TI Designs

MSP432™ With MSP430™ Microcontroller With CapTIvate™ Technology, Haptics, and LCD Reference Design

TI Designs

This TI Design demonstrates how to interface an MSP430™ microcontroller featuring CapTIvate™ technology with a host microcontroller using the CapTIvate Software Library communications module. The design integrates CapTIvate touch technology with haptics, an LCD with resistive touch technology, and an MSP432™ host.

Design Resources

TIDM-CAPTIVATE-MSP432	Design Folder
MSP430FR2633	Product Folder
MSP432P401R	Product Folder
DRV2605L	Product Folder
MSP-CAPT-FR2633	Tools Folder
MSP-EXP432P401R	Tools Folder
BOOSTXL-K350QVG-S1	Tools Folder
CapTIvate Design Center	Tools Folder

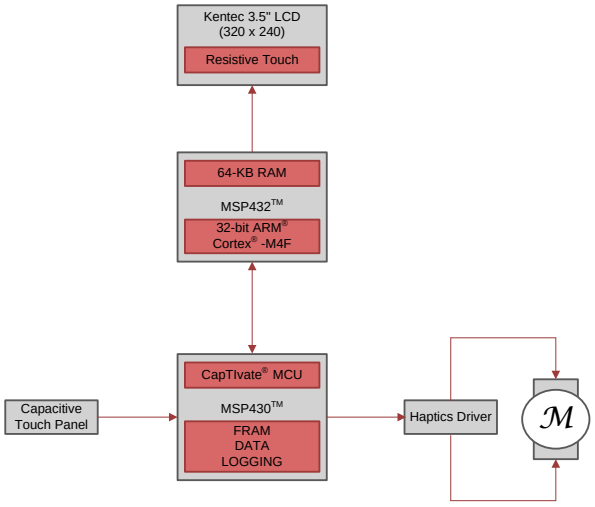
TI E2E™ Community — ASK Our E2E Experts

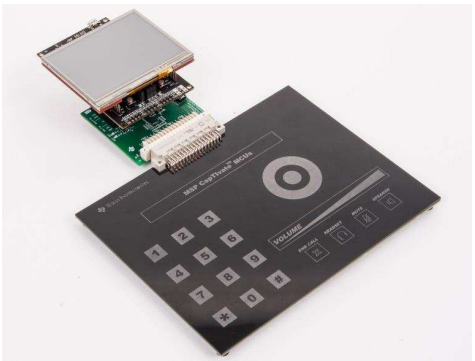
Design Features

- Ultra-Low-Power MSP430 Microcontroller Featuring CapTIvate Touch Technology and FRAM
- Mutual Capacitance HMI With 17 Buttons, 2 Sliders, 1 Wheel, a Proximity or Guard Channel, and Haptics
- MSP432 Low-Power, High-Performance 32-Bit ARM® Cortex®-M4F Host
- 320 × 240 Pixel SPI-Controlled TFT QVGA Display With Resistive Touch Screen
- CapTIvate Design Center Support

Featured Applications

- IP Phone
- Industrial Panels
- Home Appliances





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 Key System Specifications

Table 1 lists the key specifications for the MSP430 microcontroller (MCU) featuring CapTIvate technology.

Table 1. Key System Specifications

FEATURES	SPECIFICATIONS	ADDITIONAL DETAILS
Interface composition	17 buttons, 2 sliders, 1 wheel, 1 proximity or guard channel	—
Serial interfaces	UART, I^2C	—
Measurement time	31 ms to 65 ms (approximately 1 to 2 scan periods)	33-ms scan period
Touch to Display response time	I^2C, 4 ms to 6 ms; UART, 18 ms to 19 ms	—
Average power consumption	720 µA	CapTIvate hardware only, UART mode 33-ms scan period
MCUs	MSP430FR2633, MSP432P401R	—
Memory footprint (MSP430FR2633)	8317 bytes of FRAM, 2995 bytes of RAM	—
Memory footprint (MSP432P401R)	117,142 bytes flash, 4239 bytes of RAM	—

2 System Description

A capacitive touch MCU takes on two different roles in a system. The MCU is the application processor or acts as a dedicated human-machine interface (HMI). A dedicated capacitive touch HMI resolves useable information from capacitive touch sensors and usually requires an interface to a host processor. Using a dedicated capacitive touch HMI with a host processor lets a developer integrate multiple technologies into a single product design while increasing available application bandwidth and reducing power consumption.

This TI Design demonstrates how to interface an MSP430 MCU featuring CapTIvate touch technology with a host processor. The design integrates a capacitive touch HMI with haptics, a low-power and high-performance 32-bit ARM Cortex-M4F MCU, and an LCD.

2.1 MSP430FR2633

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

MSP430FR2633 features:

- 16 CapTIvate technology I/Os that supports 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 real-time clock (RTC)
- Three Enhanced Serial Communications peripherals for UART, IrDA, SPI, and I^2C
- 19 I/Os with 16 interrupt pins for wakeup from low-power modes
- High-performance, 8-channel, 10-bit analog-to-digital converter (ADC)
- Clock system with operation up to 16 MHz

2.2 MSP432P401R

The MSP432P401R is an ultra-low-power, 32-bit ARM Cortex-M4F-based mixed-signal MCU featuring a wide range of analog, timing, and communication. The device includes 256KB of flash main memory, 16KB of flash information memory, and 64KB of SRAM. With a clock system of up to 48 MHz, the large amount of available memory makes the MSP432P401R ideal for applications that require efficient data processing with enhanced low-power operation.

MSP432P401R 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 MSPWare driver libraries
- Eight-channel direct memory access (DMA)
- Four 16-bit timers, two 32-bit timers, and an RTC with calendar and alarm functions
- Four eUSCI-A modules for UART, IrDA, or SPI
- Four eUSCI-B modules for I^2C with either multiple-slave addressing or SPI
- 48 I/Os with interrupt and wakeup capability
- 14-bit, 1MSPS SAR ADC
- 128-, 192-, and 256-bit AES encryption and decryption accelerator with 32-bit hardware CRC engine

2.3 DRV2605L

The DRV2605L is a low-voltage haptic driver for eccentric rotating masses (ERMs) and linear resonant actuators (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 is used with an LRA on the touch panel in this design to provide users with high-quality mechanical feedback which simulates the click of a button upon interaction.

3 Block Diagram

Figure 1 shows a system block diagram of the MSP430.

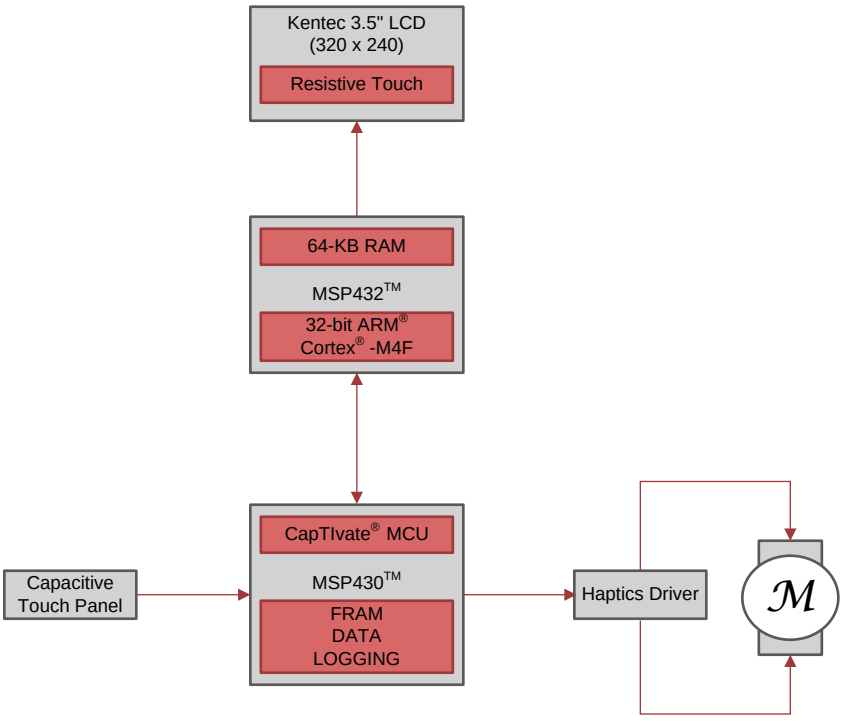


Figure 1. System Block Diagram

3.1 MSP430FR2633 MCU

Figure 2 shows a block diagram of the MSP430FR2633 MCU.

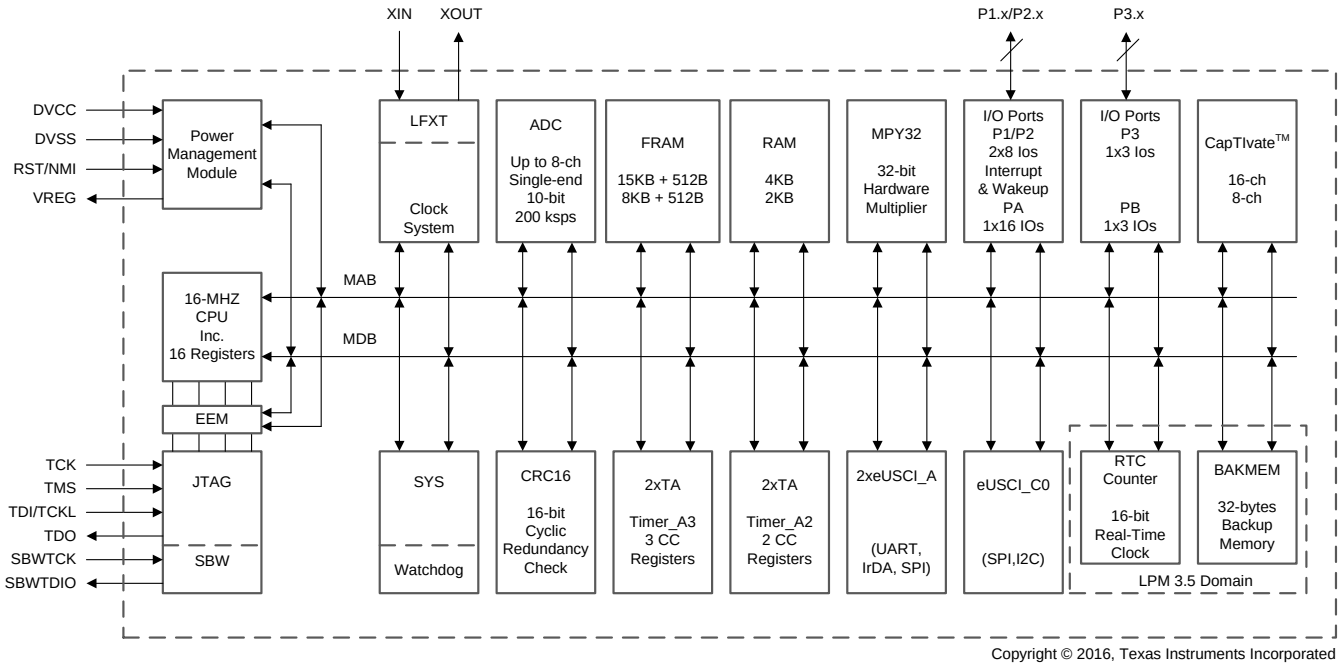


Figure 2. Block Diagram of MSP430FR2633 MCU

3.1.1 CapTIvate™ Technology

CapTIvate technology enables capacitive sensing on the TIDM-CAPTIVATE-MSP432. CapTIvate technology is an MSP430FR253x or an MSP430FR263x peripheral dedicated to providing robust capacitive-sensing measurements. Figure 3 shows a block diagram of the CapTIvate technology module.

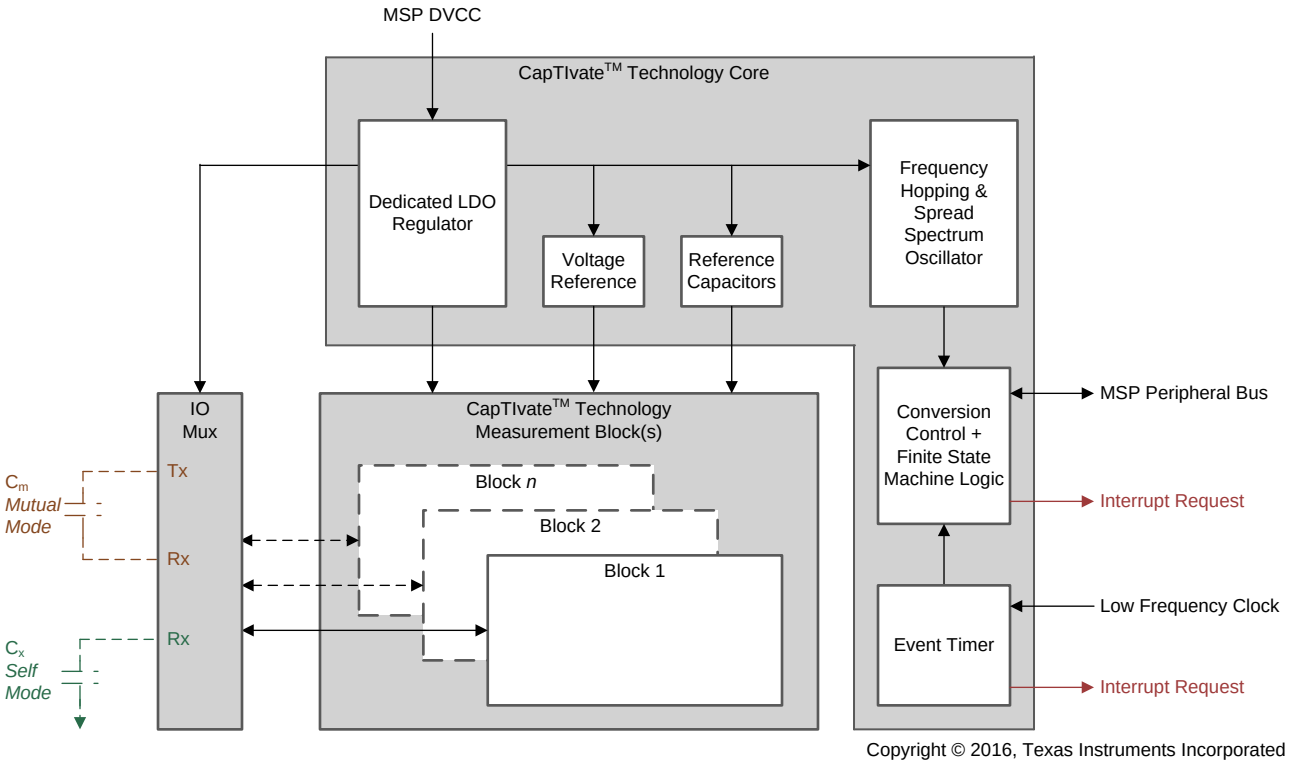


Figure 3. CapTIvate™ Technology Block Diagram

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.

3.2 MSP432P401R MCU

Figure 4 shows the functional block diagram of the MSP432P401R MCUs.

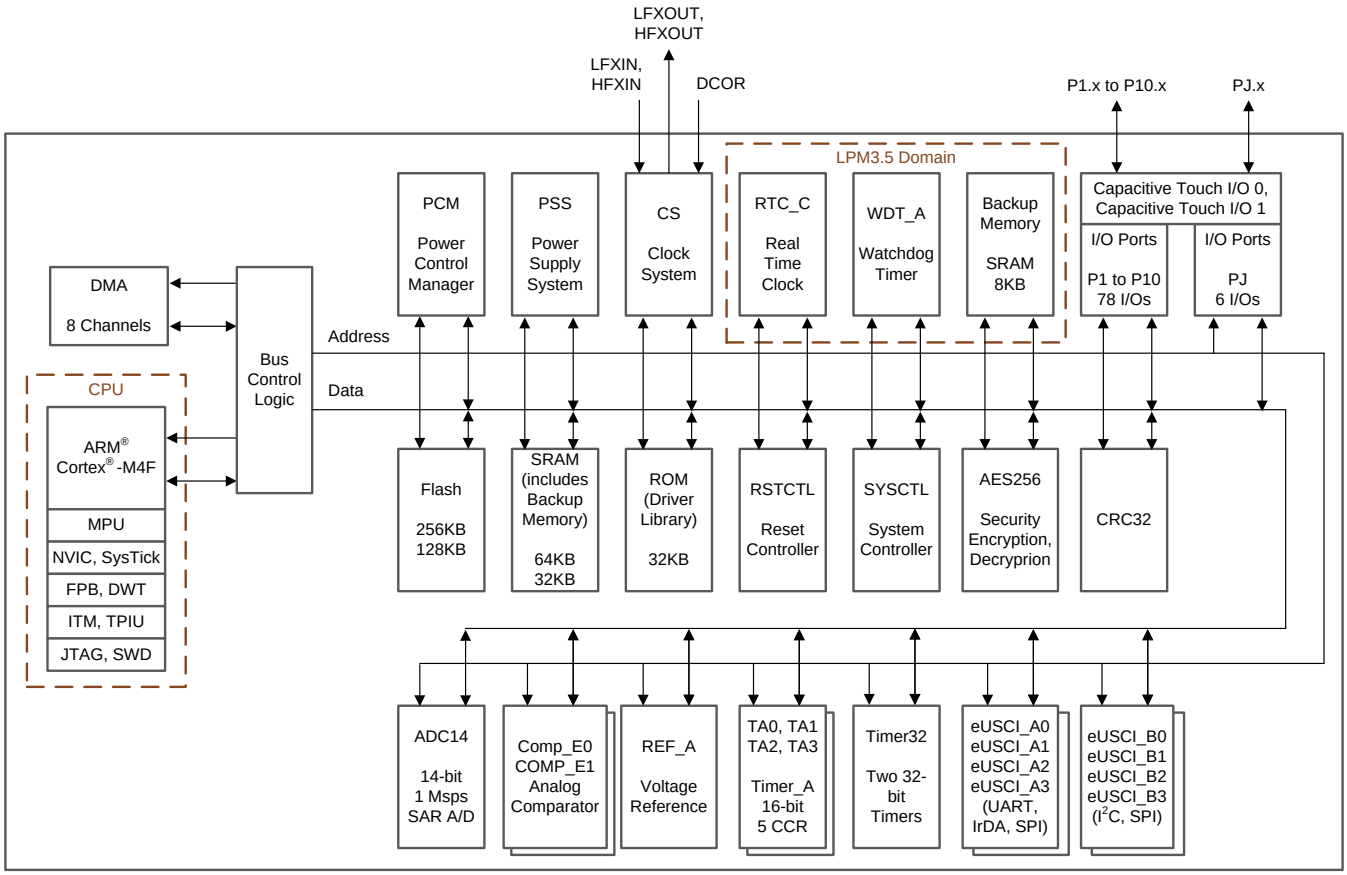


Figure 4. MSP432P401R Block Diagram

4 System Design Theory

This design takes advantage of the communications module included in the CapTIvate Software Library to create a simple interface between an MSP430 MCU with CapTIvate technology and an MSP432 MCU host.

4.1 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 a host processor through a standard, common serial interface. The communications module comprises 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

UART and I²C drivers provide the communications module to speed up development of applications that use an MSP430 MCU with CapTIvate technology with a host processor. For more information on the communications module, see the CapTIvate Technology Guide.

4.2 MSP432™ Host Communications Module

The simplest way to interface a host processor with an MSP430 MCU with CapTIvate technology is to develop firmware that is complementary to the CapTIvate Software Library communications module. The firmware enables a host to receive and interpret data packets from the CapTIvate technology HMI using the CapTIvate Serial Protocol. This TI Design reference design includes firmware that enables an MSP432 MCU to act as a host to an MSP430 MCU with CapTIvate technology.

4.2.1 Overview

The MSP432 host communications module reference design is a layered set of firmware that provides a simple API to interface an MSP432 MCU with a CapTIvate technology HMI. Figure 5 shows an overview of the organization of the module.

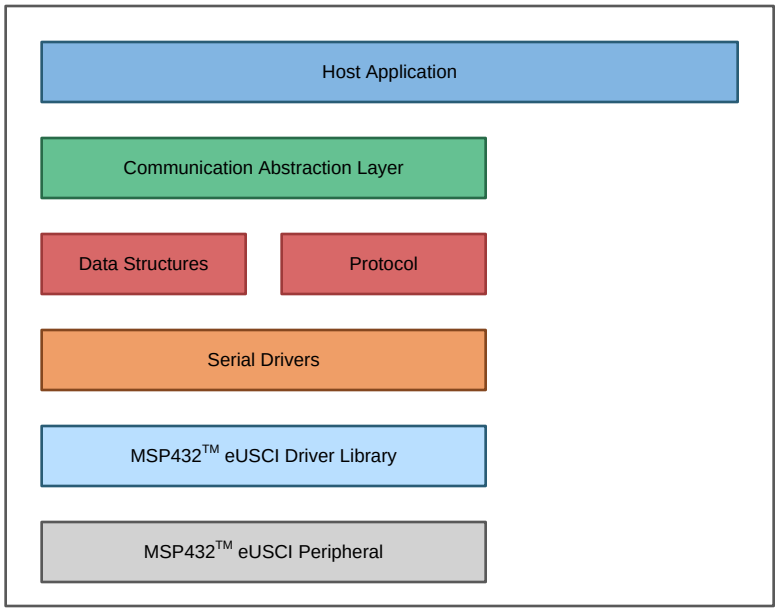


Figure 5. MSP432™ Host Application and Communications Module Organization

The interaction between each layer in the firmware changes depending on the serial interface on which the module operates. Additionally, the MSP432 communications module reference design can be separated into platform-dependent and platform-independent layers. CapTIvate technology hosts allow the reuse of platform-independent layers in firmware.

4.2.2 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 in the cycle. Cycle packets include the proximity state, touch state, count value, and 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.

4.2.3 Protocol Layer

The protocol layer is responsible for ensuring complete data is received and stored correctly during a transmission by performing the following functions:

- **UART**: parsing the receive queue for complete CapTIvate data packets
- **UART and I²C**: verifying checksums in received data

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 I²C slave address and the UART HID header bytes. For more information on the CapTIvate protocol, see the *Communications Module* under the *Software Library* section of the CapTIvate Technology Guide.

4.2.4 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 I²C 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.

4.2.4.1 UART

The UART interface configures a 250K baud rate by default. With the UART interface, data constantly streams to the host from the CapTIvate MCU. The host buffers incoming UART data by storing it in a byte queue called UART_receiveQueue.

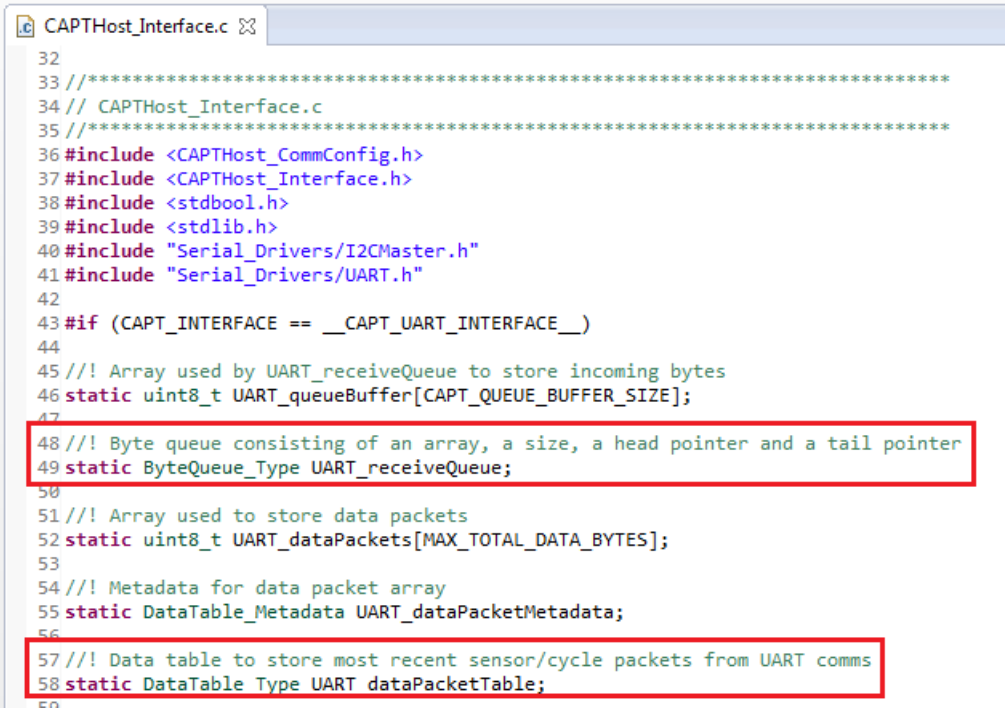
NOTE: The implementation of the receive queue is found in CAPTHost_ByteQueue.h/.c

The queue extracts individual packets and stores them in a look-up table by calling CAPT_checkForInboundPacket() periodically in the application. It is important that this function calls frequently enough to prevent the byte queue from overrunning.

UART_dataPacketTable is the look-up table. This look-up table stores the most recent version of each unique CapTIvate data packet received by the application and provides methods to access the data.

NOTE: The full implementation of the data table is found in CAPTHost_DataTable.h/.c

Figure 6 shows the declaration and initialization of both the byte queue and the look-up table in CAPTHost_Interface.c.

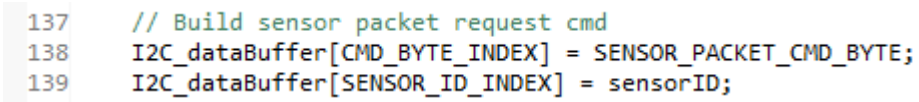


```
CAPTHost_Interface.c
32
33 //*****************************************************************************
34 // CAPTHost_Interface.c
35 //*****************************************************************************
36 #include <CAPTHost_CommConfig.h>
37 #include <CAPTHost_Interface.h>
38 #include <stdbool.h>
39 #include <stdlib.h>
40 #include "Serial_Drivers/I2CMaster.h"
41 #include "Serial_Drivers/UART.h"
42
43 #if (CAPT_INTERFACE == __CAPT_UART_INTERFACE__)
44
45 //! Array used by UART_receiveQueue to store incoming bytes
46 static uint8_t UART_queueBuffer[CAPT_QUEUE_BUFFER_SIZE];
47
48 //! Byte queue consisting of an array, a size, a head pointer and a tail pointer
49 static ByteQueue_Type UART_receiveQueue;
50
51 //! Array used to store data packets
52 static uint8_t UART_dataPackets[MAX_TOTAL_DATA_BYTES];
53
54 //! Metadata for data packet array
55 static DataTable_Metadata UART_dataPacketMetadata;
56
57 //! Data table to store most recent sensor/cycle packets from UART comms
58 static DataTable_Type UART_dataPacketTable;
```

Figure 6. Byte Queue and Look-up Table in CAPTHost_Interface.c

4.2.4.2 I²C

The I²C interface implements the register mode with a 400-kbps data rate. In register mode, the host must request specific data packets from the CapTIvate HMI each time the host requires new data. Figure 7 shows how using the I²C interface requests new data and then receives the data every time the interface calls CAPT_getSensorPacket() or CAPT_getCyclePacket(). Figure 7 also shows how the parameter-based construction of the requested data function sends the data packet and stores the data in I2C_dataBuffer[] as well as how the requested data packet sends to the MSP430FR2633 MCU.



```
137     // Build sensor packet request cmd
138     I2C_dataBuffer[CMD_BYTE_INDEX] = SENSOR_PACKET_CMD_BYTE;
139     I2C_dataBuffer[SENSOR_ID_INDEX] = sensorID;
```

Figure 7. Sensor Data Request Packet Construction from CAPTHost_Interface.c

A sensor-packet request consists of the SENSOR_PACKET_CMD_BYTE and the appropriate sensor ID. A cycle-packet request consists of the CYCLE_PACKET_CMD_BYTE, the sensor ID, and the cycle ID.

The I2C_dataBuffer array stores both the transmitted and received data. This means that each call to CAPT_getSensorPacket() or CAPT_getCyclePacket overwrites the data previously received. It is the responsibility of the application to save received data before reading in another CapTIvate data packet.

Figure 8 shows the location of the I2C_dataBuffer in CAPTHost_Inteface.c.

CAPTHost_Interface.c

```
32
33 //*****************************************************************************
34 // CAPTHost_Interface.c
35 //*****************************************************************************
36 #include <CAPTHost_CommConfig.h>
37 #include <CAPTHost_Interface.h>
38 #include <stdbool.h>
39 #include <stdlib.h>
40 #include "Serial_Drivers/I2CMaster.h"
41 #include "Serial_Drivers/UART.h"
42
43 #if (CAPT_INTERFACE == __CAPT_UART_INTERFACE__)
44
45 //! Array used by UART_receiveQueue to store incoming bytes
46 static uint8_t UART_queueBuffer[CAPT_QUEUE_BUFFER_SIZE];
47
48 //! Byte queue consisting of an array, a size, a head pointer and a tail
49 static ByteQueue_Type UART_receiveQueue;
50
51 //! Array used to store data packets
52 static uint8_t UART_dataPackets[MAX_TOTAL_DATA_BYTES];
53
54 //! Metadata for data packet array
55 static DataTable_Metadata UART_dataPacketMetadata;
56
57 //! Data table to store most recent sensor/cycle packets from UART comms
58 static DataTable_Type UART_dataPacketTable;
59
60 #elif (CAPT_INTERFACE ==  CAPT_I2C_INTERFACE  )
61 //! Transmit buffer used for I2C communication
62 static uint8_t I2C_dataBuffer[CAPT_I2C_REGISTER_RW_BUFFER_SIZE];
63 #endif
```

Figure 8. I2C_dataBuffer Array in CAPTHost_Interface.c

4.2.5 Serial Communications Drivers

The communications module provides MSP432 UART and I²C master drivers. The interface used can switch using the communication configuration file, CAPTHost_commConfig.h. Both interfaces have the same functions available, but each interface provides different performance.

4.2.5.1 UART

The UART interface configures for a default 250K baud rate. The I²C interface configures for a rate of 400K bits per second.

4.2.5.2 I²C Master

Additionally, register mode implements the I²C interface. In register mode, the host must request specific data packets from the CapTIvate HMI every time it requires new data. With a UART interface, data constantly streams to the host from the CapTIvate MCU. The host must periodically service incoming data to react to user interaction with the HMI.

4.3 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.

4.4 Haptics

In this design, the DRV2605L haptics driver activates a Samsung™ LRA to create various effects in response to a touch on the HMI. To use haptics in the demonstration, a haptics driver must use a UART interface between the CapTIvate MCU and the MSP432 host. This configuration is required because the CapTIvate MCU over an I²C interface connects and controls the haptics driver.

4.5 LCD

The MSP432 updates the Kentec QVGA Display BoosterPack™ over SPI. The example MSP-EXP432P401R_CapTIvate_PhonePanel_BOOSTXL-K35QVG-S1 updates the Kentec QVGA Display BoosterPack by using graphics library. The example labeled MSP-EXP432P401R_CapTIvate_PhonePanel_BOOSTXL-K35QVG-S1_DMA uses DMA to improve the refresh rate of the UI on the display. The latter example also implements a single general-purpose packet to restart the display with its retained state in the event of power loss.

5 Getting Started Hardware

The TI store sells each component of TIDM-CAPTIVATE-MSP432 separately.

5.1 MSP-CAPT-FR2633 MCU Development Kit

An MSP-CAPT-FR2633 MCU Development Kit is available for purchase to evaluate the CapTIvate technology in a wide range of capacitive touch configurations.

Figure 9. MSP-CAPT-FR2633 MCU Development Kit

The kit includes a CAPTIVATE-FR2633 Target MCU Module, a self-capacitance touch panel, a mutual-capacitance touch panel, and a proximity-sensing panel. These touch panels interface directly into the CAPTIVATE-FR2633 board and come with pre-made demonstration projects for easy plug-and-play use.

5.2 MSPEXP432P401R LaunchPad™ Development Kit

The 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 that includes a 48-MHz ARM Cortex-M4F, a power consumption of 95 µA/MHz active and 850 nA RTC standby operation, a 24-channel 14-bit differential 1MSPS SAR ADC, and an advanced encryption standard (AES256) accelerator.

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

5.3 BOOSTXL-K350QVG-S1 Kentec QVGA Display BoosterPack™

The BOOSTXL-K350QVG-S1 Kentec QVGA Display BoosterPack, available on the TI Store, is an easy-to-use plug-in module for adding a touch screen color display to your 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 touch screen.

6 Getting Started Firmware

The example firmware was developed using CCS v6.1.2.0015 and TI Compiler version 5.2.7. Download the latest version of CCS to evaluate the example firmware import the projects into a CCS workspace from [TI Design Software Install Root]/Software/*.

Each example project for the MSP430FR2633 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 open from the same directory as their respective CCS project.

6.1 IP Phone Panel Demonstration

The CAPTIVATE-PHONE sensing panel demonstrates the use of mutual capacitance to realize a high-density panel with many different sensor types using 12 of the 16 CapTIvate sensing pins. The panel mimics a typical office phone application that would have a 12-key numeric keypad, several mode buttons, and several selection sensors. The panel features haptic vibration feedback because the DRV2605L haptic driver IC couples with a Samsung LRA. A guard channel technique rejects palm or arm button presses as well as minor liquid spills. Data communicates back to the MSP432 by a selectable UART or I²C interface; the MSP432 then updates the Kentec QVGA display with current UI state. Buttons turn red when pressed. A yellow ring around the display indicates proximity detection. A message will appear on the display if the guard channel activates. FRAM stores the state of the display. Upon power loss, the display refreshes with the stored data packet (named startupPacket).

This panel is configured with the following settings:

- A 33-ms or 30-Hz active mode scan period providing balance between response time and power consumption when a user interacts with the panel
  - Scanning faster, at 20 ms or 50 Hz, provides a faster response time and a perceived performance benefit when working with sliders and wheels.
  - Scanning slower, at 50 ms or 20 Hz, provides lower power consumption but a higher response time.
- A 4-MHz conversion clock rate for the mutual capacitance matrix scanning at a higher frequency
- A 1-MHz conversion clock rate on the self-capacitance guard channel requiring a slower frequency
- A <2.4 ms total measurement time for all sensors

6.2 Code Examples and Operation Modes

For this example the firmware is in two parts: the host side, MSP432, and the subordinate side, MSP430FR2633. There are two projects for the host side. The first project, MSP-EXP432P401R_CapTIvate_PhonePanel_BOOSTXL-K35QVG-S1, is a demonstration source for interfacing with a CapTIvate BoosterPack, a KentecLCD BoosterPack to display the IP phone panel UI in real time. This demonstration uses graphics library functions to draw and update the UI. The second project, MSP-EXP432P401R_CapTIvate_PhonePanel_BOOSTXL-K35QVG-S1_DMA, is a demonstration that uses DMA instead of graphics library functions to improve the refresh rate of the UI on the display. This example also uses a single general-purpose packet to restart the display with the retained state after a power loss.

CAPTIVATE-FR2633_PHONE_UART_Demo supports MSP430FR2633. The CAPTIVATE-FR2633_PHONE_UART_Demo is the CapTIvate BoosterPack example firmware that uses UART communication with a host and includes I²C master drivers to drive haptics on the IP phone panel. This firmware stores features with a UI retained state in FRAM in case of power loss.

Captivate_host_comms is a folder that contains a firmware example module that can be added to an MSP432 firmware project to create an interface to a CapTIvate device. The demonstration can operate with a UART interface or an I²C interface; however, haptics will not be active when using the I²C interface. In the I²C example, the MSP430FR2633 MCU is a slave to the MSP432 which prevents the MSP430FR2633 from acting as a master to the haptic driver IC. The code ships with the UART configuration.

To switch to I²C:

- In the CAPTIVATE-FR2633_PHONE_UART_Demo project:
  - Expand the captivate_config folder.
    - In CAPT_UserConfig.h change line 102 from #define CAPT_INTERFACE (__CAPT_UART_INTERFACE__) to #define CAPT_INTERFACE (__CAPT_REGISTERI2C_INTERFACE__)
  - In the I2CMaster folder: right-click I2CMaster.c and click (check) Exclude From Build
- In the MSP-EXP432P401R_CapTIvate_PhonePanel_BOOST-XL-K350QVG-S1(_DMA) project:
  - Expand the captivate_host_comms folder
    - In CAPTHost_CommConfig.h change line 57 from #define CAPT_INTERFACE (__CAPT_UART_INTERFACE__) to #define CAPT_INTERFACE (__CAPT_I2C_INTERFACE__)

6.3 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:

- It has integrated ROM effect libraries that are prelicensed from Immersion Corporation.
- It supports ERM and LRA haptic actuators.
- It has a simple I²C register interface.

The demonstration firmware includes an I²C master driver and a DRV26x driver to communicate with the DRV2605L. Setting up the DRV2605L with these modules is accomplished in the Demo_init() function is shown in Figure 10.

```
//
// Open the I2C Master driver, which the DRV26x driver will use to
// communicate with the DRV2605L haptic driver IC via I2C.
// Enable the haptic driver by setting P1.0, which is connected
// to the DRV2605L ENABLE pin. Then, load the configuration for the
// actuator, run an auto-calibration routine, set up for internal
      trigger
// mode, and select the linear resonant actuator (LRA) effect
      library.
//
I2CMaster_open();
P1OUT |= BIT0;
DRV26x_reset();
DRV26x_exitStandby();
DRV26x_loadActuatorConfig(&DRV26x_actuator_DMJBRN1030);
DRV26x_runAutoCalibration();
DRV26x_setMode(DRV26x_mode_internalTrigger);
DRV26x_selectEffectLibrary(DRV26x_lib_ROM_LRA);
```

Figure 10. Demo_init() Function

The CapTIvate Software Library uses the callback capability to trigger playback of haptic events directly from the library. Figure 11 shows a sample callback function. Effects are fired on a *new* touch—if *touch* is true and *previous touch* is false. The callback exits if the guard mask is active, which occurs if the guard channel is in detect and the touch on this sensor is masked.

```
void Demo_numericKeypadHandler(tSensor* pSensor)
{
//
// If the guard mask is activated, abort here and do not process
// any events.
//
if (Demo_guardMaskActive == true)
{
return;
}
//
// If the sensor has a new touch, fire a "strong click" effect.
//
if ((pSensor->bSensorTouch == true)
&& (pSensor->bSensorPrevTouch == false))
{
DRV26x_fireROMLibraryEffect(
DRV26x_effect_strongClick_100P,
true
);
}
}
```

Figure 11. Sample Callback Function

NOTE: Haptics are available only with the UART interface.

6.4 Guard Channel Integration

The guard channel works as a detection mask for all other sensors in the system to provide palm or arm recognition and spill rejection. The guard electrode wraps around the panel between the other sensors. When the guard electrode is not measured, it serves as a grounded shield. The data from the guard channel identifies when a user puts their hand against the sensing panel, which would otherwise cause all of the sensors to go into detect. The guard channel also detects if the panel surface is wiped with a cloth or if a liquid spills on the panel.

Guard channel tuning requires the following considerations:

1. Tune the touch threshold to be sensitive enough to trigger a detect when a user is incorrectly touches between keys but not when a user correctly touches one sensor.
2. Set the touch debounce in parameter to 0 and the debounce out parameter to a maximum of 15. This causes the guard channel to engage immediately in a detect situation and to remain in detect for 15 samples even after the user clears the threshold, which improves the power of the mask.
3. Set the touch debounce in to a value of at least 1 on all other sensors so the guard channel mask has one sample to kick, which prevents false touch detections.
4. Test multiple use cases and approach angles to the panel to ensure that the guard channel touch detection flag is setting before the other sensors. These tests make it easy to determine the correctness of the sensors because of the haptic feedback.

The two LEDs on the CAPTIVATE-FR2633 module indicate the status of the guard channel. When a valid touch is detected on an element, LED1 lights. When the guard channel is active, LED2 lights.

Figure 12 is an sample callback handler for the guard channel.

```
void Demo_guardChannelHandler(tSensor *pSensor)
{
//
// If the guard channel is detecting a touch,
// set the guard mask active flag to mask all other
// touch processing.
//
if (pSensor->bSensorTouch == true)
{
Demo_guardMaskActive = true;
if (pSensor-> bSensorPrevTouch == false)
{
DRV26x_fireROMLibraryEffect(
DRV26x_effect_smoothHum3_30P,
false
);
}
}
//
// If the guard channel is not detecting a touch,
// clear the guard mask active flag to allow standard
// touch processing.
//
else
{
Demo_guardMaskActive = false;
}
}
```

Figure 12. Sample Callback Handler for the Guard Channel

NOTE: The guard channel will not mask the reporting of touch and proximity events on other sensors to the CapTIvate Design Center. Instead, the mask is application-level and controls the haptic-effect playback and LED illumination. The data flowing back to the design center provides the true state of each sensor at all times.

7 Testing

7.1 Power Consumption

With the MSP432 MCU detached and the MSP430FR2633 MCU using the UART interface, a high-resolution source meter was used to measure power consumption. Figure 13 shows the unaltered example running at an average of 720.6 µA with a measurement cycle period of about 32.504 ms.

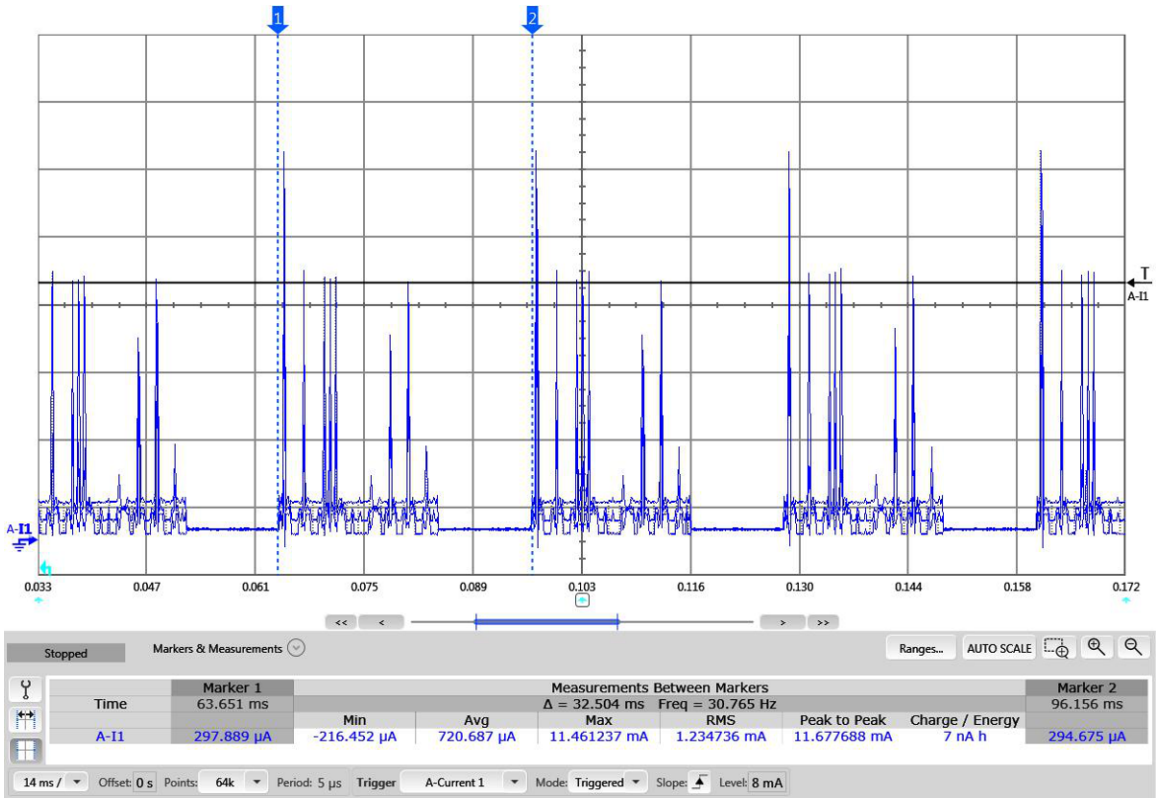


Figure 13. Current Consumption Timing With UART and LPM0 Sleep

Figure 14 zooms in to show seven distinct current spikes corresponding to measurement of the seven sensors returning to LPM0. The activity between the spikes corresponds to the UART activity and post-processing after measurements.

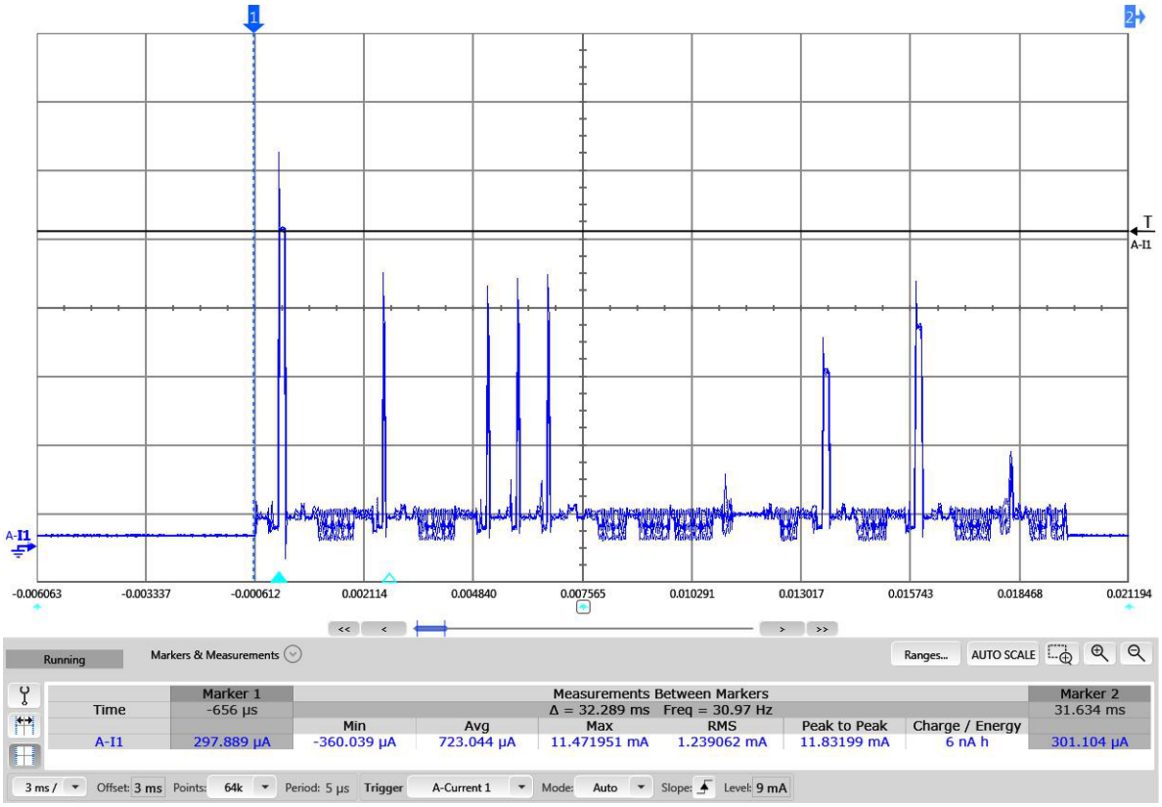


Figure 14. Current Consumption With UART and LPM0 Sleep, Zoomed

Figure 15 shows how disabling the UART interface drops the current average to 582 µA.

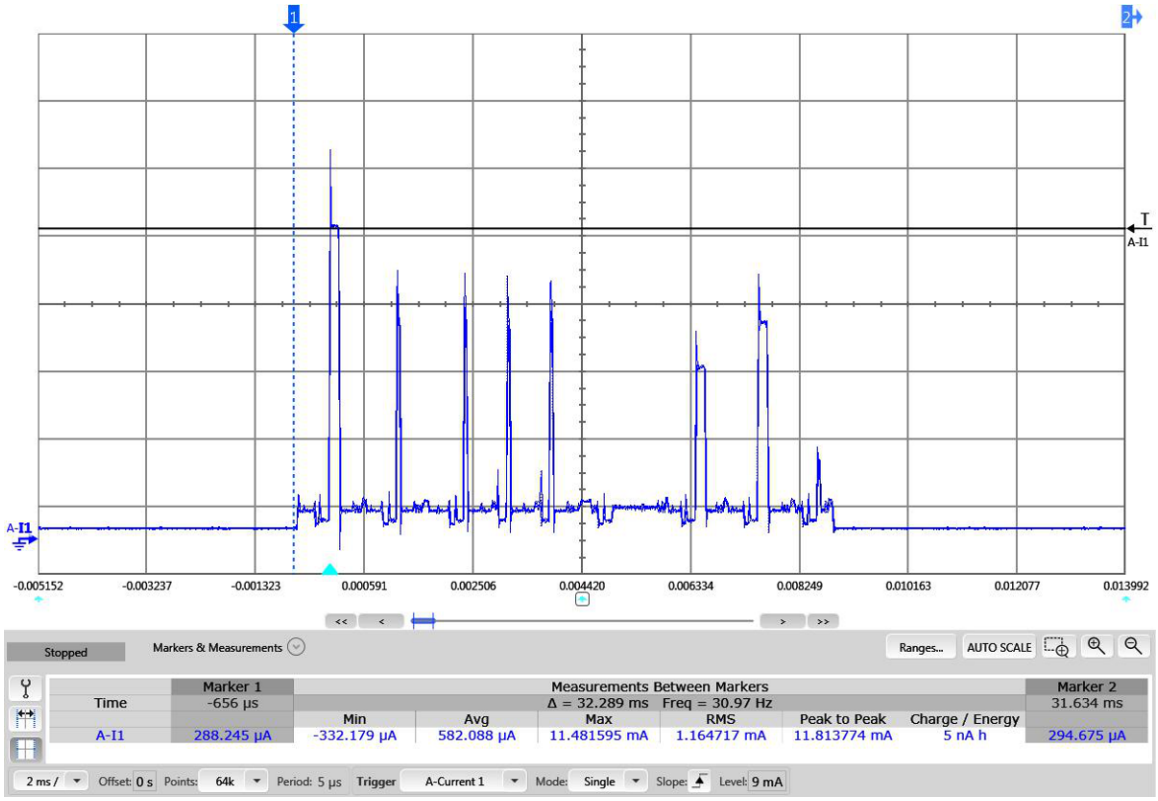


Figure 15. Current Consumption Without UART and LPM0 Sleep

Figure 16 shows how further power saving can be accomplished by placing the MSP432 into LPM3 between measurement cycles, which results in an average current of 386.62 µA.

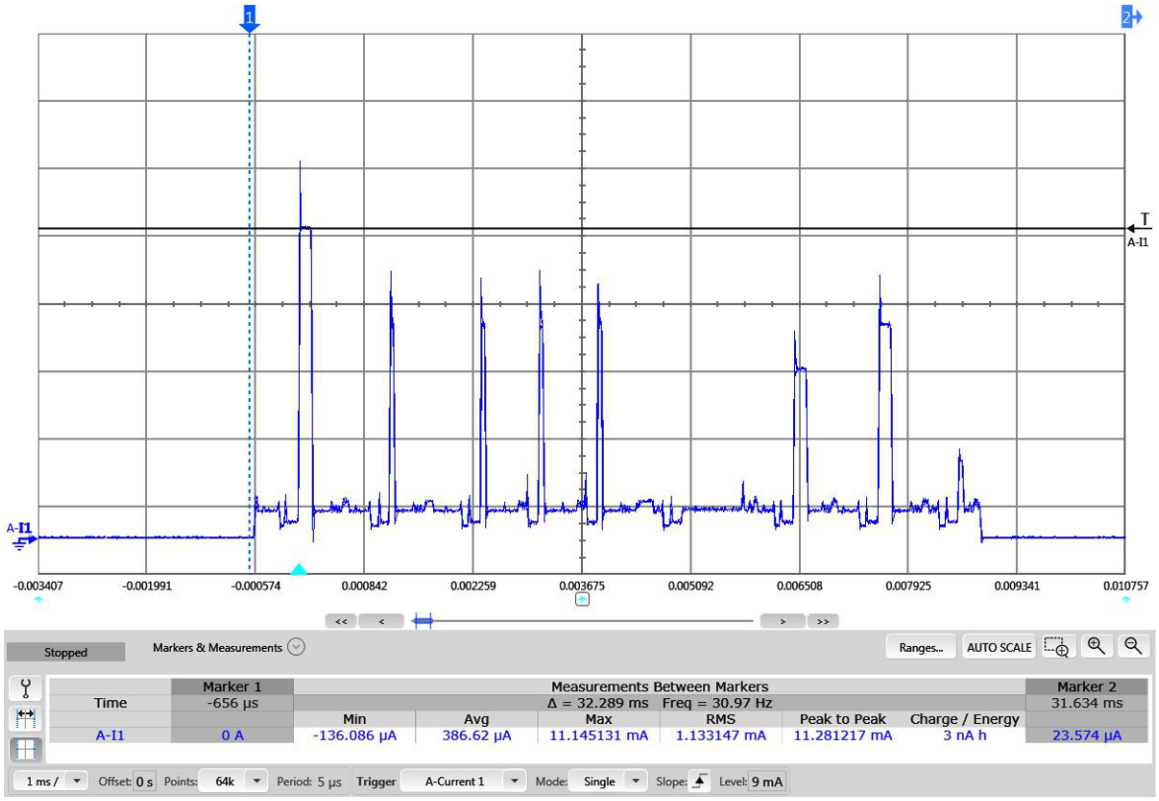


Figure 16. Current Consumption Without UART and LPM3 Sleep

7.2 Measurement and Response Time

The supplied firmware toggles three GPIOs at specific times: when a CapTIvate detect flag occurs, when a CapTIvate touch flag occurs, and when the MSP432 updates the Kentec display. A logic analyzer measured the timing of the pins to gain insight of the delay. The logic analyzer measured the time from when the sensor changes to when a touch processes, propagates to the host, and updates the display. The detect-to-touch delay is approximately one to two scan periods long. If a detect happens after the sensor occurs for that scan period, an additional scan period must occur to process the touch.

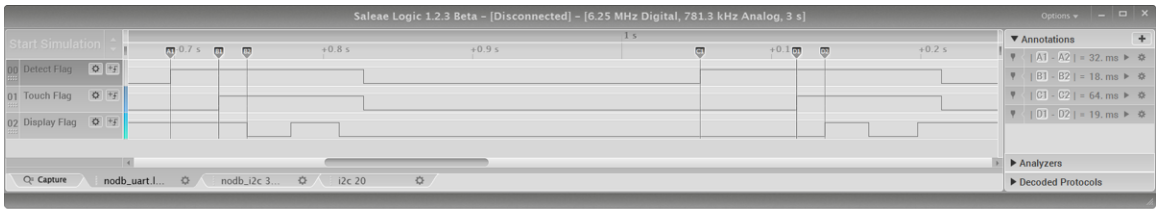


Figure 17. UART Response Time, 33-ms Scan Rate

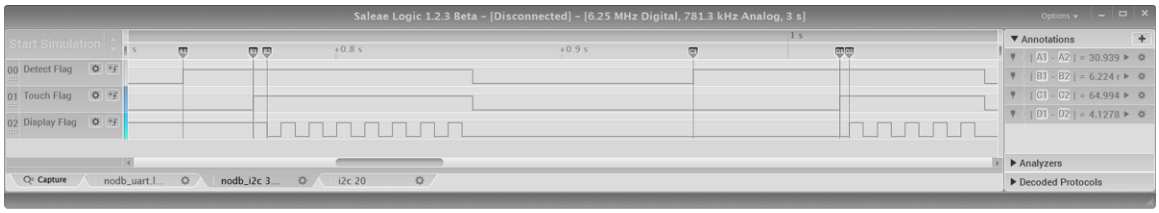


Figure 18. I²C Response Time, 33-ms Scan Rate

Table 2 shows the worse-case response time measurements.

Table 2. Worst-Case Response Time Measurements

SETTING AND MEASUREMENT NUMBER	DETECT-TO-TOUCH DELAY (ms)	TOUCH-TO-DISPLAY UPDATE DELAY (ms)	TOTAL DELAY (ms)
I²C 1	33.03	4.13	37.16
I²C 2	30.94	6.22	37.16
I²C 3	64.99	4.12	69.11
I²C 4	63.88	4.09	67.97
I²C 5	32.15	5.01	37.16
I²C 6	33.03	4.13	37.16
I²C average	43.00	4.62	47.62
UART 1	32.27	18.66	50.93
UART 2	64.51	19.01	83.52
UART 3	64.53	19.31	83.84
UART 4	64.53	18.97	83.50
UART 5	32.27	18.84	51.11
UART 6	64.90	17.96	82.86
UART average	53.83	18.79	72.63

8 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	Download
Kentec QVGA BoosterPack	Download
CapTIvate Development Kit	Download

8.1 Schematics

To download the schematics, see the design files at TIDM-CAPTIVATE-MSP432.

8.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDM-CAPTIVATE-MSP432.

8.3 PCB Layout Recommendations

See the recommendations outlined in the TIDM-CAPTIVATE-MSP432.

8.3.1 Layout Prints

To download the layer prints, see the design files at TIDM-CAPTIVATE-MSP432.

8.4 Altium Project

To download the Altium project files, see the design files at TIDM-CAPTIVATE-MSP432.

8.5 Layout Guidelines

Follow the guidelines provided at TIDM-CAPTIVATE-MSP432.

8.6 Gerber Files

To download the Gerber files, see the design files at TIDM-CAPTIVATE-MSP432.

8.7 Assembly Drawings

To download the assembly drawings, see the design files at TIDM-CAPTIVATE-MSP432.

9 References

1. Texas Instruments CapTIvate Technology Guide: http://www.ti.com/CapTIvateTechGuide
2. Texas Instruments CapTIvate Design Center: http://www.ti.com/CapTIvate

10 About the Author

BENJAMIN MOORE is an Applications Engineer on the MSP Microcontroller System Applications Team working with TI since 2013 in the C2000 and MSP business units. Benjamin earned his Bachelor of Science in Electrical and Computer Engineering (BSECE) from Ohio State University in Columbus, Ohio.

CAMERON P. LaFOLLETTE is an Applications Engineer on the MSP Customer Applications Team working with TI since 2013 in the MSP business unit. Cameron earned his Bachelor of Science in Electrical and Computer Engineering (BSECE) from Iowa State University in Ames, Iowa.

IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ('TI") reference designs are solely intended to assist designers ("Designer(s)") who are developing systems that incorporate TI products. TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.

TI's provision of reference designs and any other technical, applications or design advice, quality characterization, reliability data or other information or services 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 reference designs or other items.

TI reserves the right to make corrections, enhancements, improvements and other changes to its reference designs and other items.

Designer understands and agrees that Designer remains responsible for using its independent analysis, evaluation and judgment in designing Designer's systems and products, and has full and exclusive responsibility to assure the safety of its products and compliance of its products (and of all TI products used in or for such Designer's products) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to its applications, it has 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. Designer agrees that prior to using or distributing any systems that include TI products, Designer will thoroughly test such systems and the functionality of such TI products as used in such systems. Designer may not use any TI products in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death (e.g., life support, pacemakers, defibrillators, heart pumps, neurostimulators, and implantables). Such equipment includes, without limitation, all medical devices identified by the U.S. Food and Drug Administration as Class III devices and equivalent classifications outside the U.S.

Designers are authorized to use, copy and modify any individual TI reference design only in connection with the development of end products that include the TI product(s) identified in that reference design. HOWEVER, 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 published by TI regarding third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of the reference design or other items described above 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 REFERENCE DESIGNS AND OTHER ITEMS DESCRIBED ABOVE ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, 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 DESIGNERS AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS AS DESCRIBED IN A TI REFERENCE DESIGN 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 THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TI's standard terms of sale for semiconductor products (http://www.ti.com/sc/docs/stdterms.htm) apply to the sale of packaged integrated circuit products. Additional terms may apply to the use or sale of other types of TI products and services.

Designer will fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's non-compliance with the terms and provisions of this Notice.

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