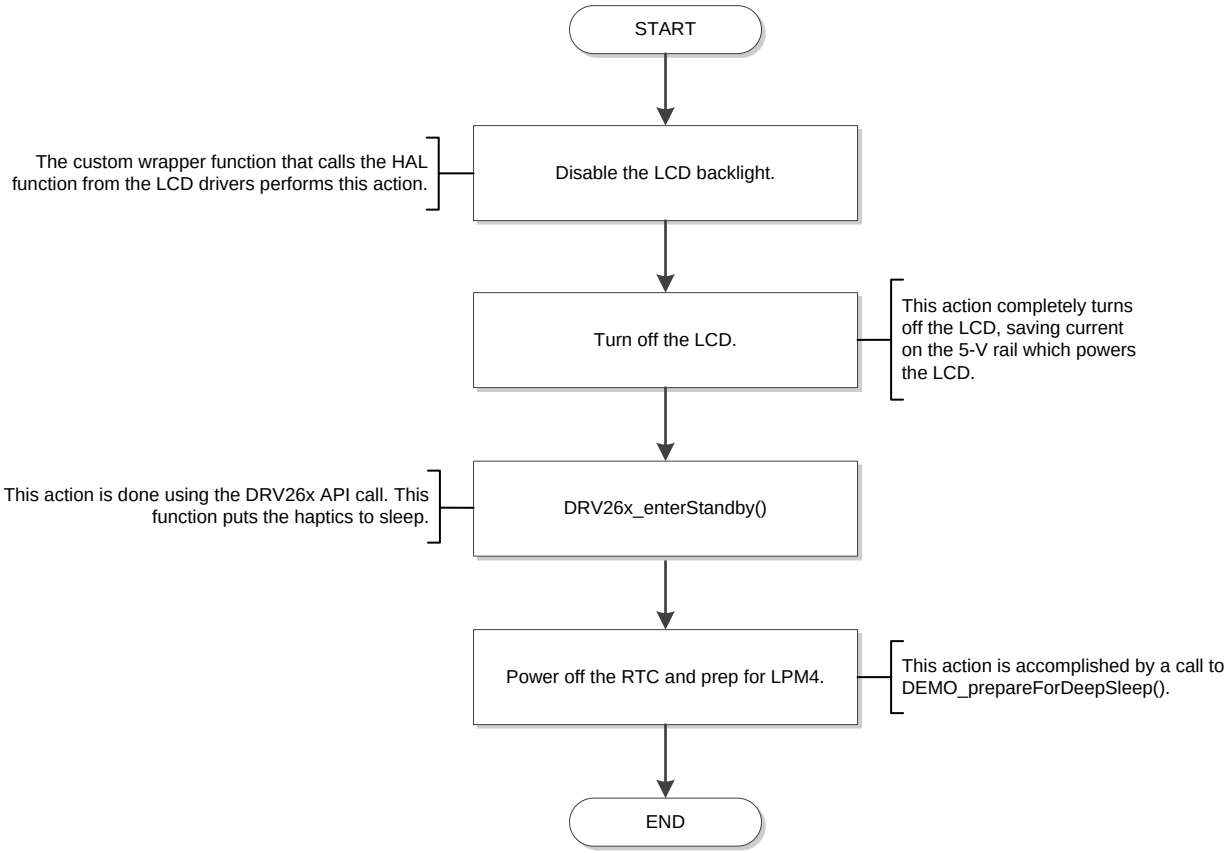


Figure 39. Demo_sleep() Flow Diagram

- Demo_awake(): This function executes in the opposite direction of Demo_sleep(), waking up the DRV26x device and the LCD, then re-drawing the menu, and finally turning the LCD backlight back on.

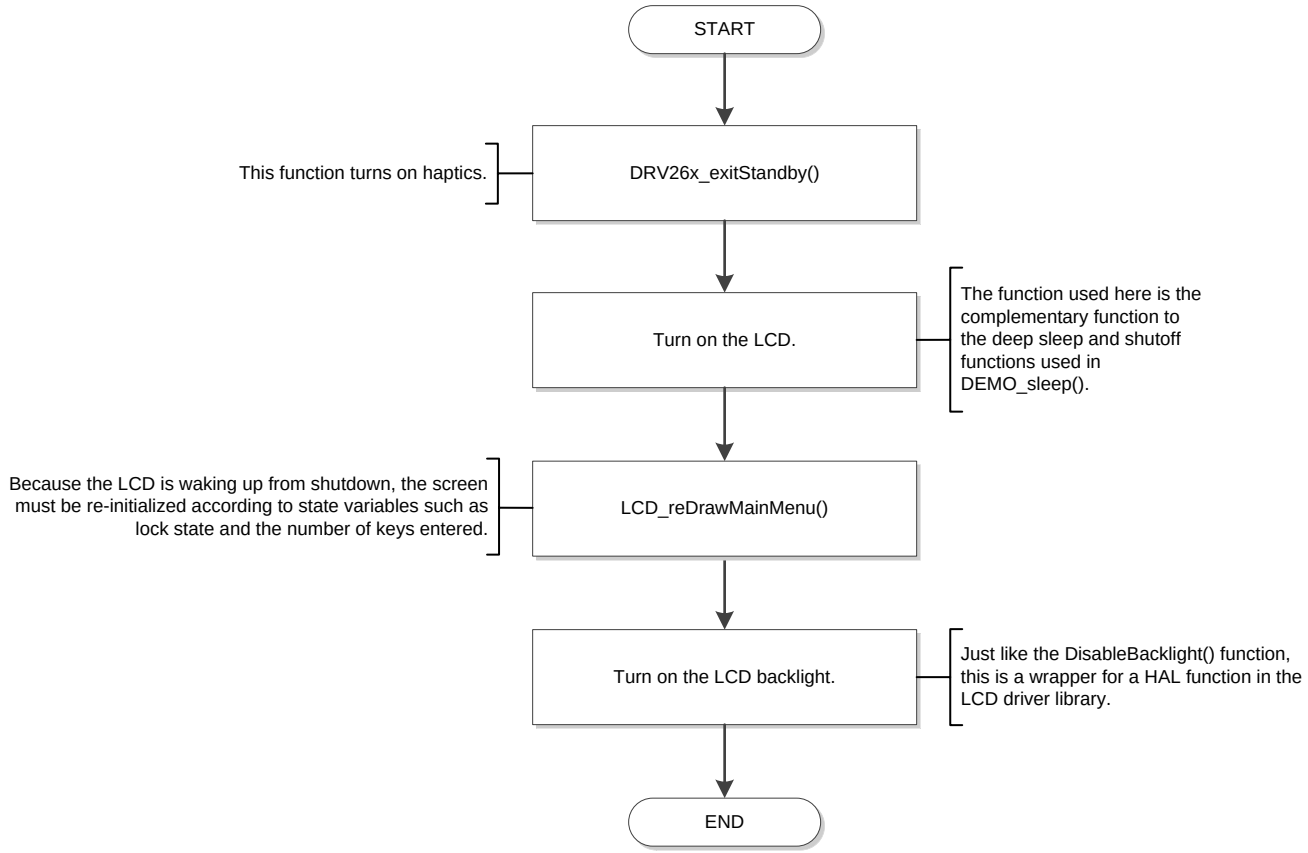


Figure 40. Demo_awake() Flow Diagram

- DEMO_RTC_C_IrqHandler(): This is the ISR that reads the RTC and manually triggers a RTOS tick every time the RTC rolls over. This ISR enables TI-RTOS to run off of a low-power clock. See the power saving section for more details on this ISR and the role of the RTC in the system.

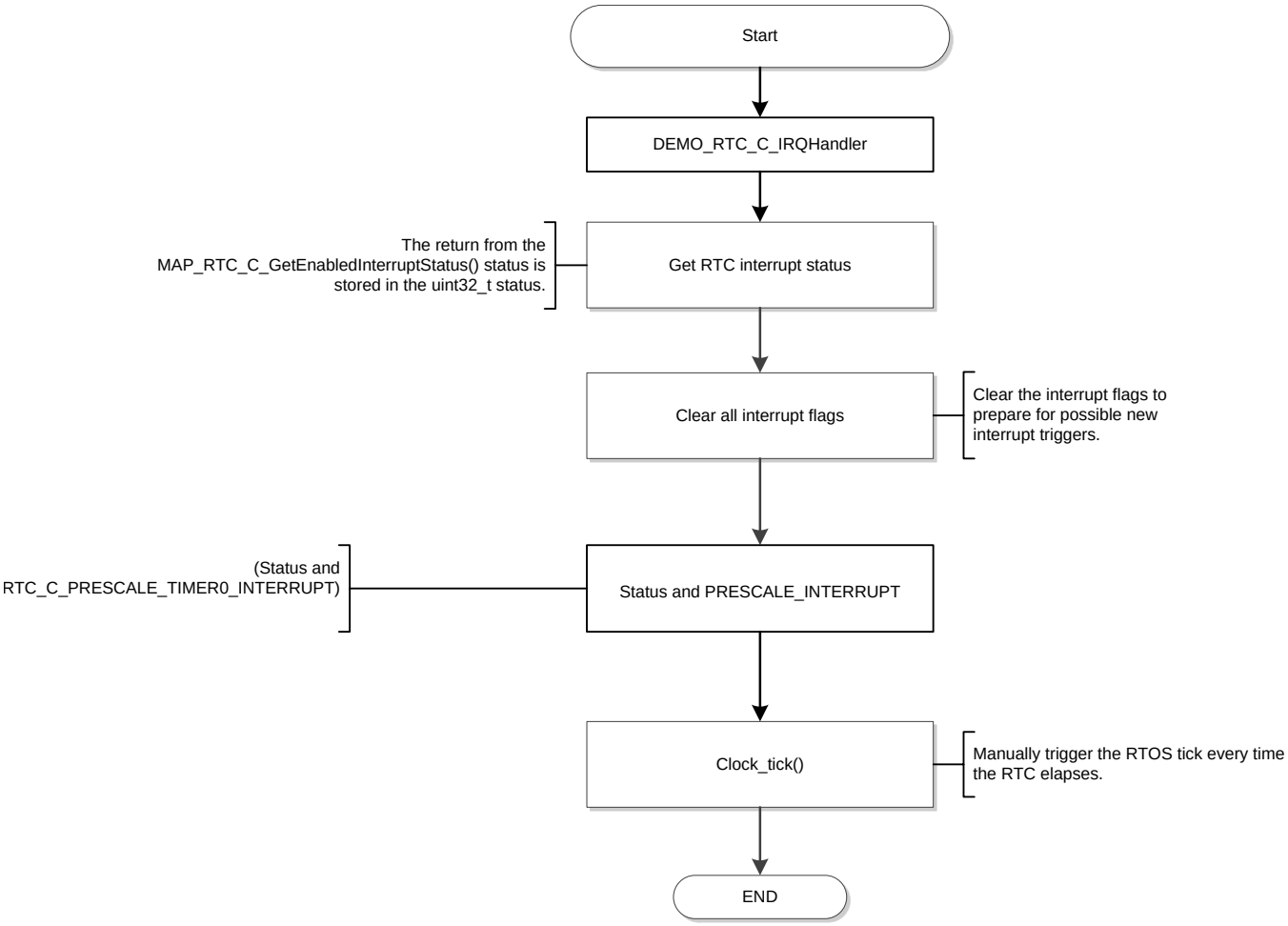


Figure 41. Demo_RTC_C_IrqHandler Flow Diagram

Demo_prepareForDeepSleep() and Demo_sleep() are called within DEMO_TIMEOUT_taskFxn(), but the wake-up functions, Demo_comeOutFromDeepSleep() and Demo_sleep(), are called from captivate_taskFxn() and AP_taskFxn(), to ensure that the system wakes up from LPM4 properly, to either advertise to a mobile device through Bluetooth or to respond to touch panel activity. For details on AP_taskFxn(), see Section 3.2.3.1. For details on captivate_taskFxn(), see Section 3.2.3.3.

3.2.4 MSP430™ Software: MSP430FR2633 MCU Interface To MSP432 bMCU

The MSP430 MCU handles the capacitive sensing and processes all raw CapTIvate data and flags to determine if there has been a valid touch or close-proximity gesture, explained in the software diagram which follows. The MSP430FR2633 MCU then sends a buffer of touch data to the SimpleLink MSP432 MCU through I^2C, which indicates which sensor reported activity and which button or key was pressed. The MPS432 MCU then acts on that information appropriately.

The MSP430FR2633 software is implemented on top of an automatically-generated software package through the CapTIvate Design Center. From there, the software interface to the MSP432 MCU is implemented by several callback functions that trigger whenever the appropriate sensor detects touch or proximity activity. These functions share the same buffer, uint8_t txBuffer[8], where the buffer is simply an array of 8 integers. These callback functions check to see what activity has occurred on the touch panel and then populate txBuffer[] accordingly (see Figure 42).

txBuffer[] Message Format

Index	0	1	2	3	4	5	6	7
Description	Keypad sensor activity flag	Dominant key pressed (represents the key number)	Keypad keypress and no keypress flag	Proximity sensor activity flag	Proximity sensor touch and no touch flag	Proximity sensor proximity and no proximity flag	Wheel button activity flag	Wheel button touch and no touch flag
Value	[0, 1]	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 1]

Figure 42. txBuffer

This interface can be quickly customized by changing the code within the callback functions in the KeypadDemo.c file. If a new callback function is added with a new touch panel sensor, the function must be registered as a callback function within the InitDemo() function in KeypadDemo.c. The user will have to write a custom callback function for any new touch sensor functionality they wish to generate.

The three sensors follow:

- numericKeypadSensor: represents the keypad
- proxAndGuardSensor: represents the proximity sensor
- wheelButtonSensor: represents the wheel button on the phone panel at the right. This button is used to lock the device from the touch panel.

There are three callback functions: touchDetect() for the keypad, proximityDetect() for proximity sensing, and buttonDetect() for the lock button. They are all registered to the appropriate sensor using the callback function MAP_CAPT_registerCallback(&function_name,&sensor_name) at the bottom of KeypadDemo.c starting at line 266, within the function InitDemo(). Each function uses the CapTIvate flags, with preprocessor definitions to simplify software development, to determine if there has been a touch. While each function has different conditions and implications for touch sensing, they share elements of the same execution flow.

The first callback function, touchDetect(), checks to see if there has been a new touch. If there has been a new touch, touchDetect() grabs the index of the pressed key by grabbing the *ui16DominantElement* member inside the pSensor->pSensorParams object in memory. Then, touchDetect() populates the first three integers inside txBuffer[] according to the previous interface and triggers the IRQ line by calling I2CSlave_setRequestFlag(). If there is just a touch, touchDetect() does not do anything. If the touch event ends, represented by the current touch flag lowered and the previous touch flag raised, defined as EXIT_TOUCH, then the MSP430 MCU populates txBuffer[] according to the interface, but with a 0 in the third array index to indicate that the key is no longer pressed down. After populating the buffer, the IRQ line is left alone.

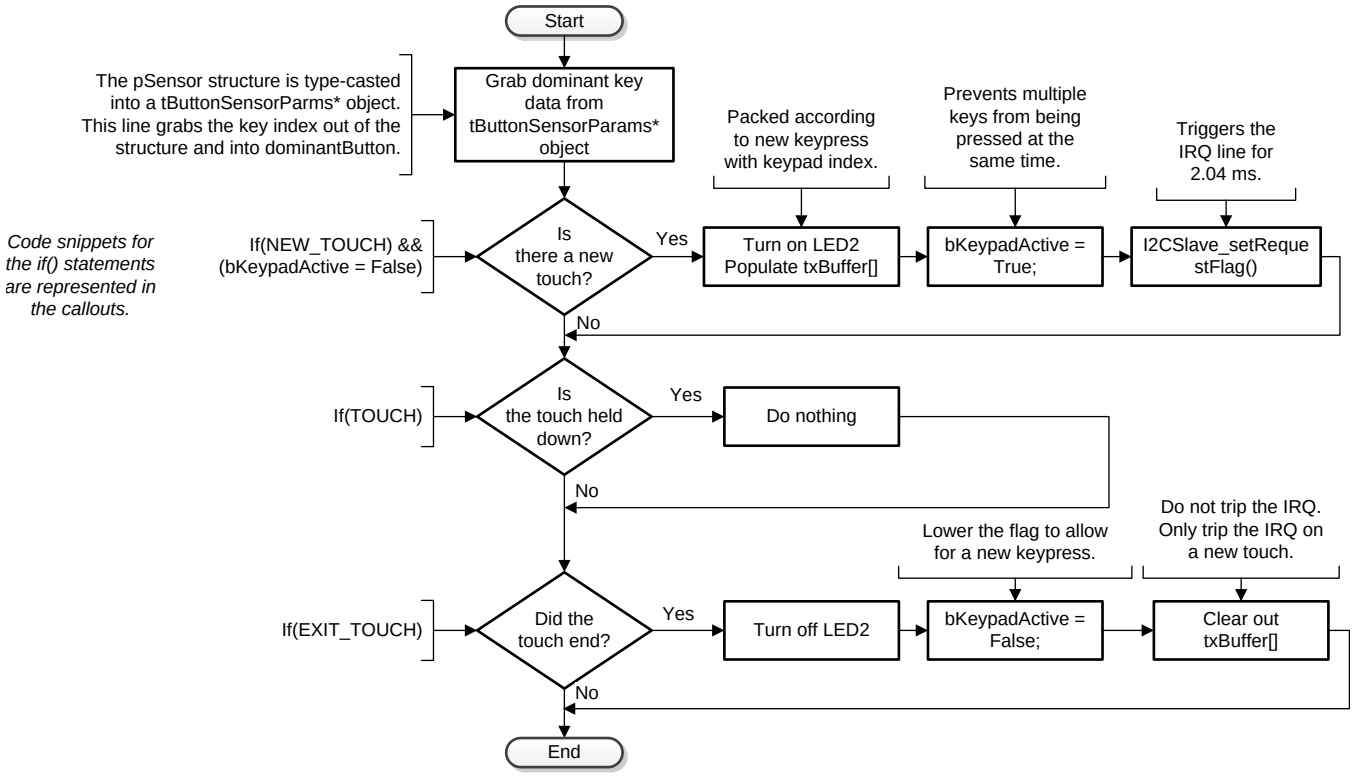


Figure 43. Keypad Callback Execution Flow

The second callback function, proxDetect(), works in a similar fashion, but in conjunction with a Boolean flag, bProximity, to indicate if there has been a previous proximity detection. If there has been a proximity detection and bProximity is false, the MSP430 MCU populates txBuffer[3] and txBuffer[5] with 1 and 1, to indicate that there has been proximity sensor activity and that a body has been sensed in close proximity to the panel. The IRQ line is also triggered with I2CSlave_setRequestFlag(). Checking bProximity helps to determine whether there has previously been a proximity event, so that the proximity sensor only triggers on the first sensing. If there is no proximity detection and bProximity is true, then the bProximity flag is lowered. After both if() conditions, the MSP430 MCU populates txBuffer[3] and txBuffer[5] with 0s but it does not raise the IRQ line.

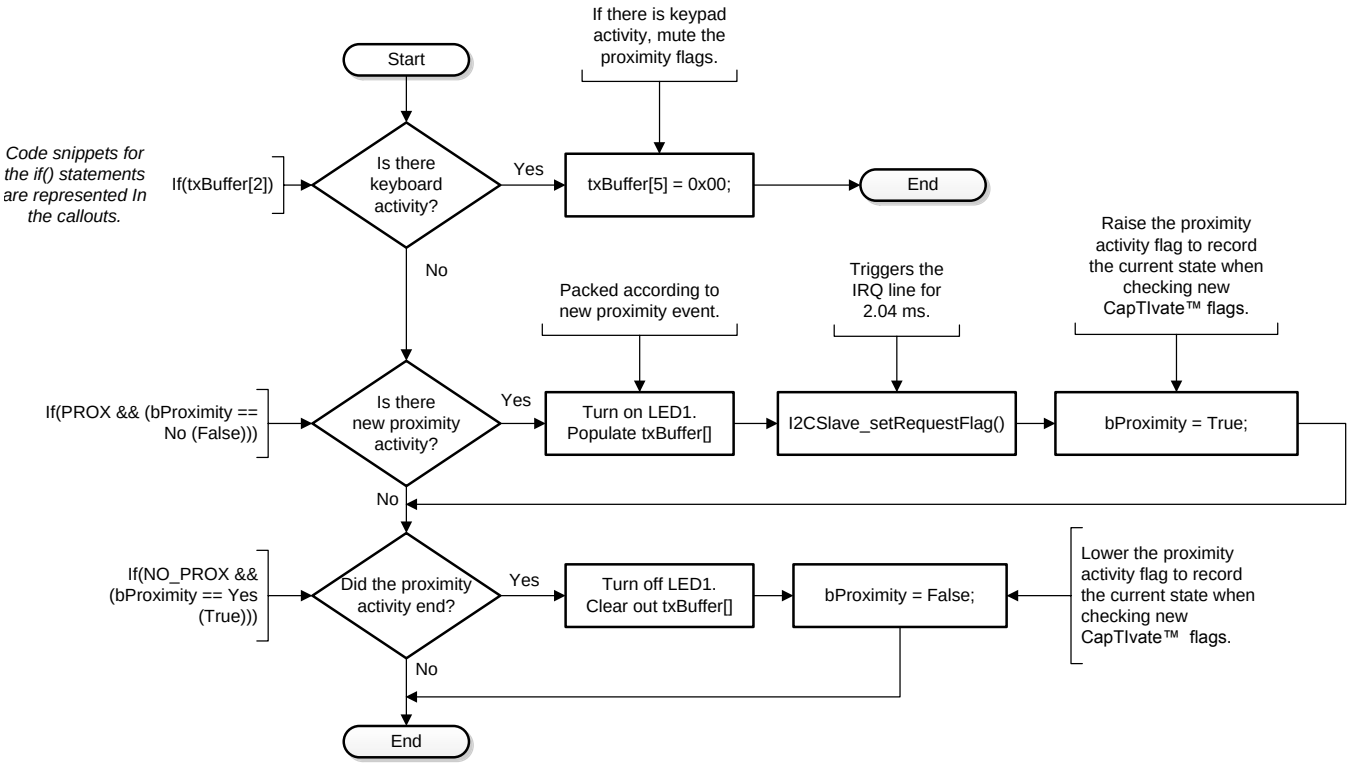


Figure 44. Proximity Sensor Callback Execution Flow

The third callback function, buttonDetect(), works almost identically to touchDetect(), without having to grab raw CapTIvate data from the tButtonSensorParams data structures. The buttonDetect() function checks to see if there has been a new touch. If there has been a new touch, buttonDetect() populates the first two integers inside txBuffer[], txBuffer[6], and txBuffer[7], according to the previous interface, and then triggers the IRQ line by calling I2CSlave_setRequestFlag(). If there is just a touch, buttonDetect() does not do anything. If the touch event ends, represented by the current touch flag being lowered and the previous touch flag raised, defined as EXIT_TOUCH, then the MSP430 MCU populates txBuffer[] according to the interface, but with 0s to indicate that the button is no longer pressed down. After populating the buffer, the IRQ line is left alone.

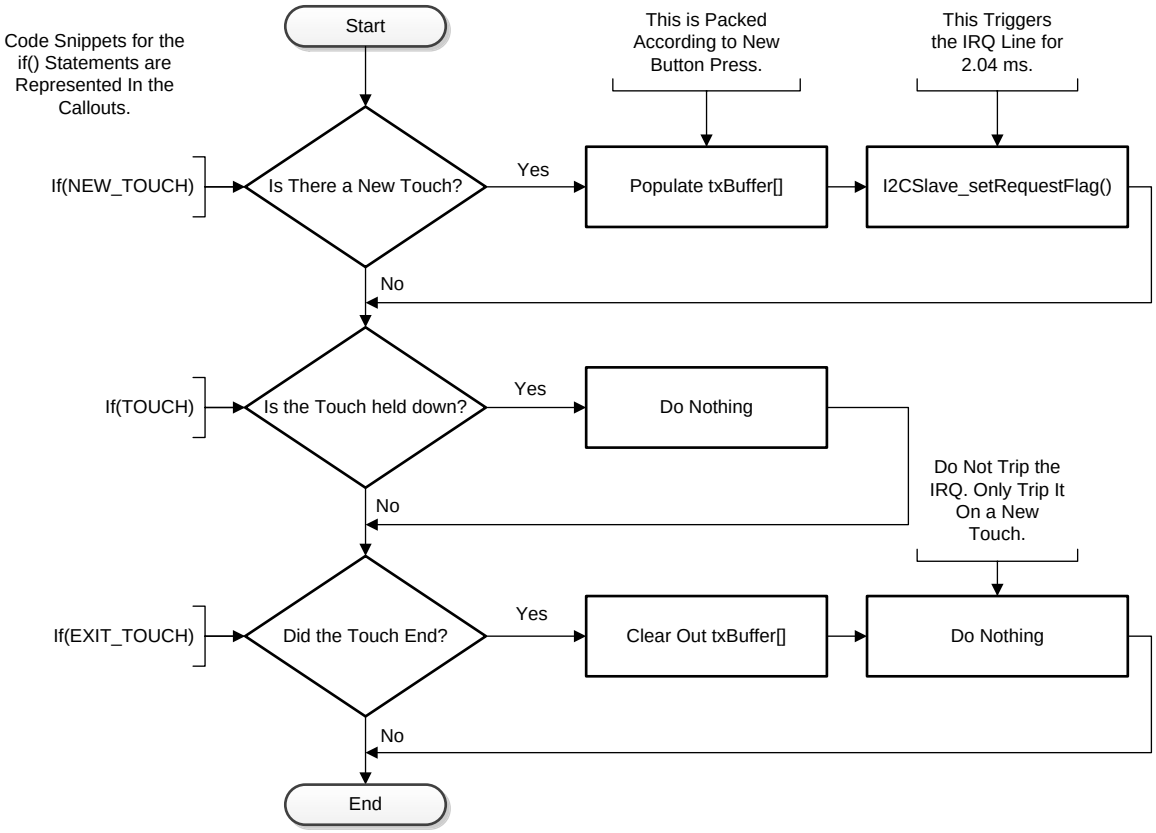


Figure 45. Button Callback Execution Flow

In the main program, the software executes in a standard infinite loop. If there is any touch activity, LED2 illuminates. If there is any proximity activity, LED1 illuminates. Afterwards, the device sleeps until woken up again through the proximity sensor named proxAndGuardSensor.

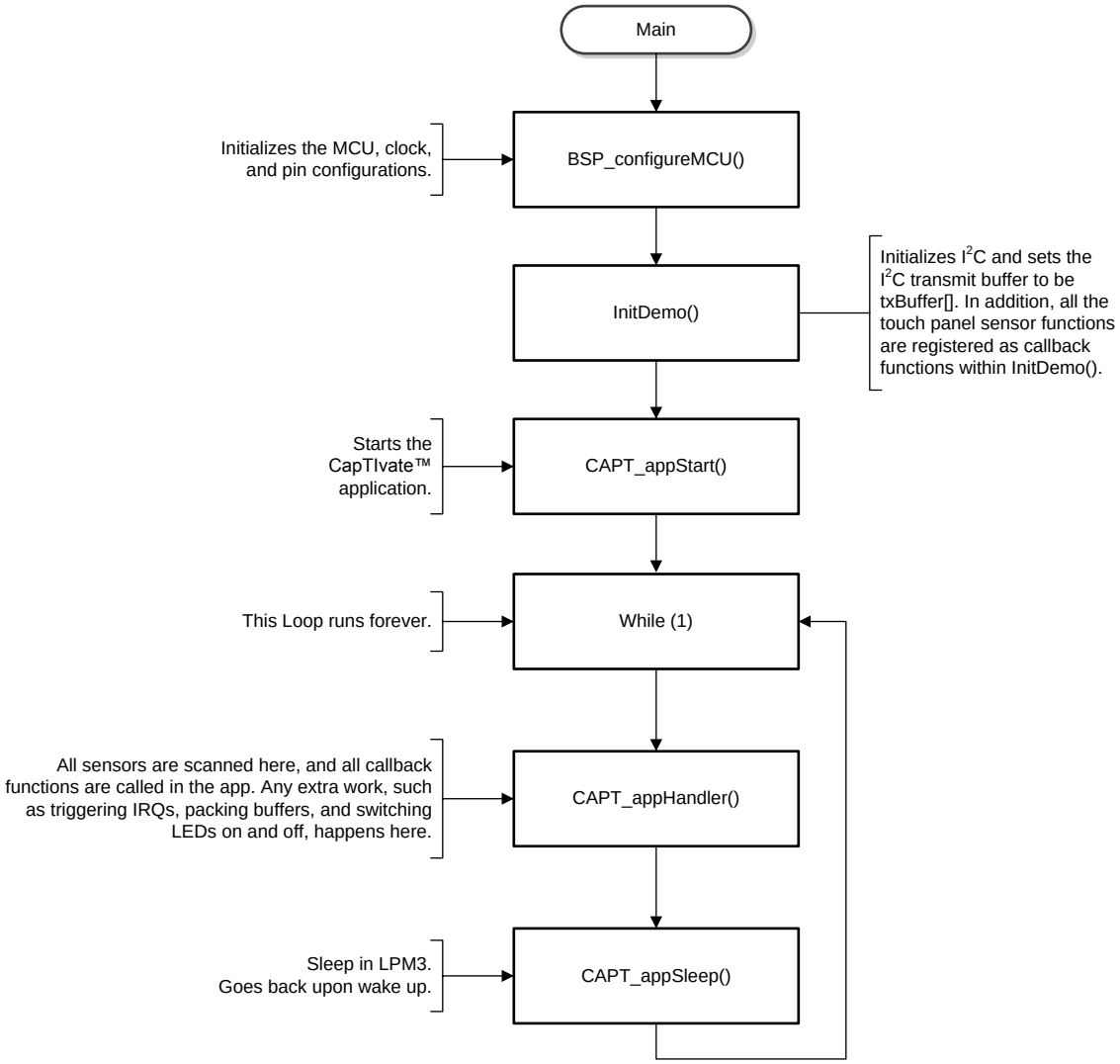


Figure 46. blelock_capTIvate Main Flow Diagram

4 Testing and Results

4.1 Power Consumption and Savings

Power consumption is a critical factor in the intended end equipment for this reference guide. Many applications for this access panel are powered by a mobile power source, usually represented as either a lithium polymer battery or a coin cell battery. This section elaborates on several power-optimization choices and the power savings achieved with those design choices.

Overall power consumption however, is detailed by three use cases: sleeping, active with keypad input, and active with Bluetooth low energy input. The system consumes an average of 18 µA while in sleep mode. This is the first system-level use case. The system consumes an average of 3 mA while actively processing keypad input. Lastly, the system consumes an average of 4 mA while processing Bluetooth low energy input. All measurements are taken with the MSP-EXP432P401R LED and XDS110 jumpers removed.

4.1.1 MSP432P401R MCU Power Consumption

Figure 47 shows the power consumption of the MSP432 MCU with sleep modes enabled to LPM4. Here, both the CC2650 MCU and the MSP430FR2633 MCU are also asleep in their deepest mode.

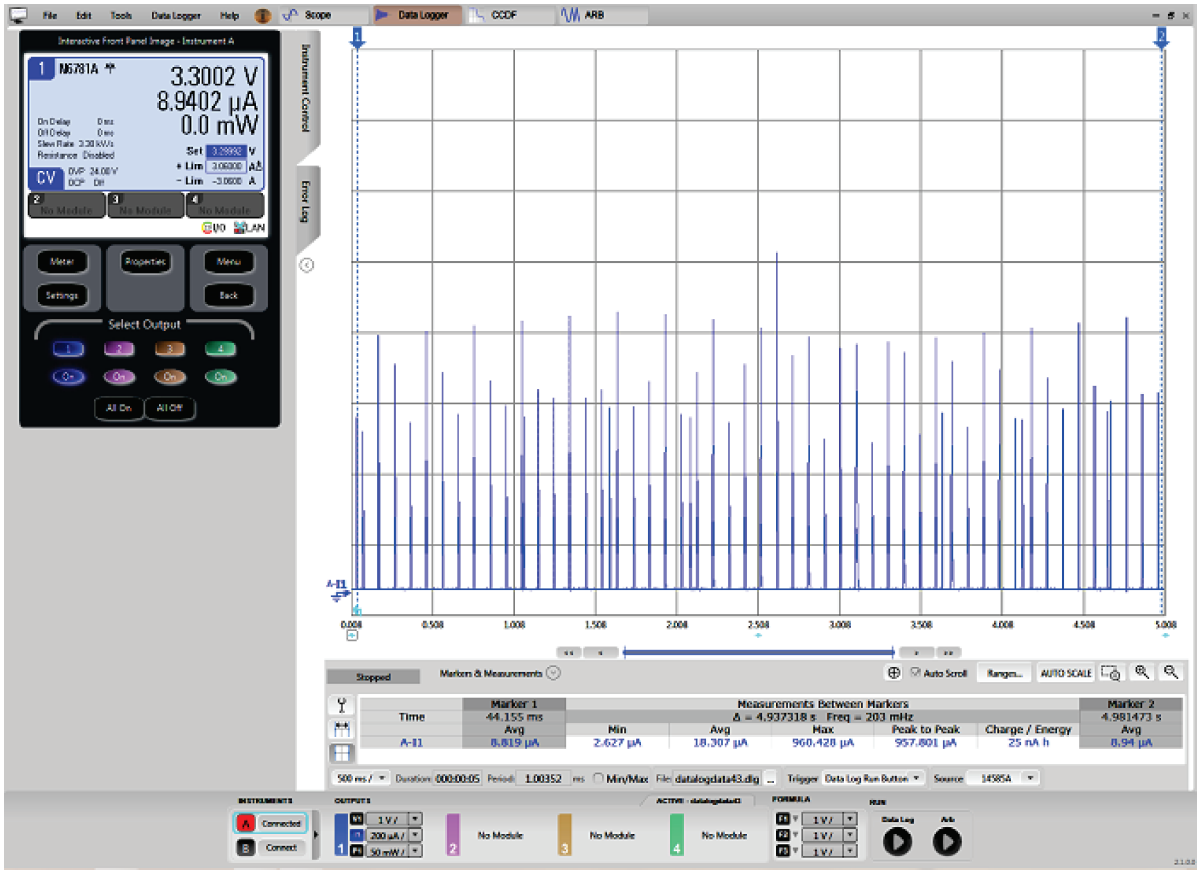


Figure 47. TIDM-1004 Global Sleep Current Consumption

The two use cases for the MSP432 MCU are the device running through the captivate_taskFxn() task, sleeping while idling with the LCD powered off for 5 seconds, and then waking up to process input. The first use case locks using touchpad input, then unlocks using the keypad. The second use case locks and unlocks using Bluetooth low energy. Both use cases then end with an approximately 5-second rest, in which the device goes back to sleep.

To get to LPM3, this design sources the RTOS tick source from the real-time clock (RTC). This results in a standby current consumption of 0.1 mA. This TI Design does not use the LPMx.5 modes because of clocking requirements for TI-RTOS. Dipping lower than LPM3 would necessitate stopping clock sources for the RTOS tick, which then completely halts program execution.

For further power savings, this system sleeps in LPM4. In LPM4, the CPU and all timers shut down, including the RTC. To prepare for this, the system must set and release constraints for LPM4 transitions, and also ensure that the RTC is shut down correctly so that TI-RTOS does not crash from the absence of a tick rate. However, to wake up from LPM4, any task that responds to an interrupt must call the wakeup functions in the demo_task.c file, DEMO_comeOutOfDeepSleep() and Demo_awake(), in that order. See Section 3.2.3.4 on DEMO_TIMEOUT_taskFxn() for details on these two functions. See Section 3.2.3.3 and Section 3.2.3.2 on captivate_taskFxn() and AP_taskFxn() for details on the transition of those task from low power states.

Using LPM4 enables idle current consumption under 1 µA for the MSP432, which further cuts down on idle current draw. This feature extends battery life consumption on the system.

In the final software, parameters were updated to decrease the wake-on-proximity sampling rate, to improve battery life. Final global current consumption dropped to around 10 mA while the MSP432 MCU was actively processing data from the touch panel, as shown in Figure 48, and idle current consumption fell to 18 µA, providing an average current consumption of approximately 3 mA. For processing Bluetooth input, the average current consumption was 3.9 mA, while the idle current consumption was 9 µA, as shown in Figure 49.

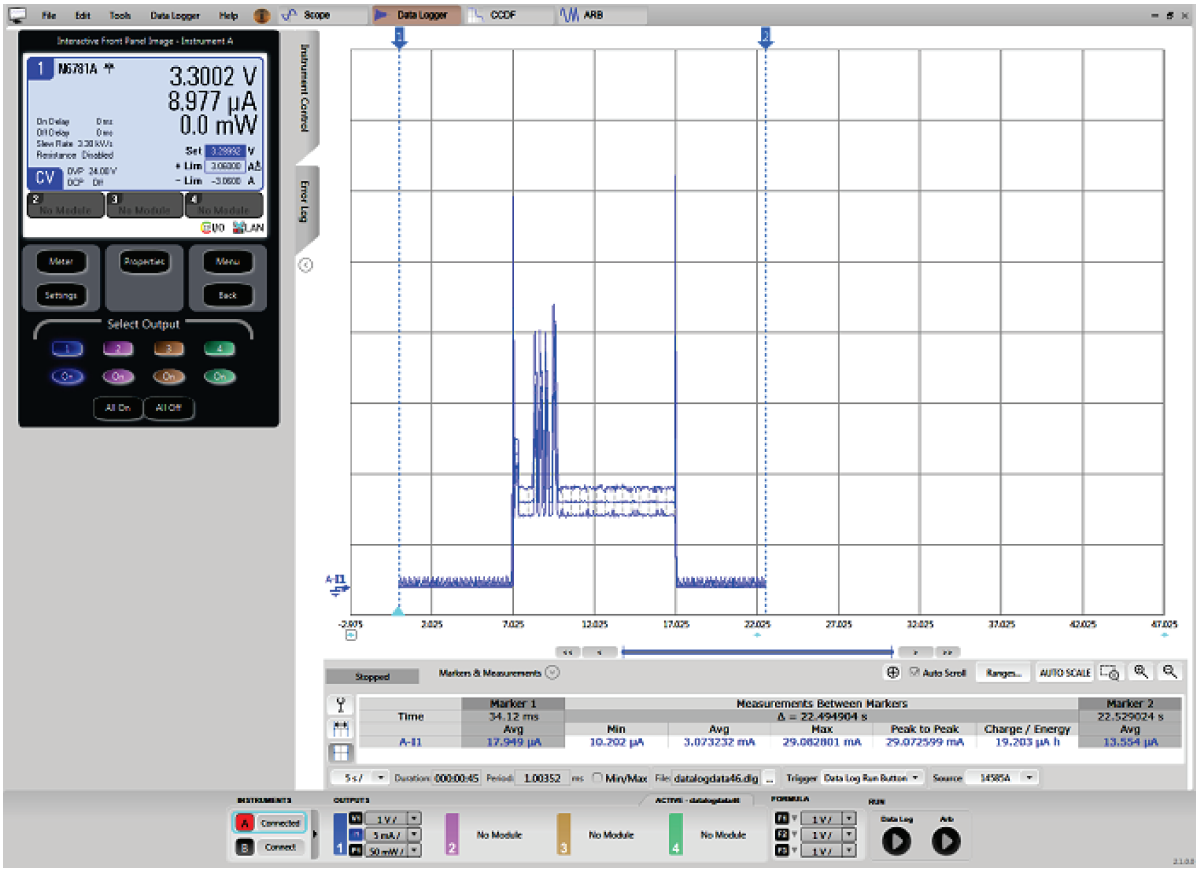


Figure 48. CapTIvate Use Case Current Consumption Curve

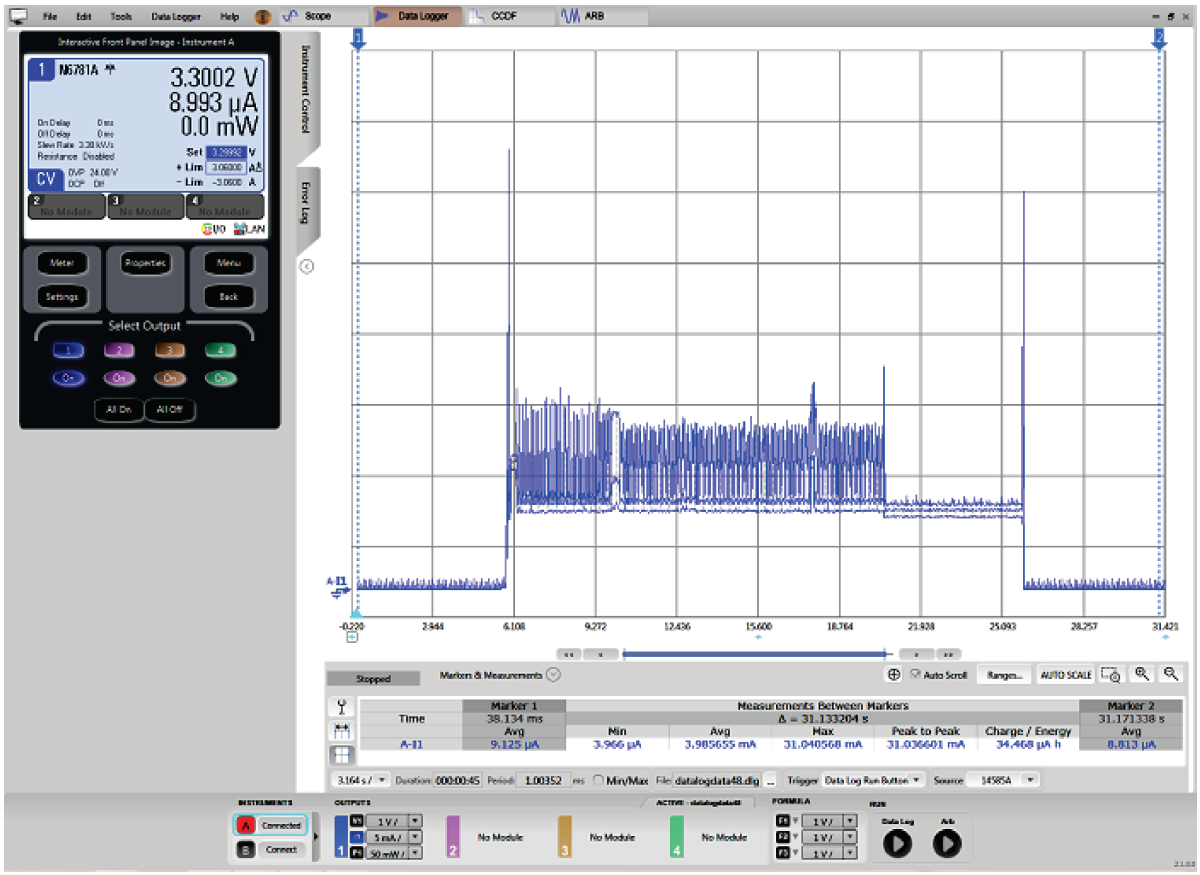


Figure 49. Bluetooth Low Energy Use Case Current Consumption Curve on MSP432 MCU

4.1.2 MSP430FR2633 MCU Power Consumption

The MSP430FR2633 MCU driving the CapTIvate touch panel achieves its power savings by using the existing CapTIvate software architecture. The following two measures were taken:

- The wake-on-proximity functionality is activated with the sampled sensor set to the proximity sensor. This action is done at the top of CAPTUserConfig.h, by modifying the preprocessor macros CAPT_WAKEONPROX_ENABLE and CAPT_WAKEONPROX_SENSOR, but it can also be done using the CapTIvate design center, by clicking on the symbol of the MCU in the GUI.
- The low-power mode is set to LPM3. This action is done by changing the struct member *.ui8AppLpm* in the tCaptivateApplication struct g_uiApp at the bottom of CAPTUserConfig.c. The reason LPM3 is used instead of a more extreme low-power mode is because to wake up from an external interrupt, the CapTIvate peripheral of the MSP430 must sample the sensors while the rest of the system is sleeping, using low-power sections of the architecture of the IP to continually scan for possible touchpad activity. Using a lower-power mode, such as LPM4, would shut off the clock sources and stop the system from reading the panel entirely, leaving it unable to wake from that potential interrupt source. To change the low-power mode, set *.ui8AppLpm* to either:
  - *LPM0_bits* for LPM0
  - *LPM1_bits* for LPM1
  - *LPM2_bits* for LPM2
  - *LPM3_bits* for LPM3
  - *LPM4_bits* for LPM4

These definitions are found in msp430fr2633.h within the MSP430 driver libraries (called driverlib).

These measures result in a sleep mode current draw of 9 µA, with a 100-ms sleep-mode sampling period, represented in the struct g_uiApp as *.ui16WakeOnProxModeScanPeriod*.

Figure 50 shows the current curve of the device when sleeping, waking up from the proximity sensor, reading button input, reading keypad input, and then going back to sleep.

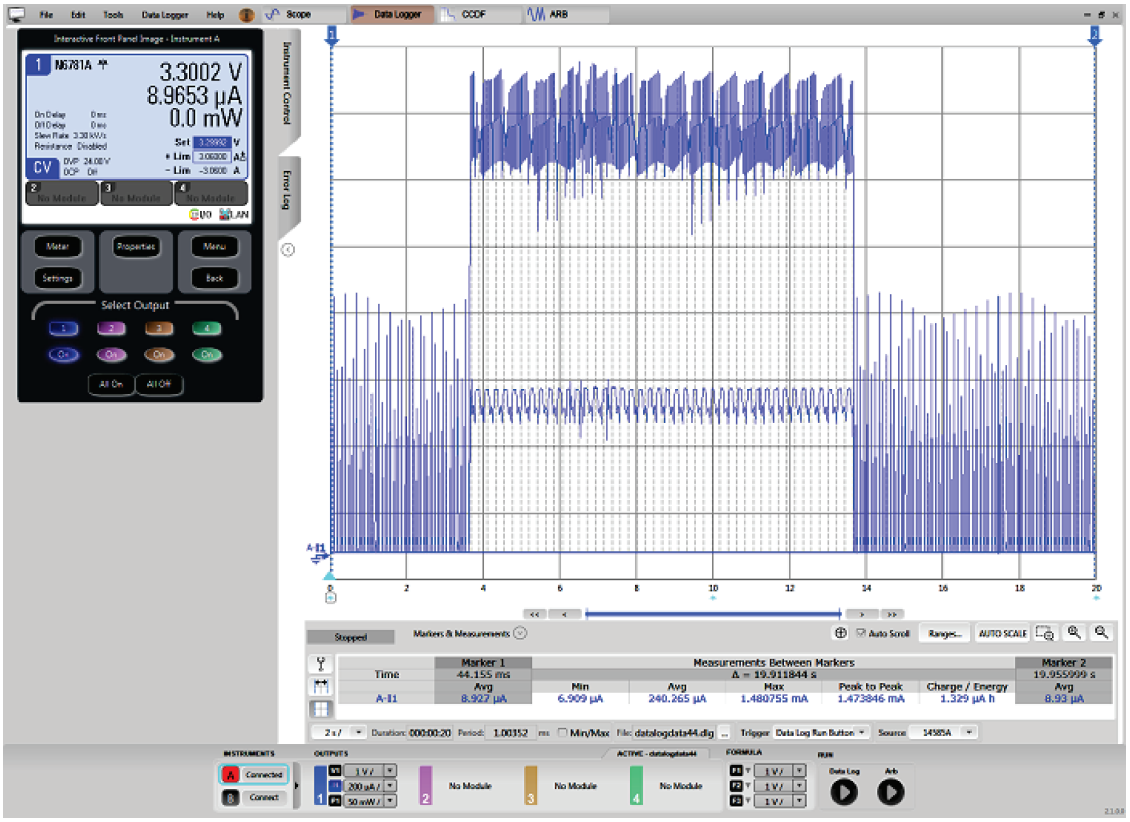


Figure 50. MSP430FR2633 MCU Current Consumption

4.1.3 CC2650 MCU Power Consumption

The CC2650 SNP runs a power-optimized executable from the Simple Access Point example project within the SimpleLink MSP432 SDK Bluetooth Plugin. Four states exist: standby, advertising, connected, and firmware re-flash. The current curve in Figure 51 shows the curve of the CC2650 MCU over this use case.

The first flat portion of Figure 51 shows the power consumption of the CC2650 SNP while sleeping. The second flat portion of Figure 51 shows the active power consumption of the hardware stack when the user is using the Bluetooth low energy access point. In this use case, only the MSP432 MCU and the CC2650 SNP are running. The MSP430FR2633 MCU remains in sleep mode. The third flat portion of Figure 51 shows the current consumption of the CC2650 MCU while communicating with the local device. The final section of the current curve in Figure 51 represents the current consumption of the CC2650 MCU while the MSP432 MCU reprograms the SNP.

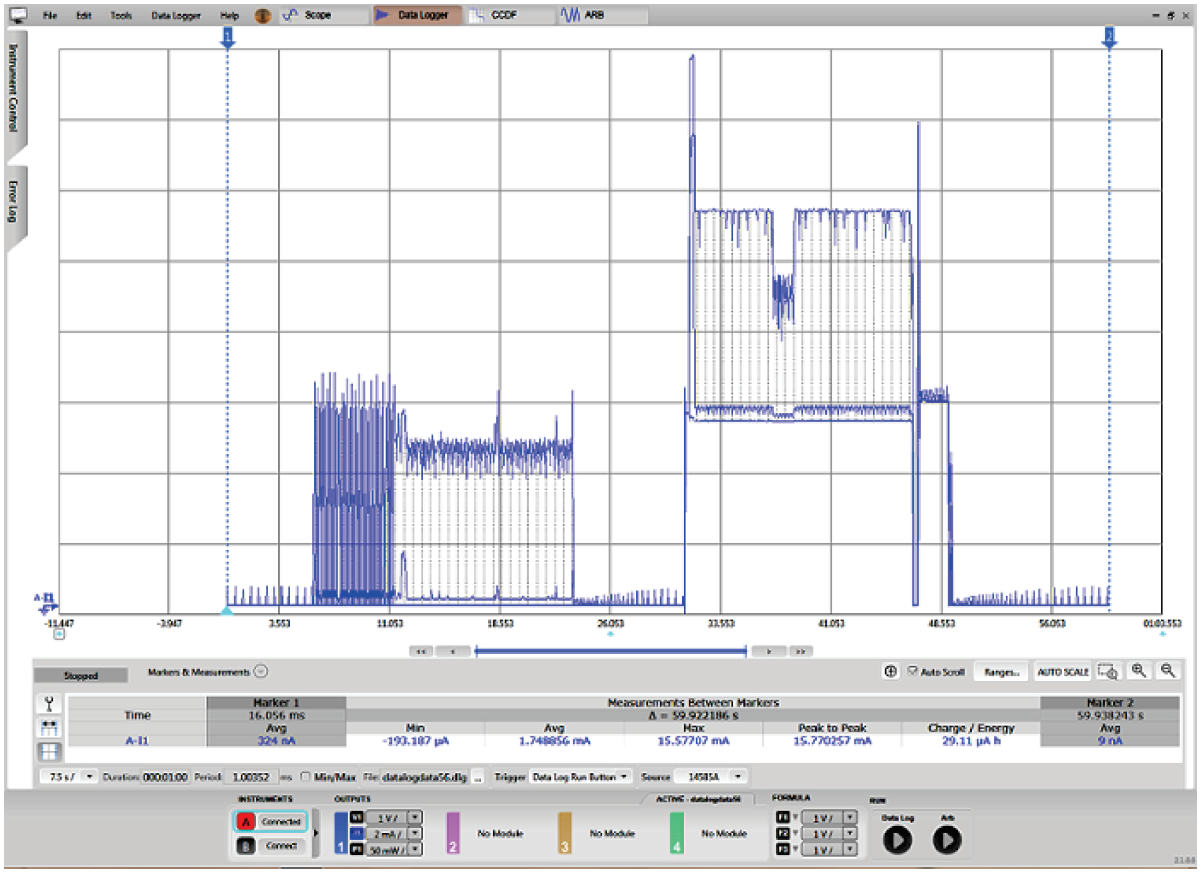


Figure 51. CC2650 MCU Current Curve

The final numbers for the CC2650 MCU are 211 µA for advertising current consumption, 200 µA for an active connection, 6 mA while reprogramming, and 1.3 µA for sleep mode.

4.1.4 Power Consumption From a Visual Glance

Figure 52 shows the individual current consumption per component under the sleeping case as well as the touch panel and Bluetooth use cases. Because the LCD is the largest current sink in the system, any general power-saving scheme must be able to power-gate the LCD. Fortunately, the hardware interface of the LCD provides the ability to do so. The MSP430FR2633 device and the CC2650 device are minor contributions to the total current draw under normal use cases; the CC2650 draws a large amount of current if it is being reprogrammed. See Figure 52 for a visual description of the individual current contribution.

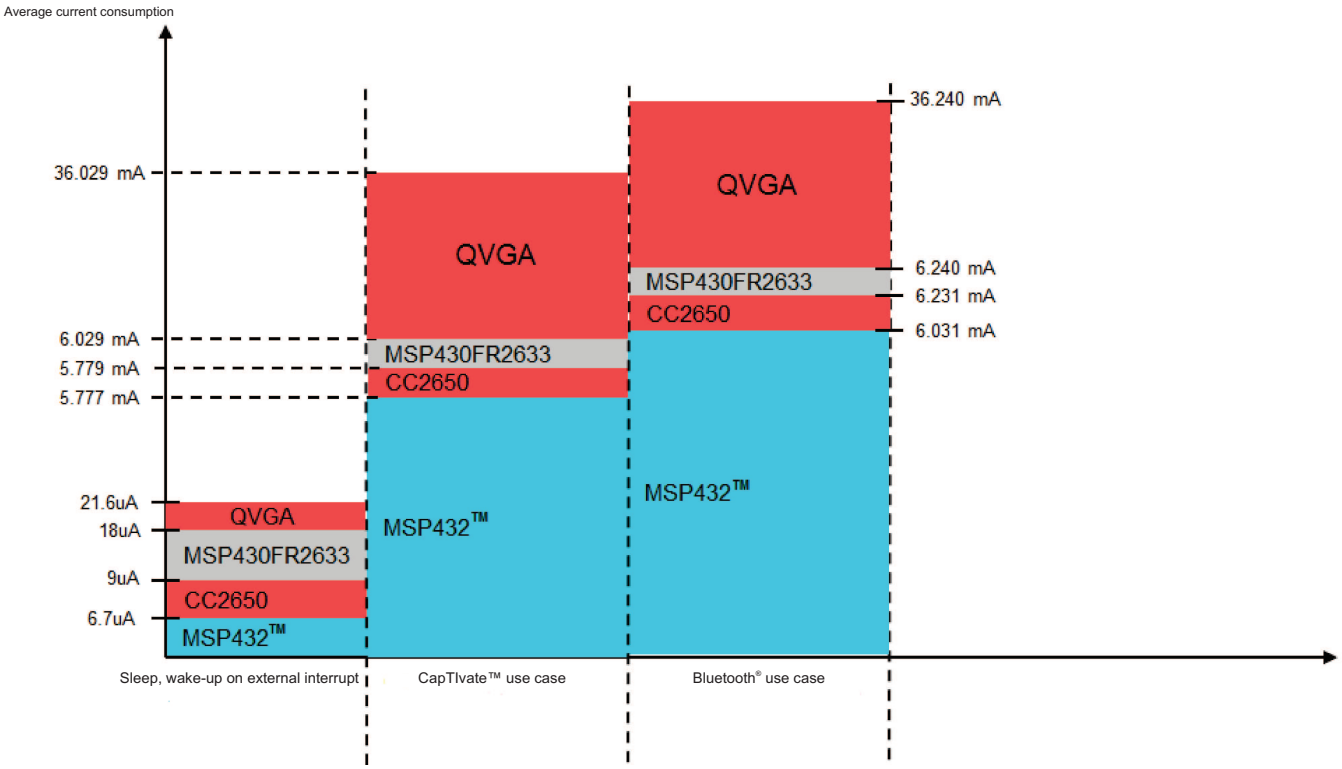


Figure 52. TIDM-1004 Power Consumption Breakdown

5 Design Files

This design guide describes only the touch panel design files. To download files relating to other parts of the system, go to:

- MSP-EXP432P401R
- Kentec QVGA BoosterPack
- CapTIvate Development Kit
- CC2650 Bluetooth low energy BoosterPack

5.1 Schematics

To download the schematics, see the design files at TIDM-1004.

5.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDM-1004.

5.3 PCB Layout Recommendations

See the recommendations outlined in the CapTIvate Technology Guide.

5.3.1 Layout Prints

To download the layer plots, see the design files at TIDM-1004.

5.4 Altium Project

To download the Altium project files, see the design files at TIDM-1004.

5.5 Gerber Files

To download the Gerber files, see the design files at TIDM-1004.

5.6 Assembly Drawings

To download the assembly drawings, see the design files at TIDM-1004.

6 Software Files

To download the software files, see the design files at TIDM-1004.

7 Related Documentation

1. Texas Instruments, SimpleLink™ MSP432™ SDK
2. Texas Instruments, SimpleLink™ MSP432™ SDK Bluetooth Plugin
3. Texas Instruments, *CapTIvate™ Technology Guide*
4. Texas Instruments, CapTIvate™ Design Center

7.1 Trademarks

8 About the Authors

ABHINAV N. CHEVULA is an applications engineer on the MSP430 Systems Applications Team at TI in the MSP business unit since 2016. Abhinav earned his bachelor of science in electrical engineering from the University of Illinois at Urbana-Champaign, in Urbana, Illinois.

DAVID LARA is an applications engineer on the MSP432 Customer Applications Team at TI in the EP business unit since 2012. David earned his masters of science in electrical engineering from New Mexico State University in Las Cruces, NM.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265