

MSP430™ Capacitive Touch Power Designer GUI

The *MSP430 Capacitive Touch Power Designer* enables the calculation of the estimated average current draw for a given MSP430™ capacitive touch system.

Contents

1	Introduction	2
2	Installing the MSP430 Capacitive Touch Power Designer	2
3	Tool Setup	3
4	Understanding the System Statistics Section	4
5	Understanding the Estimated Device Current Section	6
6	Generating a Compare and Contrast Plot	7
Appendix A	Procedure for Adding a Device	10
Appendix B	Example Code to Reproduce Tool-Estimated Results	14

List of Figures

1	GUI Tab Selection	3
2	System Statistics Section	5
3	Estimated Average Current Section	6
4	Compare and Contrast: X-Axis Parameter of Interest Selection.....	8
5	Compare and Contrast: Series Selection	9
6	Example Timing Diagram.....	12

List of Tables

1	Device Power Profile Table (Microsoft Excel version)	11
---	--	----

1 Introduction

The *MSP430 Capacitive Touch Power Designer* enables the calculation of the estimated average current draw for a given MSP430 capacitive touch system. By entering system parameters such as operating voltage, frequency, number of buttons, and button gate time, the user can have a power estimate for a given capacitive touch configuration on a given device family in minutes. In addition, the tool allows each design parameter to be compared with average current, allowing the user to understand how various system configurations can affect power. The software is available for Microsoft® Windows 7® and Windows XP® operating systems. It is comprised of two components: the core tool (GUI), and the individual device power profiles (CSV data files).

DISCLAIMER

The estimated currents calculated by the *MSP430 Capacitive Touch Power Designer* are only estimates, and are not guarantees of system performance. The estimates are based upon lab testing and provide accurate predictions of what users can expect, but every system implementation has some variations.

1.1 Scope of Tool

The *MSP430 Capacitive Touch Power Designer* provides average supply current estimates for various capacitive touch system configurations. The estimates are for MSP430 systems running the TI Capacitive Touch Sense Library ([CAPSENSELIBRARY](#)).

This includes:

- *Active mode (software processing) duty cycle.* This includes register configurations, baseline tracking, and button status reporting, all per the CTS library. In other words, all of the software that is needed for a detect and no detect CapTouch button.
- Electrode scan duty cycle
- Sleep mode duty cycle

This does NOT include:

- *Any communication peripherals or software.* Power goes up when a UART, SPI, or I2C communications protocol is added.
- *Any gesture interpretation software.* Another software layer for gestures adds to the active mode duty cycle. TI does not provide software for interpreting gestures.
- *Any debouncing.* With 1x/2x debouncing for a given key, average supply current consumption rises during touches.

2 Installing the MSP430 Capacitive Touch Power Designer

The MSP430 Capacitive Touch Power Designer can be downloaded from <http://www.ti.com/lit/zip/slac551>. The download is compressed as a zip file. The zip file includes an installation utility for the GUI, a power profile template, power profiles for MSP430G2xx2, MSP430G2xx3, MSP430FR58xx, and MSP430FR59xx, and this user's guide.

To install the GUI:

1. Download and extract the *MSP430 Capacitive Touch Power Designer* package.
2. In a file browser, navigate to the extracted directory and open the GUI_Installer directory.
3. If any previous versions of the *MSP430 Capacitive Touch Power Designer* are installed, remove in the Windows control panel using the *Programs and Features* tool if you are using Windows 7 or the *Add and Remove Programs* tool if using Windows XP.
4. In the GUI_Installer directory, run *Setup_Run_Me.exe*. This installs the GUI tool to Windows. You may see an "Application Install – Security Warning" indicating that the publisher is not signed by Microsoft. Click "Install" to disregard.
5. The GUI tool opens automatically after it is installed. It is also accessible in the Windows 7 start menu.
6. Save the extracted *MSP430 Capacitive Touch Power Designer* package somewhere on your computer, as it contains templates for adding additional devices to the GUI tool.
7. Enjoy using the tool! See [Section 3](#) through [Section 6](#) of this user's guide to learn the functionality of the GUI. Problems or errors with the tool can be reported to the TI MCU Applications team.

3 Tool Setup

The MSP430 Capacitive Touch Power Designer GUI is composed of two tabs: The System Setup tab and the Compare and Contrast tab. These are identified in Figure 1. When the tool is opened, a default system configuration is loaded into the System Setup tab. All of the system parameters for a given system are customizable in the System Parameters section (green).

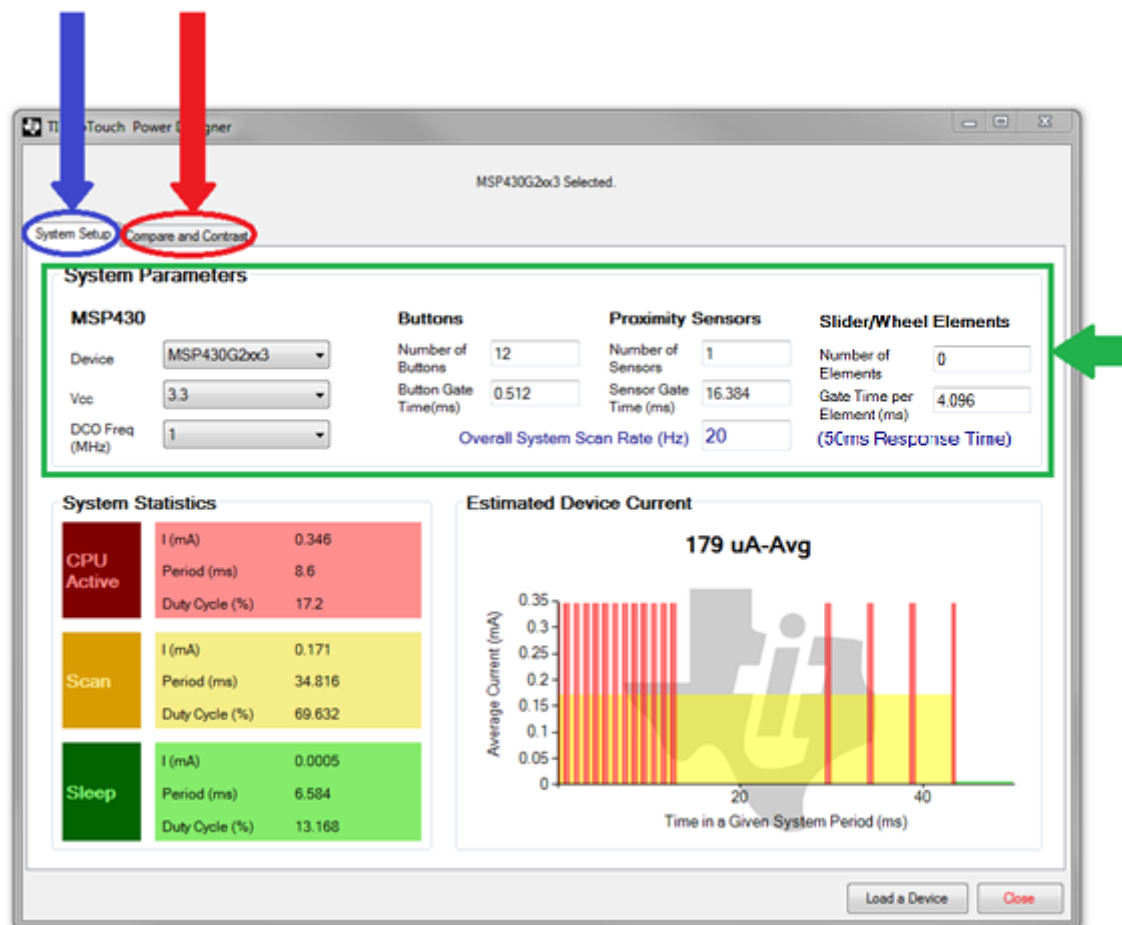


Figure 1. GUI Tab Selection

3.1 Selecting a Device

Different devices can be selected and compared in the GUI. The "Device" drop-down in the System Parameters section displays all of the currently loaded devices. The GUI comes pre-loaded with MSP430G2xx3, MSP430FR58xx, and MSP430FR59xx. Different devices can be added by selecting the "Load a Device" button at the bottom right of the GUI.

3.2 Selecting a Supply Voltage

One of the available supply voltages for the currently selected device is selected in the "Vcc" drop-down. Not all devices have the same operating voltages available.

3.3 Selecting a Frequency

The tool assumes that the device is operating off of the digitally controlled oscillator (DCO) at a given frequency. The frequency options that are available for the selected device and voltage are under the "DCO Freq (MHz)" drop-down. Note that higher frequencies may not be available at lower supply voltages. If a higher frequency is required for a design, the voltage may need to be higher.

3.4 Configuring Buttons, Proximity Sensors, Sliders, and Wheels

The tool offers three types of capacitive sensors that can be added to a system. Three categories exist because buttons, proximity sensors, and slider or wheel combinations require different gate times. While buttons, proximity sensors, sliders, and wheels are all based upon the same hardware, a button requires a lower resolution measurement than a proximity sensor or a slider or wheel group. For this reason, the tool includes a group for each sensor type.

The required gate time for a button is dependent upon the electrode size, overlay thickness, nearby grounded conductors, and desired feel, and other features. See the *Capacitive Touch Sensing, MSP430™ Button Gate Time Optimization and Tuning Guide* ([SLAA575](#)) for more details on how to select a gate time.

NOTE: The "Number of Elements" field under "Slider/Wheel Elements" defines how many total elements there are, not simply how many sliders or wheels there are. For example: a system with a 3-element slider and a 4-element wheel would have 7 elements, not 2.

NOTE: At least one button, proximity sensor, slider, or wheel element must be set to produce an estimate. If no elements exist, the Estimated Device Current section displays the "System has no elements" error.

3.5 Selecting a Scan Rate (Response Time)

The final system parameter is "Overall System Scan Rate (Hz)." The scan rate defines how many times per second the system (collection of all elements: buttons, proximity sensors, slider, and wheel element) is scanned. The inverse of the scan rate ($1 / \text{System Scan Rate}$) is equivalent to the approximate response time of the system to a touch event. Response time is typically dictated by the HMI specification for the system. The corresponding response time for a given scan rate is shown to the right of the scan rate entry box in blue font.

The tool only allows for one scan rate to be entered. MSP430 capacitive touch firmware designs can include what is referred to as power tiers. This simply means that a system can have two or more scan rates, during which only some of sensors may be scanned. For example: a system may scan a proximity sensor at a slow rate (5 Hz) and use a detection from this power tier to move up to the next power tier, where the system scans all of the elements and scans them at a faster rate (20 Hz) to achieve the desired response time. This allows the system scan slower until a user interaction is detected, which allows the system to be in ultra-low-power sleep modes more often. While this type of low-power firmware design is recommended, the tool does not support estimation for it. To characterize such a system, use the tool to calculate estimates for each power tier, and use those figures as a starting point.

4 Understanding the System Statistics Section

The parameters set in the System Parameters section are used to calculate the duty cycle of the system to estimate the power consumption. An MSP430-based capacitive touch solution has three operating modes: active mode (CPU time), electrode scan (the time during which a given sensor in the system is scanned), and sleep mode. The System Statistics section includes what the typical current is for each mode, how long the MSP430 is in that mode, and what percent of a given system period that mode takes up (duty cycle). [Figure 2](#) shows the three operating modes in the System Statistics section.

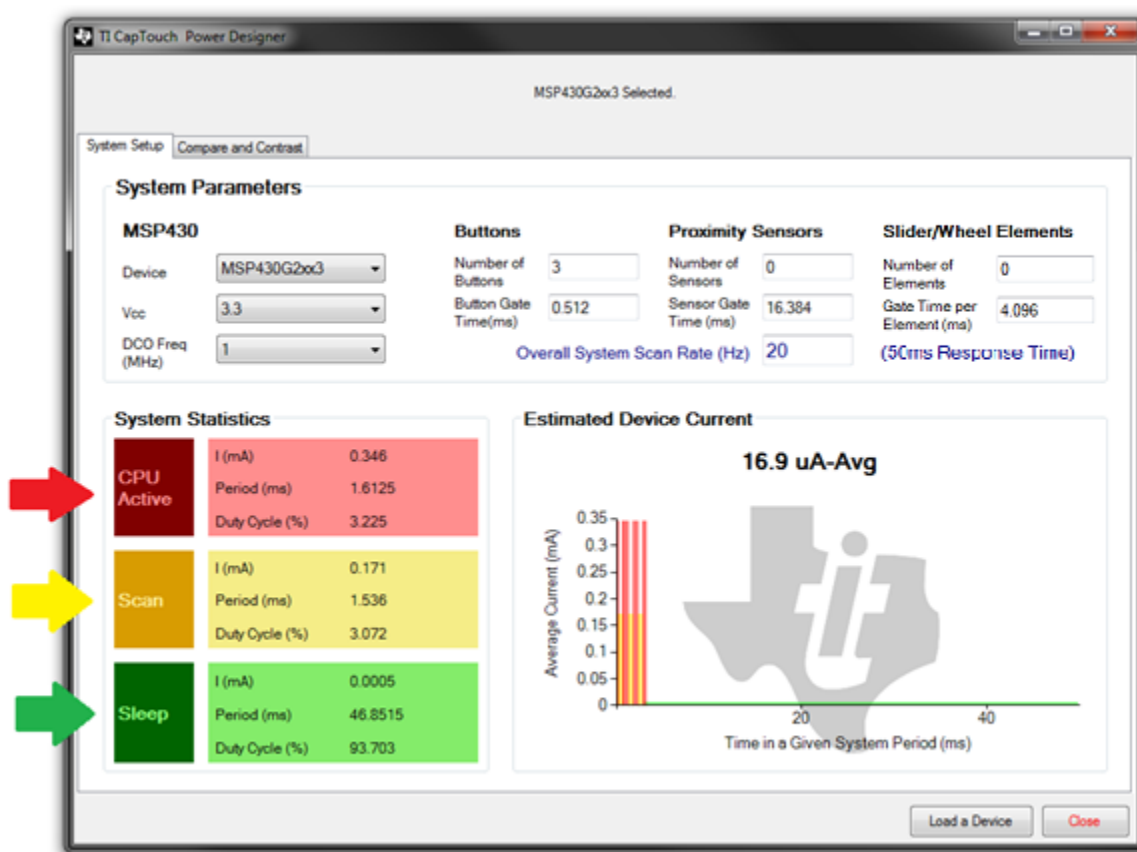


Figure 2. System Statistics Section

4.1 CPU Active Pane

The CPU Active pane (red arrow in Figure 2) indicates the supply current draw during code execution. The CPU is active in a system before and after an element is scanned. Before the scan, the CPU is used to configure the appropriate timer registers as well as any touch peripherals. After the element scan, the CPU is used to capture the measurement, determine whether the element is touched, and possibly update the element's baseline (long term average).

Increasing the DCO frequency decreases the CPU Active duty cycle but increases the CPU Active current (I). However, as frequency is increased, the ratio of $\mu\text{A}/\text{MHz}$ decreases. Therefore, the CPU is more efficient per clock cycle at a higher clock speed. Note that running the DCO at a higher speed also means that the DCO draws more power during the electrode scan phase.

4.2 Scan Pane

The Scan pane (yellow arrow in Figure 2) indicates the supply current draw during electrode scan. The period here indicates the amount of time in milliseconds that it takes to scan the entire system once. The duty cycle represents the amount of time spent in scan out of one system period.

Scanning is typically performed in low power mode 0 (LPM0), as the DCO is typically used for the measurement. If ACLK is used as the measurement timer, and the sub-main clock (SMCLK) and DCO can be disabled, then the system can be in LPM3 during electrode scan.

4.3 Sleep Pane

The Sleep pane (green arrow in [Figure 2](#)) indicates the sleep supply current draw, as well as the time spent in sleep per system period. Sleep time is typically spent in LPM3, with the very-low-power low-frequency oscillator (VLO) feeding a timer to trigger the next measurement.

5 Understanding the Estimated Device Current Section

The Estimated Device Current section displays the average current for the system (calculated from the information in the System Statistics section). It also displays a plot for a given system period (1 / system scan rate). This plot shows the duty cycle of each mode within a system period. This plot can be thought of as repeating at the frequency specified in the system scan rate.

5.1 Estimated Device Current

The estimated average device current for the system as configured in the System Parameters section is displayed in the Estimated Device Current section above the plot, as seen in [Figure 3](#). The estimated average supply current is a weighted average based upon the System Statistics. It is calculated via the equation below. Lab measurements have demonstrated the accuracy of this calculation method. To reproduce the same results on hardware, see the sample code in [Appendix B](#).

$$I_{ave} = (DC_{CPU} \times I_{CPU}) + (DC_{SCAN} \times I_{SCAN}) + (DC_{SLEEP} \times I_{SLEEP}) \quad (1)$$

5.2 Duty Cycle Plot

Underneath the estimated average device current is the duty cycle plot. This plot offers a view into the current draw over time for one system period. A system period is defined as (1 / System Scan Rate). The period repeats at the system scan rate. This plot provides a visual way for the designer to see how much time is really being spent in sleep versus scan and CPU Active. It allows the designer to visually observe the effects on current of reducing gating times, changing DCO frequencies, and changing system scan rate.

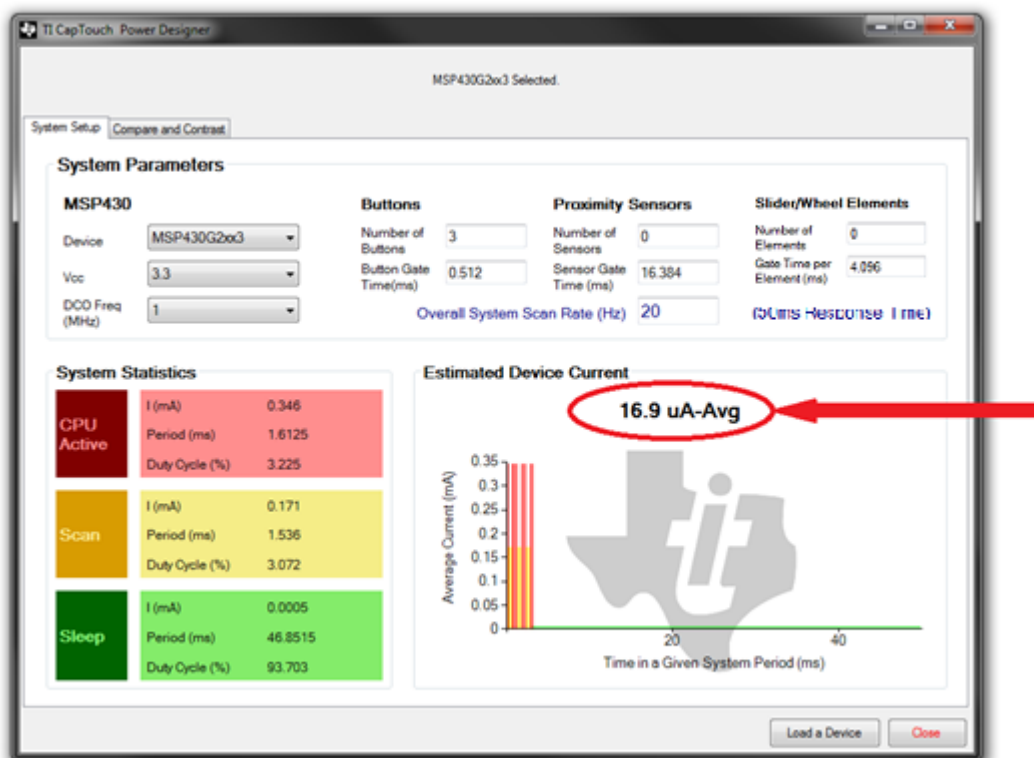


Figure 3. Estimated Average Current Section

The duty cycle plot is always the length of one system period ($1 / \text{System Scan Rate}$). As the system scan rate is increased, the system period decreases, which indicates a decrease in response time. Along those lines, the duty cycle plot demonstrates how an increase in system scan rate leads to a decrease in the total number of buttons that can be scanned.

NOTE: If a system scan rate is set that is higher than the system configuration allows, the Estimated Device Current section displays "Scan rate too high" and does not calculate any estimates.

Duty cycle plot areas that appear in red correlate to CPU active time and represent a portion of the time in the CPU active period. The CPU active period is shown in the System Statistics section.

Duty cycle plot areas that appear in yellow correlate to scan time and represent a portion of the time in the scan period. The scan period is shown in the System Statistics section, and is the sum of all element gate times.

Duty cycle plot areas that appear in green correlate to sleep time and represent the entire sleep period. The sleep period is shown in the System Statistics section, and is derived from the system period less the CPU active period and scan period.

NOTE: The green "sleep" plot area on the duty cycle plot has been enhanced 10X in the Y direction for visibility on the plot. Actual current draw in sleep is 10X less than what is shown on the plot. The values in the System Statistics section and the values used to calculate average estimated device current reflect the correct, actual sleep currents.

6 Generating a Compare and Contrast Plot

In addition to providing estimated average device currents and duty cycle plots, the *MSP430 Capacitive Touch Power Designer* is capable of generating compare and contrast plots that show the effects of system parameter variance on average current.

To use the compare and contrast feature of the GUI, a given system must be configured under the System Setup tab. The compare and contrast feature allows the variance of a system parameter, but inherits all other system parameters from the System Setup tab. In this way, the compare and contrast feature displays a plot with the parameter of interest being varied from a minimum value out to, typically, the maximum value allowed by the system configuration.

The compare and contrast feature has two parameters: "X-Axis Parameter of Interest" and "Series Selection". The plot is generated by clicking the "Plot" button at the bottom right of the window.

6.1 Compare and Contrast: Selecting a Parameter of Interest

The compare and contrast feature operates by varying a system parameter over the X-axis, and displaying the corresponding estimated average supply current on the Y-axis for that configuration. The estimated average supply current at any given point on the X-axis is the current calculated by taking into account all of the system parameters on the System Setup tab with the exception of the selected parameter of interest, which is taken from the position on the X-axis.

The available parameters of interest that the user can vary over the X-axis are: gate time per button, system scan rate, and number of buttons. The parameter of interest is selected in the drop-down menu shown in [Figure 4](#).

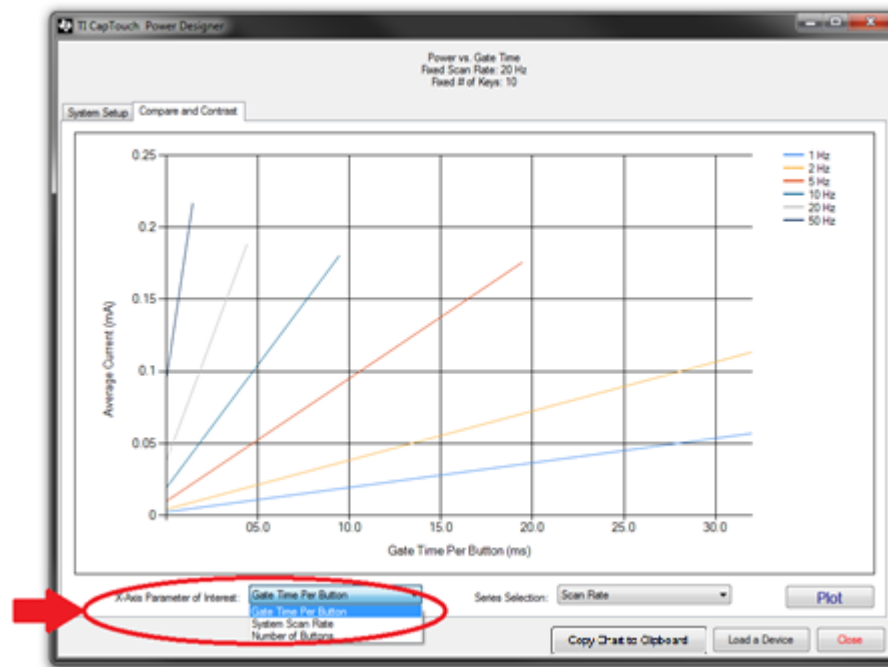


Figure 4. Compare and Contrast: X-Axis Parameter of Interest Selection

6.2 Compare and Contrast: Selecting a Series Group

In addition to allowing a system parameter to vary across the X-axis, the compare and contrast feature allows for sets of data to be varied. For example, if the X-axis parameter of interest was selected to be "Gate Time Per Button," then a series variance such as supply voltage could be selected. This would plot a series for each voltage that the selected device file supported.

The available series selections are: Device, Vcc (Supply voltage), and Frequency (DCO Freq), and Scan Rate. The series selection is made in the series selection box as shown in [Figure 5](#).

NOTE: The Scan Rate series selection is only available for the "Gate Time Per Button" X-axis parameter of interest.

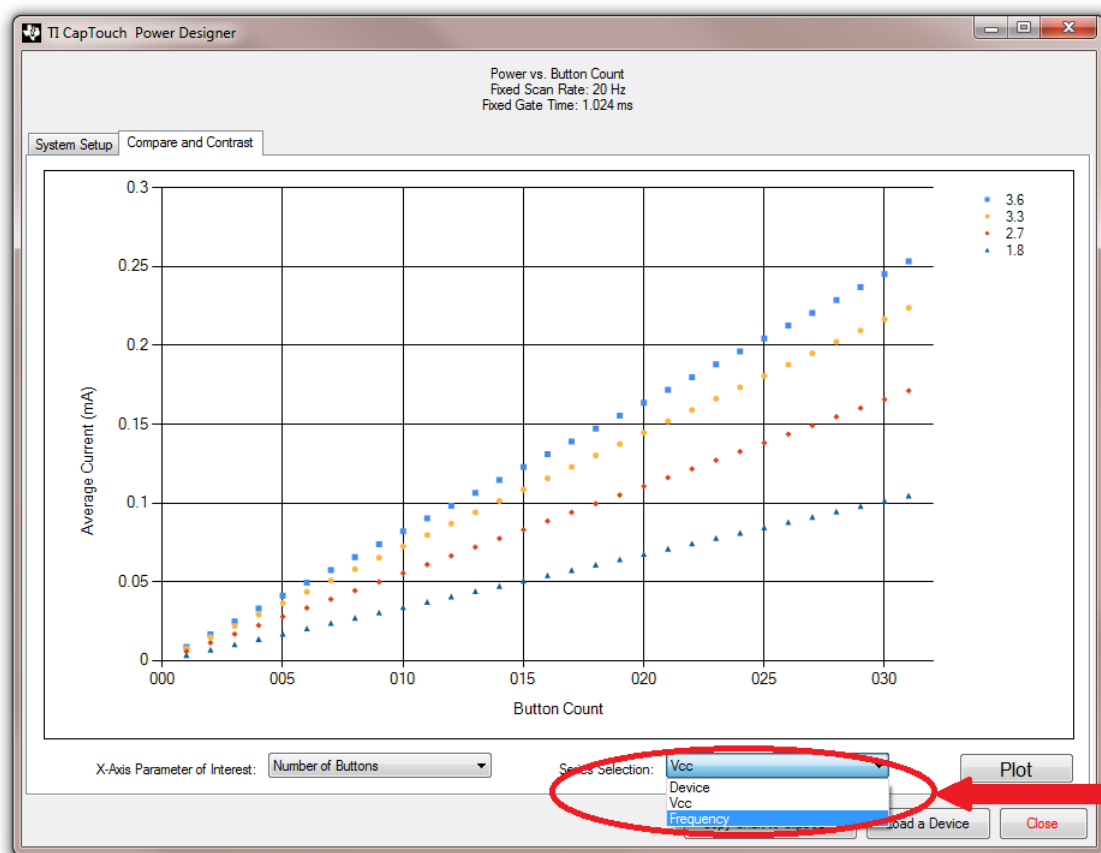


Figure 5. Compare and Contrast: Series Selection

6.3 Compare and Contrast: Copying the Chart to the Clipboard

The plot shown in the Compare and Contrast tab can be copied to the Microsoft Windows system clipboard so that it can be pasted into another document, or saved. To copy the chart, plot the desired chart then click the "Copy Chart to Clipboard" button at the bottom right of the program. Then paste the image into the program of your choice. To save the image, paste it into Microsoft Paint and use Paint to save the image.

Appendix A Procedure for Adding a Device

Devices other than the pre-loaded devices can be added to the GUI. Device power profile files use the file extension "*.ctp" and are formatted as comma-delimited text files. [Section A.1](#) describes how to add a file to the GUI. [Section A.2](#) describes how to generate a new power profile for a device.

A.1 Adding a Device Power Profile File to the GUI

A new device can be added to the GUI at any time. The file addition lasts only for runtime; if the GUI is closed, the file must be added again when the GUI is opened again. Adding a new device allows that device to be selected in the System Parameters section and also includes it in the device series set when using the Compare and Contrast feature.

To add a device, select the "Load a Device" button at the bottom of the GUI. A file browser opens. Browse to the desired device file, and select it. If the file is formatted correctly, the device is added to the current device list.

A.2 Generating a New Device Power Profile File (*.ctp)

The *MSP430 Capacitive Touch Power Designer* generates power estimates by pulling data from a device power profile file. The power profile file (*.ctp) contains several sets of data that need to be measured. The steps for how to generate a new device file are listed below. Adherence to the standards below is required for the file to be compatible with the GUI.

A.2.1 Step 1 – Opening the Template Power Profile

Step 1 in generating a new device power profile is opening the template Microsoft Excel file. The file "TI_CAPT_PWR__template.xlsx" is included in the installer .zip file with the *MSP430 Capacitive Touch Power Designer*. If Microsoft Excel is not available, there is also a text template: "TI_CAPT_PWR__template.ctp" that can be edited in a text editor, if standard comma-delimited practices are observed. A sample template spreadsheet can be seen in [Table 1](#).

Every device power profile file begins with the string **TI_CAPT_PWR** on the first line of the file.

The second line of the file is always the **device or device family name** for the following data. Typically, this is in the format MSP430tXYZZ, where 't' represents the device type, X represents the generation, Y represents the device family, and Z represents the series and device number.

The third line of the file is always left **blank**.

A.2.2 Step 2 – Identifying DCO Frequencies

The DCO frequencies to be listed for the new device are listed in the fourth line of the file (after the blank third line.) The first cell always reads **DCO FREQUENCIES**, followed by a blank cell. After the blank cell, each frequency is listed in its own cell. Frequencies are listed in Hz. See the example in [Table 1](#) for 1 MHz, 8 MHz, 12 MHz, and 16 MHz. For each frequency that is listed, CPU setup times and CPU processing times need to be measured.

Table 1. Device Power Profile Table (Microsoft Excel version)

TI_CAPT_PWR					
MSP430XXXX					
DCO FREQUENCIES		1000000	8000000	12000000	16000000
CPU_SETUP_TIMES		0	0	0	0
CPU_PROC_TIMES		0	0	0	0
SUPPLY_VOLTAGES					
3.6 CPU_I		X	X	X	X
3.6 SCAN_I		X	X	X	X
3.6 SLEEP_I		X	X	X	X
3.3 CPU_I		X	X	X	X
3.3 SCAN_I		X	X	X	X
3.3 SLEEP_I		X	X	X	X
2.7 CPU_I		X	X	X	X
2.7 SCAN_I		X	X	X	X
2.7 SLEEP_I		X	X	X	X
1.8 CPU_I		X	X	X	X
1.8 SCAN_I		X	X	X	X
1.8 SLEEP_I		X	X	X	X

A.2.3 Step 3 – Identifying CPU Times for Each Frequency

Lines 5 and 6 of the file always represent the CPU setup times for each frequency and the CPU processing times for each frequency, respectively. The lines begin with **CPU_SETUP_TIMES** (line 5) and **CPU_PROC_TIMES** (line 6), followed by a blank cell as seen in [Table 1](#). Following the blank cells, the associated CPU setup time or CPU processing time for each frequency is recorded. For example, the CPU setup time and CPU processing time for 1MHz operation are entered in rows 5 and 6 below the 1-MHz cell in row 4.

CPU times are measured by setting up a test MCU with one capacitive touch element.

The CPU setup time is defined as the time from the moment the MSP430 exits sleep until the moment it begins to scan the element. This includes all of the code execution to come out of sleep, make library calls, configure the registers in the HAL, all up until the HAL sends the MCU into LPM0 or LPM3 for the element scan. This can be seen in [Figure 6](#).

The CPU processing time is defined as the time from the moment the MSP430 exits LPM0 or LPM3 after the element is scanned until it returns to sleep after the measurement has been processed. In an actual design, this may be followed by CPU setup for another button, and not sleep. The CPU processing time can be seen in [Figure 6](#).

NOTE: CPU setup and processing times can be measured on a test MCU by toggling a GPIO pin at the beginning and end of each window, and observing the pin's status on an oscilloscope.

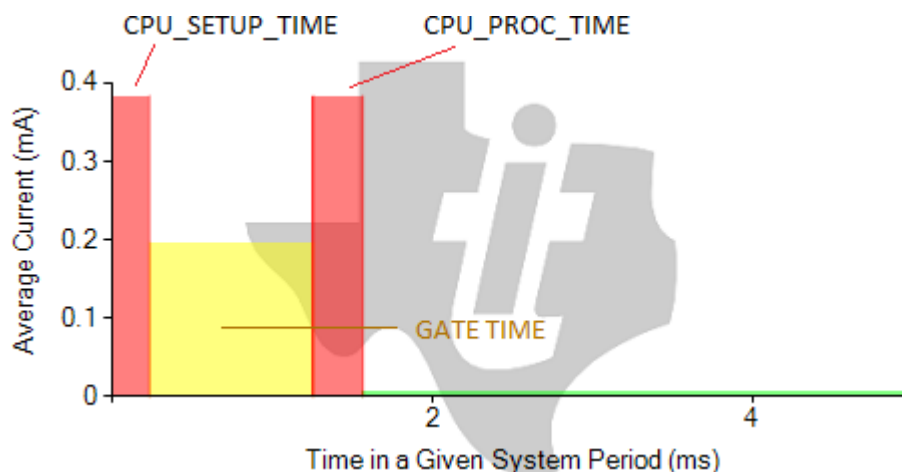


Figure 6. Example Timing Diagram

NOTE: If elements are grouped into a common sensor structure in the capacitive touch library, code execution is slightly more efficient and CPU times are slightly different. For example, three elements grouped into one sensor have a CPU active setup time at the beginning of the measurement period followed by the three gate times intermixed with three short CPU bursts to adjust the measurement to the next element. Then there is a long CPU active period during which all three keys are processed sequentially. The duty cycle looks slightly different for this configuration, and average power consumption is reduced slightly due to less software overhead. The power designer GUI only considers the case where each button is a separate sensor.

CPU times are all recorded in seconds (s).

A.2.4 Step 4: Identifying Test Voltages

Line 7 in the device file is always blank.

Line 8 in the device file always contains **SUPPLY_VOLTAGES** in the first cell.

The supply voltages the user would like to include for the device are to be listed below **SUPPLY_VOLTAGES**, starting on line 9. For each desired voltage, the value is repeated on the two lines below it (each voltage appears three times total, as in [Table 1](#)). This is to allow for the current in each mode (active, scan, and sleep) to be recorded at that voltage.

One cell to the right of the listed voltages (all identical), the three mode flags should be listed as seen in [Table 1](#). The mode flags in order are: **CPU_I**, **SCAN_I**, and **SLEEP_I**.

Additional voltages can be added in a descending fashion (the user is not limited to the four shown in the example in [Table 1](#)). If fewer than four are used, the extra lines should be removed from the file. The last line of the file should be the last line of the last voltage (the **SLEEP_I** line of the last voltage).

Voltages are all recorded in volts (V).

A.2.5 Step 5: Measuring Mode Currents

Now that supply voltages and frequencies have been identified, the supply current draw in each mode, for each frequency, and for each voltage needs to be measured and added to the file in the appropriate place. To measure the current in a given mode, a test MCU can be configured to run at the given frequency and voltage continuously running locked in the given mode.

If a voltage is specified to be measured at a certain frequency and the device does not support that frequency at the given voltage, an 'X' should be placed in the three corresponding cells for that voltage and frequency combination. This tells the GUI that the voltage does not support that frequency. See the device-specific data sheet for what frequencies are supported at different voltages.

To measure CPU Active current, or CPU_I in the device file, the library code should be run continuously on a test MCU and configured such that the electrode scan never actually takes place. As an alternative to modifying the library, a simple *while(1) {}* loop could be used to approximate CPU activity. Then a digital multi-meter (DMM) can be used to measure the CPU current.

To measure scan current, or SCAN_I, a test MCU should be configured to enable the scanning functionality on an electrode (for the PinOsc port functionality, a port pin should be connected to an electrode and configured for PinOsc) followed by going to LPM0 or LPM3 (depending on whether the SMCLK or ACLK is used during scanning). Then a DMM can be used to measure the scan current.

To measure sleep current, or SLEEP_I, a test MCU should be configured to go directly into its sleep mode (typically LPM3). Then a DMM can be used to measure the sleep current.

If hardware is not available for measurement, rough current estimates can be created by compiling information from the device datasheet.

Mode currents are all recorded in milliamps (mA).

A.2.6 Step 6: Saving the Template to a *.ctp File

After the device power profile template has been populated for the new device, the file should be saved with the *.ctp extension. If the file was modified in Microsoft Excel, then it should be exported as a comma-separated-value file (*.csv), then re-named to a *.ctp file). Then, the file may be added to the GUI through the steps in [Section A.1](#).

Appendix B Example Code to Reproduce Tool-Estimated Results

This appendix shows the source code for an MSP430G2553 attached to the MSP-EXP430G2 LaunchPad. This code should reproduce power performance as seen in the power estimator.

B.1 Sample: *main.c*

The main source file configures the MCU and controls when electrodes are scanned.

```
/*
 * Power Consumption Test Bench
 * Baseline Measurements
 * MCLK = DCO = 1MHz
 * SMCLK = Variable
 *
 * main.c
 *
 * Functionality:
 *     Start & Configure MCU
 *     Initialize CapTouch Tracking
 *     Utilize TimerA0 for CapTouch and TimerA1 for triggering scan/setting scan period
 *     Sit in LPM3 until Timer A1 triggers scan
 *
 * Texas Instruments, Inc.
 * MSP430 Capacitive Sensing Solutions
 * Fall 2012
 *
 * Walter Schnoor
 * Alan Manlick, Mgr.
 *
 */

/* TEST TOGGLES ARE BELOW */
// #define calibrateMode           // Define to tune thresholds and baselines

/***** Scan Period *****/
#define scanTime1                // [2.048ms Scan Time for Buttons]
// #define scanTime2                // [1.024ms Scan Time for Buttons]
// #define scanTime3                // [0.512ms Scan Time for Buttons]

/***** Number of Electrodes to Scan *****/
#define oneElectrode
// #define twoElectrode
// #define threeElectrode

/***** Sample Rate *****/
#define VLO_CNT                  11710           // VLO cycles in one second
#define sample1                  (VLO_CNT)       // 1Hz System Scan Rate
// #define sample2                  (VLO_CNT/2)    // 2Hz System Scan Rate
// #define sample5                  (VLO_CNT/5)    // 5Hz System Scan Rate
// #define sample10                 (VLO_CNT/10)   // 10Hz System Scan Rate
// #define sample20                 (VLO_CNT/20)   // 20Hz System Scan Rate
// #define sample50                 (VLO_CNT/50)   // 50Hz System Scan Rate

/***** Sample Period *****/
// #ifdef sample1
// #define sleepTime                sample1
// #endif
// #ifdef sample2
// #define sleepTime                sample2
// #endif
// #ifdef sample5
// #define sleepTime                sample5
```

```

#endif
#ifdef sample10
#define sleepTime    sample10
#endif
#ifdef sample20
#define sleepTime    sample20
#endif
#ifdef sample50
#define sleepTime    sample50
#endif

/* Includes */
#include "msp430.h"
#include "CTS_Layer.h"

/* Globals */
#ifdef calibrateMode
unsigned int raw[3];
#endif

unsigned char scanFlag = 0x00;    // Scan Trigger Flag, 0x00 = NO SCAN, 0x01 = SCAN

/* Prototypes */
void configAll(void);
void initializeCAPT(void);

/* Main Routine */
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;    // Stop WDT
    configAll();                // Configure MCU for testing
    initializeCAPT();            // Establish Captouch Baselines

#ifdef calibrateMode
    while (1) {
        TI_CAPT_Custom(&button1,&raw[0]);
        TI_CAPT_Custom(&button2,&raw[1]);
        TI_CAPT_Custom(&button3,&raw[2]);
        __no_operation();
        __delay_cycles(250000);
    }
#endif
    while(1) {
        if (scanFlag == 0x01) {
            scanFlag = 0;

#ifdef oneElectrode
            TI_CAPT_Button(&button1);
#endif
#ifdef twoElectrode
            TI_CAPT_Button(&button1);
            TI_CAPT_Button(&button2);
#endif
#ifdef threeElectrode
            TI_CAPT_Button(&button1);
            TI_CAPT_Button(&button2);
            TI_CAPT_Button(&button3);
#endif
            __bis_SR_register(LPM3_bits | GIE); // Enter LPM3 with interrupts
        }
    }
}

/* Functions */
void configAll(void) {
    // Configure Clocks
    BCSCCTL1 = CALBC1_1MHZ;    // Set range to 1MHz

```

```

        DCOCTL = CALDCO_1MHZ;          // Set DCO to 1MHz
        BCSCTL3 |= LFXT1S_2;           // LFXT1 = VLO

#ifdef scanTime1
        BCSCTL2 |= DIVS_2;             // Set SMCLK Divider to 4 (250kHz)
#endif
#ifdef scanTime2
        BCSCTL2 |= DIVS_1;             // Set SMCLK Divider to 2 (500kHz)
#endif
#ifdef scanTime3
        BCSCTL2 |= DIVS_0;             // Set SMCLK Divider to 1 (1MHz)
#endif

    // Configure IO Ports
    P1OUT = 0x00;
    P2OUT = 0x00;
    P1DIR = 0xFF;
    P2DIR = 0xFF;
    P1SEL = 0x00;
    P2SEL = 0x00;
    P1SEL2 = 0x00;
    P2SEL2 = 0x00;

    // Configure Timer
    TA1CTL = MC_1 | TACLK | TASSEL_1;   // Count Up to CCR0, Clear Reg, Use ACLK
    TA1CCR0 = sleepTime;                // Set Sleep Time in CCR
    TA1CCTL0 &= ~CCIFG;                // Clear Flag
    TA1CCTL0 |= CCIE;                  // Enable Interrupt
}

void initializeCAPT(void) {
    TI_CAPT_Init_Baseline(&button1);
    TI_CAPT_Init_Baseline(&button2);
    TI_CAPT_Init_Baseline(&button3);

    TI_CAPT_Update_Baseline(&button1,5);
    TI_CAPT_Update_Baseline(&button2,5);
    TI_CAPT_Update_Baseline(&button3,5);
}

/* Test ISR */
#pragma vector=TIMER1_A0_VECTOR
__interrupt void TIMER1_A0(void) {
    scanFlag = 0x01;
    __bic_SR_register_on_exit(LPM3_bits);
}

```

B.2 Sample structure.c

The structure.c file specifies the CapTouch hardware configuration and selects the RO_PinOsc HAL.

```

/*
 * Power Consumption Test Bench
 * Baseline Measurements
 * MCLK = DCO = 1MHz
 * SMCLK = Variable
 *
 * structure.c
 *
 * Texas Instruments, Inc.
 * MSP430 Capacitive Sensing Solutions
 * Fall 2012
 *
 * Walter Schnoor
 * Alan Manlick, Mgr.

```

```

*
*/

#include "structure.h"

//PinOsc P2.0
const struct Element element1 = {
    .inputPxselRegister = (unsigned char *)&P2SEL,
    .inputPxsel2Register = (unsigned char *)&P2SEL2,
    .inputBits = BIT0,
    .threshold = 50
};
//PinOsc P2.1
const struct Element element2 = {
    .inputPxselRegister = (unsigned char *)&P2SEL,
    .inputPxsel2Register = (unsigned char *)&P2SEL2,
    .inputBits = BIT1,
    .threshold = 50
};
//PinOsc P2.2
const struct Element element3 = {
    .inputPxselRegister = (unsigned char *)&P2SEL,
    .inputPxsel2Register = (unsigned char *)&P2SEL2,
    .inputBits = BIT2,
    .threshold = 50
};

/** Sensor *****/
// This defines the grouping of sensors, the method to measure change in
// capacitance, and the function of the group

const struct Sensor button1 =
{
    .halDefinition = RO_PINOSC_TA0_WDTP,
    .numElements = 1,
    .baseOffset = 0,
    // Pointer to elements
    .arrayPtr[0] = &element1,
    // Timer Information
    .measGateSource= GATE_WDT_SMCLK,           // 0->SMCLK, 1-> ACLK
    .accumulationCycles= WDTp_GATE_512        //512
};
const struct Sensor button2 =
{
    .halDefinition = RO_PINOSC_TA0_WDTP,
    .numElements = 1,
    .baseOffset = 1,
    // Pointer to elements
    .arrayPtr[0] = &element2,
    // Timer Information
    .measGateSource= GATE_WDT_SMCLK,           // 0->SMCLK, 1-> ACLK
    .accumulationCycles= WDTp_GATE_512        //512
};
const struct Sensor button3 =
{
    .halDefinition = RO_PINOSC_TA0_WDTP,
    .numElements = 1,
    .baseOffset = 2,
    // Pointer to elements
    .arrayPtr[0] = &element3,
    // Timer Information
    .measGateSource= GATE_WDT_SMCLK,           // 0->SMCLK, 1-> ACLK
    .accumulationCycles= WDTp_GATE_512        //512
};

```

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components 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 such information 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.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com