Application Report
SLAA668A–March 2015–Revised March 2017

Designing an Ultra-Low-Power (ULP) Application With SimpleLink™ MSP432™ Microcontrollers

Dung Dang — *MSP Applications*
Atul Lele — *Architect – Senior Member Technical Staff*
Evan Wakefield — *MSP Applications*

ABSTRACT

With the growing system complexity in ultra-low-power microcontroller (MCU) applications, minimizing the overall energy consumption is one of the most difficult problems to solve. Multiple aspects including silicon, other onboard hardware components, and application software must be considered. There are some obvious generic techniques that can be used to reduce energy consumption such as reducing operating voltage or frequency. Many of these generic techniques may not greatly reduce energy consumption independently, but taken as a whole, the results can be significant, as there are many interdependencies across these components.

This application report offers an overview of various low-power features of the industry's leading ultra-low-power microcontroller and, more importantly, how to exploit optimal combinations of these features to achieve the lowest energy consumption for a given application. This application report uses the industry-standard ULPBench™ benchmark from the Embedded Microprocessor Benchmark Consortium (EEMBC) as a case study to map individual and combinations of low-power features on the SimpleLink™ MSP432™ MCU to a relevant application scenario.

Contents

1 Power Optimization Vectors on a Microcontroller 2
2 Designing a ULP Application Using the SimpleLink MSP432 SDK 3
3 Power Management 3
4 Power System 4
5 Clock System 6
6 Memory Execution Optimization 7
7 Optimizing Power of an Entire MCU System 9
8 Summary 12
9 Other Methods to Optimize Device Power Consumption 12
10 Disclaimer 12
11 References 13

List of Figures

1 DC-DC and LDO Regulator: Features, Advantages, and Tradeoffs 5
2 MSP432P4xx ULPBench Results 10
3 MSP432P4xx ULPBench Results 10
4 MSP432P4xx ULPBench Results 11

List of Tables

1 LDO and DCDC Current Consumption 5
2 Effects of Increasing System Operating Frequency 9

Trademarks

1 Power Optimization Vectors on a Microcontroller

SimpleLink MSP432 microcontrollers integrate a number of power enhancements that include lower active power consumption, low-power (LP) mode current consumption, and more efficient peripherals with lower current consumption. In addition, the MSP432 devices also provide a number of options and power configurations that enable developers to further optimize the power consumption for a specific MSP432 application. Nonetheless, for any given microcontroller, there are a number of vectors that can be used to optimize the power consumption of the device.

[Vector 1] Reduce operating voltage

Power is the product of voltage and current. For a given application system that consumes a certain amount of current, lower supply voltage can help reduce the power consumption, with the constraint that minimum voltage requirement is met. This requirement can sometimes be related to the microcontroller supply voltage level itself (for MSP432P4xx minimum V_{CC} = 1.62 V) or the setting the internal voltage regulator to appropriate voltage level. It could be dependent on the CPU operating frequency or a minimum voltage required by a certain peripheral to be fully functional. For example, if a 2.5-V reference voltage is required by the analog-to-digital converter (ADC) in the application, supply voltage must be greater than or equal to 2.5 V.

[Vector 2] Reduce the operating frequency

As can be deduced from any microcontroller data sheet, power and current consumption are directly proportional to the operation frequency. In many scenarios, higher operating frequency means that the CPU can execute code and complete the task faster. However, in certain real-time scenarios, many application activities are timing-dependent or event-driven. Having the CPU running faster in an idle loop waiting for a certain event to trigger or waiting for serial data to come in at a lower baud rate can consume additional power that can be saved. For these scenarios, either putting the device into a low-power mode or reducing the operating frequency might be worth investigating.

[Vector 3] Maximize sleep time

A typical low-power embedded application spends most of its lifetime in two modes: active mode executing meaningful tasks and low-power mode with minimal activity other than time-keeping or waiting for an interrupt or event to wake up to active mode. Low-power mode consumes much less current than active mode—on the MSP432P4xx family, LPM3 current can be as low as approximately 700 nA while active mode current can be up to several milliamps. Minimizing active mode time and maximizing time spent in low-power modes can significantly reduce the overall current consumption. This can usually be achieved by minimizing the active mode task, optimizing the active mode code, or increasing the operating speed in active mode.

[Vector 4] Minimize transition time

In addition to time allocated for active and low-power modes, some applications might unknowingly spend a considerable amount of time transitioning between these power modes. If the transitions times go unnoticed, they might contribute to the total energy consumption increase. One of the power optimization activities is to identify all of the system transitions and determine if any of them can be reduced or removed.

[Vector 5] Solving intermodule dependencies

The previous vectors are all possible options to optimize individually. However, on a given microcontroller platform, these vectors might have some interdependencies, or optimizing on a particular vector might negatively affect another. For example, reducing operating frequency could potentially increase active duty cycle. Therefore, it is even more important to take into account the intermodule dependencies and determine optimal combinations of settings for a given platform.

The remaining sections of this application report describe the system peripherals of MSP432 MCUs, identify critical features and options, correlate them back to one or more of the five vectors mentioned above, and determine how they can help to optimize power and energy consumption. The key core peripherals on MSP432P4xx devices include: power system [Power Controller Module (PCM)], clock system, and memory system (flash, SRAM, and ROM). More importantly, the intermodule dependencies specific to MSP432 are discussed to help show the combinations that makes sense for an MSP432 application.

2 Designing a ULP Application Using the SimpleLink MSP432 SDK

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 an ultimate goal of 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 (SDK) lets you reuse often, opening the door to create unlimited applications. For more information, visit www.ti.com/simplelink.

Inside of the SimpleLink MSP432 SDK, there are a variety of software tools and libraries that can be leveraged to automatically integrate the power optimization vectors discussed in the previous section as well as bring other benefits.

3 Power Management

Power management offers significant extension of the time that batteries used to power an embedded application last. However, the application, operating system, and peripheral drivers can be adversely affected if dynamic power-saving transitions occur when they are performing important operations. To manage such effects, it is useful to provide power management capabilities for these components to coordinate and safely manage the transitions to and from power saving states. The SimpleLink MCU SDK includes a Power Manager framework that supports the CC13xx/CC26xx, CC32xx, and MSP432 devices. The same top-level APIs, concepts, and conventions are used for all three MCU families.

The SimpleLink MSP432 SDK takes an integrated approach at to power management using TI-RTOS and FreeRTOS. Enabling the Power Manager and using TI Drivers results in automatic power savings when the processor is idle. Application developers do not need to write power management code; it is available by default.

Applications that include the Power Manager framework reduce their power usage because a target-specific power policy is invoked during application idle time to make informed decisions about when to activate and maximize power savings. These target-specific power policies understand the reduced power states available on the target. The TI Drivers communicate with the Power Manager to enable and disable peripheral resources and transitions.

From an application development perspective, minimal or no application code is necessary to manage power usage. However, APIs are available for the application to customize its power usage as desired. For more information on the Power Manager API, see *SDK Power Management: MSP432, CC13xx/CC26xx, and CC32xx SimpleLink MCUs*.

3.1 Power Policies

For the MSP432, the SimpleLink SDK provides two power policies. Within these power policies, there are a variety of performance profiles that developers can select from that have a range of clock frequencies, power settings (LDO vs DC-DC operation), and flash options that help reduce power consumption. If developers want more control over the power policies to factor in application-specific information into the decision making process, they can create their own power policy. This technique is described in Section 4.6 of *SDK Power Management: MSP432, CC13xx/CC26xx, and CC32xx SimpleLink MCUs*.

3.2 Performance Levels

Within each power policy, there is an MSP432 performance level. There are four performance levels that come predefined in the MSP432Power TI Driver. These performance levels can be found in the SimpleLink SDK documentation overview on TI Resource Explorer. From there, go to MSP432Power.h, scroll down to PowerMSP432_PerfLevel, and see how you can leverage the four existing performance levels or create your own custom performance levels.

4 Power System

Behind the scenes of an RTOS that incorporates the Power Manager API is the power system of the MSP432 MCU. Many MCUs scale performance with voltage; as the voltage drops, the MCU core operates at a reduced frequency. This can increase energy consumption if the MCU must stay awake longer to perform an equivalent amount of work. With MSP432 MCUs, however, the core is able to operate at relatively good speed at the lowest voltage. Thus, operating at a lower voltage gives a true improvement in energy efficiency. [Vector 1]

4.1 CPU and Digital Logic Core Voltage

The MSP432P4xx family of devices requires a secondary core voltage (V_{CORE}) for its internal digital operation in addition to the primary one applied to the device (V_{CC}). In general, V_{CORE} supplies the CPU, memories (flash and RAM), and other digital modules, while V_{CC} (connected to the DVCC and AVCC pins of the device) supplies I/Os and all analog modules (including the oscillators). The V_{CORE} output is maintained using a dedicated voltage regulator. V_{CORE} is programmable with two predefined voltage levels, and each level is restricted to particular maximum operating frequency. This provides programmable dynamic voltage frequency scaling (DVFS) operation.

This directly correlates to [Vector 1], because the digital logic of the system is now supplied with a lower voltage, it consumes less power than on a microcontroller where the entire system operates out of the higher V_{CC} supply rail.

Furthermore, if the application operating frequency does not require the higher V_{CORE} voltage, reducing the V_{CORE} voltage level further reduces the power consumption of the device. For example, specifically for MSP432P4xx family, if maximum operating clock frequency is ≤24 MHz, the VCORE0 option can be chosen, which is 1.2 V (typical) compared to VCORE1 which is 1.4 V (typical). VCORE0 provides better power consumption at a given frequency (≤24 MHz) than VCORE1.

Developers can programmatically control the VCORE level using the Power TI Drivers in the SimpleLink SDK. To learn how to use VCORE, see the documentation referenced in Section 3.2.

4.2 Regulators: LDO and DC-DC

To generate the secondary core voltage (V_{CORE}) from the primary voltage (DVCC), the MSP432P4xx family uses two regulators: a low-dropout voltage regulator (LDO) as the default regulator, and an inductor-based DC-to-DC step-down switching regulator (DC-DC) as a secondary or optional one. The LDO and DC-DC are connected in parallel. The two modules share a common output (the digital supply VCORE rail) and a common reference or target voltage.

Developers can programmatically control whether to use the DC-DC or LDO regulator using the Power TI Drivers in the SimpleLink SDK. To learn how to use this feature, see the documentation referenced in Section 3.2.

For the majority of microcontrollers with multiple voltage rails (V_{CC} and V_{CORE}), the LDO regulator is typically used due to a number of advantages such as cost-effective, relatively noise-free output due to no switching component, faster to ramp up and down from low-power modes, among others. These advantages might be critical for certain applications with stringent requirements related analog or RF performance or cost-sensitivity. However, the LDO regulator might not be ideal for power savings, because current drawn by supply is same as that of the current drawn from LDO: $I_{in} = I_{load} = I_{vcore}$.

On the other hand, the DC-DC regulator's main advantage is the significant power saving compared to the LDO. Unlike a linear regulator, which generates a voltage rail from another by dropping the extra voltage across a linear, effectively passive, element (thereby wasting power equivalent to the voltage dropped multiplied by the load current drawn), the DC-DC efficiently generates one voltage rail from another using active (inductor and capacitor) elements. In an ideal DC-DC, power drawn from the output rail

(V_{CORE} × I_{load}) equals the power drawn from the input rail (V_{CC} × I_{in}). I_{in} is therefore smaller than I_{load} by the fraction V_{CORE} / V_{CC}. Note that this gives additional advantage if VCORE0 level is chosen compared to VCORE1 as described in the previous section. After accounting for efficiency losses, the DC-DC typically has efficiency in the range of 75% to 90% for moderate to maximum loads. This allows the DC-DC regulator to provide significant power saving of up to 45% compared to the LDO regulator. Table 1 shows relative current consumption comparison between the DC-DC and LDO when the system operates at 24 MHz and 48 MHz.

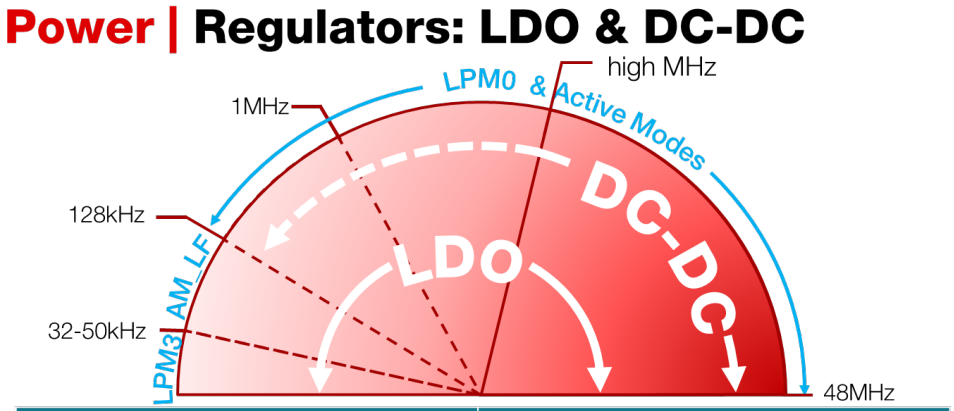
Table 1. LDO and DCDC Current Consumption

Regulator	CPU Frequency = 24 MHz	CPU Frequency = 48 MHz
LDO	3270 µA	6760 µA
DC-DC	1920 µA	4270 µA

While the DC-DC power saving for active mode is apparent, the LDO's advantages and flexibility might be required by the application in certain scenarios, as explained previously. The LDO is also the default regulator that is used upon device startup or after returning to active mode from low-power mode. So any DC-DC regulator usage should also take into account the transition time from LDO to DC-DC as well as DC-DC back to LDO before going back to low-power mode.

There are several additional advantages and drawbacks for both regulators. Figure 1 summarizes the advantages and disadvantages of the LDO and DC-DC to help developers determine how and when it is possible to use the DC-DC regulator to maximize power savings.

Typically, developers must choose between using either an LDO or DC-DC convertor for all use cases. By providing two options for power regulation, developers can dynamically optimize regulation based on the current mode of operation. When the system is in a standby mode of operation, for example, the LDO can be used to minimize wake time. For operating modes or use cases where active current plays a larger role in power consumption, the DC-DC convertor can be used. For more details on DC-DC usage as well in-depth analysis on its advantages and tradeoffs, see *Maximizing MSP432P4xx Voltage Regulator Efficiency: DC-DC and LDO Features and Tradeoffs*.



LDO	DC-DC
Default regulator at startup	Secondary, requires external inductor
VCC = [1.62V-3.7V]	VCC = [2.0V-3.7V]
Available in all power modes	Available in LPM0 & Active Modes
Flexible with scalable output loads for low power modes	Efficient, optimized for high-speed/high-load operations
Fast on/off switching operations	Slow on/off/failsafe switching from/to LDO

Figure 1. DC-DC and LDO Regulator: Features, Advantages, and Tradeoffs

4.3 Low-Power Modes

The MSP432P4xx family provides a number of low-power operating modes that originated in the MSP430™ devices including LPM3, LPM4, LPM3.5, and LPM4.5. Similar to earlier MSP devices, these low-power modes on MSP432P4xx consume extremely little power, putting the MSP432P4xx in a great position for power saving.

In addition to the traditional MSP low-power modes, the MSP432P4xx family introduce another class of low-power modes that can provide additional power saving: low-frequency power modes. The low-frequency modes are special low-power low-frequency options that can be used in conjunction with both active and LPM0 modes. In these low-frequency modes, memory and peripherals can execute at maximum speed of 128 kHz. Because the regulator can reduce the drive strength to supply minimal current to drive the entire low-frequency system, the device's total current consumption can be as little as <80 µA when using these modes.

See *Leveraging Low-Frequency Power Modes on SimpleLink™ MSP432P4xx Microcontrollers* for more details on these special power modes.

5 Clock System

As described in Section 1, power and current consumption is directly proportional to the frequency of operation (f). Naturally if (f) is reduced, current consumption also reduces linearly. But, reducing frequency may reduce overall system throughput or may increase the overall energy consumption. The reason is that the device may have to be in an active operating mode (AM_VCORE0 or AM_VCORE1) for more time and in low-power mode for less time. Always operating at high frequency to complete an active mode task also may not help, because of the peak current requirements during such short bursts, which in turn can affect the battery capacity and hence battery life. Therefore, the optimal selection of clock frequency for system operation is critical to optimize the energy consumption of the system.

The Clock System (CS) provides options such as choices of different oscillators as different clock sources, dynamic selection, and clock dividers to enable optimal power consumption.

Developers can programmatically control the clock system using the Power TI Drivers in the SimpleLink SDK. To learn how to use this feature, see the documentation referenced in Section 3.2.

The digitally controlled oscillator (DCO) provides options such as changing frequency on the fly and fine tuning to an intermediate frequency between 0.98 MHz to 52 MHz, minimal current consumption of approximately 150 µA at 24 MHz and 200 µA at 48 MHz. The maximum DCO clock frequency drift with temperature in external resistor mode is ±35 ppm/°C, and the drift for internal resistor mode is ±250 ppm/°C. An application can program the DCO for the best combination of optimal power consumption and required accuracy. If higher accuracy is needed, the application needs to use the on-chip high-frequency crystal oscillator (HFXT) with the trade-off being higher current consumption.

The CS also provides flexible clock source and frequency selection for different on-chip clock sources (MCLK, SMCLK, and ACLK). It can also divide the clocks dynamically, and all of this can be done with one control register (CSCTL1).

For example, DCO can be used as the main clock source for MCLK and SMCLK, HSMCLK. LFXT can be used as the clock source for RTC. For slower peripherals such as I^2C, the DIVS and DIVHS register bits can be configured to select slower source clock frequency. For ADC, MODOSC can be used as a quick ON/OFF oscillator that remains enabled only when ADC conversion operation is ongoing.

The ULPBench example in Section 7 uses the DCO as the main clock source at 16 MHz instead of at 48 MHz for optimal current consumption.

For more details, see the CS sections in the following documents:

- *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual*
- *MSP432P401R, MSP432P401M SimpleLink™ Mixed-Signal Microcontrollers*
- *Multi-Frequency Range and Tunable DCO on MSP432P4xx Microcontrollers*

6 Memory Execution Optimization

Although typical code execution involves opcode and literal fetches from memory, it may also involve continuous and intermittent data writes to and reads from stack (SRAM), function calls from software library provided in flash or ROM, data writes to or reads from peripherals, DMA data transfers, and other operations. A significant portion of the power consumption in a typical MCU originates from code execution from a nonvolatile memory such as flash. Hence, it is beneficial to optimize memory accesses to reduce as much power consumption as possible during code execution.

6.1 Flash

MSP432P401x MCUs have flash as the main nonvolatile memory for application program code and data.

6.1.1 Wait State and Clock Dependency

Low-power flash memory is typically slower than the high-frequency clocks used at system level; for example, for the CPU or DMA. Therefore, at higher operating frequency, flash memory requires wait-stated accesses, which means that the CPU is halted for certain number of clock cycles depending on the CPU-to-flash frequency ratio. See the device data sheet for details.

Developers can programmatically control how many wait states they need using the Power TI Drivers in the SimpleLink SDK. To learn how to use this feature, see the documentation referenced in Section 3.2.

6.1.2 Data and Instruction Buffers

To offer optimal power consumption and performance across predominantly contiguous memory accesses, the flash controller offers a read buffering feature. Developers can programmatically control the read buffer feature using the Power TI Drivers in the SimpleLink SDK. To learn how to use this feature, see the documentation referenced in Section 3.2.

If read buffering is enabled, the flash memory is always read in entire 128-bit chunks, even though the read access may only be 8, 16, or 32 bits wide. The 128-bit data and its associated address is buffered by the flash controller, so that subsequent accesses (expected to be contiguous in nature) within the same 128-bit address boundary are serviced by the buffer. Using this scheme, the flash accesses have wait states only when the 128-bit boundary is crossed, while read accesses within the buffer's range are serviced without any bus stalls. If read buffering is disabled, accesses to the flash bypasses the buffer and the data read from the flash is limited to the width of the access (8, 16, or 32 bits). Each bank has independent settings for the read buffer. In addition, within each bank, the application has the flexibility of enabling read buffering either for instruction fetches only, for data fetches only, or for both.

6.1.3 Flash Power Consumption

Flash power consumption is dependent on the number of bits read. If 128 bits are read, power consumed is higher than if 32 bits are read but not four times the power of 32-bit read power. So for contiguous or linear code execution or data fetches, TI recommends enabling the buffer to enhance performance and reduce power consumption.

6.1.4 What Options to Choose for Code Execution From Flash

- Enabling buffer definitely gives performance advantage but whether it can help for getting better power consumption or not depends on the linear or nonlinear execution of the code or data reads. If an application needs higher throughput always at the cost of current consumption, buffers can always be enabled. With wait-states >0 configured for flash, buffers are recommended to be always enabled for higher throughput.
- In general, TI recommends enabling instruction and data buffers if the code execution or data reads happen in linear fashion to prevent the controller from constantly accessing flash and ultimately reduce overall current consumption (see Section 6.1.2).
  - Specific example is when the code is running with 0-wait states, there is no performance or throughput advantage. In that case, the code profile really determines whether to enable the buffer (with a power advantage for linear code execution) or to disable the buffer.

- Because the buffer configuration is dynamic, it can be enabled or disabled for specific code sections or function calls. If there is a function with significant size with mostly linear code execution, TI recommends enabling and disabling the buffer dynamically before and after the function call respectively.
- It is also recommended to check the current consumption for your specific application and determine specific combinations of settings that work best for that scenario. MSP432P4xx family of devices support EnergyTrace™ technology, which can enable developers to construct a detailed energy profile of the system and correlate back to areas in the code with high contribution to energy consumption. Based on the findings, developers can leverage capabilities provided by MSP432P401xx to dynamically change wait-states, enabling/disabling buffers which can be chosen to cater to specific application needs.
- See ULPBench results for an example analysis on how and when to use flash buffer.

6.1.5 Flash Access Benchmark Registers (*BMRK* Registers in Flash Controller)

The flash controller offers two counters for application benchmarking purposes. This feature is extremely useful for monitoring the number of flash accesses. Using these counts, the application can manage the optimal wait-states and the buffer enable or disable settings as discussed in previous sections.

- Instruction fetch benchmark counter (32 bits) – Increments on each instruction fetch to the flash
- Data fetch benchmark counter (32 bits) – Increments on each data fetch to the flash

6.2 SRAM

SRAMs are standard read-write memories on MCUs. Particularly on MSP432P401xx, SRAMs work at the same speed as the CPU clock frequency so code execution from SRAM clearly gives a performance and throughput advantage compared to code execution from flash. There is a power consumption advantage for SRAM compared to flash as well. Power consumption (commonly measured in µA/MHz) for SRAM is significantly lower than the µA/MHz consumption of flash memory. Therefore, TI recommends executing small loops or functions that are frequently used from SRAM instead of flash for performance and power benefits.

This may be handled by using a smart function to copy a function or subset of code from flash to SRAM and execute from SRAM whenever needed. This SRAM bank can be kept enabled during LPM3 mode so that the contents are retained and copying the same function over is not needed after LPM3 wakeup.

SRAM retention

- **SRAM_BANKRET** in the SYS module controls the retention of SRAM contents in LPM3 mode. Depending on the application, either ALL or a FEW banks are programmed to be retained in LPM3 mode which can affect the current consumption in LPM3.

6.3 ROM

In addition to flash, the MSP432P4xx also provides read-only memory (ROM) as a second nonvolatile memory option. As the name indicates, the ROM content is programmed when the device is manufactured, and after that memory contents can only be read or executed but not modified. MSP432P4xx provides 32Kbytes of on-chip ROM including the built-in peripheral driver library (DriverLib) application program interfaces (APIs) to provide developers with a highly-abstracted set of APIs to exercise various functionalities of the MSP432P4xx peripherals. The easy-to-understand ROM DriverLib APIs allow developers to speed up the process of learning how to configure peripheral registers, change power modes, and other standard tasks.

ROM DriverLib offers a number of benefits. First, the DriverLib in ROM is thoroughly tested and optimized, and it represents the TI recommended procedures to exercise the peripherals. In the context of this application report, executing out of ROM yields both higher performance (0 wait state access to ROM) and better power consumption (much lower than flash execution and a little higher than SRAM due to size differences—32KB of ROM bank compared to 8KB of SRAM bank). Both of these performance increases, faster execution time, and lower power consumption directly result in minimizing the energy consumption of the device. [Vector 3]

Maximizing the driver library use helps to reduce flash memory accesses and enable better power consumption.

Other than the power benefit, there is an indirect benefit of using software driver library. Leveraging DriverLib in ROM also frees valuable flash memory space for application code. Shifting this additional approximately 31Kbytes of memory towards the high-level application code for intelligent software to uniquely differentiate the application is a much better allocation than taking up the memory for routine peripheral configuration tasks that, if implemented perfectly, would offer only marginal improvement over the TI-provided ROM solution.

7 Optimizing Power of an Entire MCU System

Now that we have examined several modules within the MSP432 microcontroller system and how they can be configured to optimize and reduce power consumption, let's take a look at how these components can be used together and any potential dependencies or constraints they might have on each other. ULPBench by EEMBC is used as a case study to understand possible trade-offs.

7.1 Is Faster Always Better?

The MSP432 microcontroller as a whole is a heavily clock-gated system, so the CPU and operating clock frequency can ultimately have a significant effect on not only the clock power consumption but also on how much power required to sustain other peripherals. Multiple peripherals' power consumption can be throttled or boosted depending on the CPU frequency requirement.

As shown in Table 2, flash memory operation with 0 wait states is limited to 16 MHz in AM_*_VCORE0 mode and 24 MHz in AM_*_VCORE1 mode. Flash memory operation with 1 wait state is limited to 24 MHz in AM_*_VCORE0 mode and 48 MHz in AM_*_VCORE1 mode. Running the CPU at any frequency with more than 0 flash wait-states stalls the CPU for number of clock cycles equal to the number of wait-states configured until the flash memory content can be completely fetched.

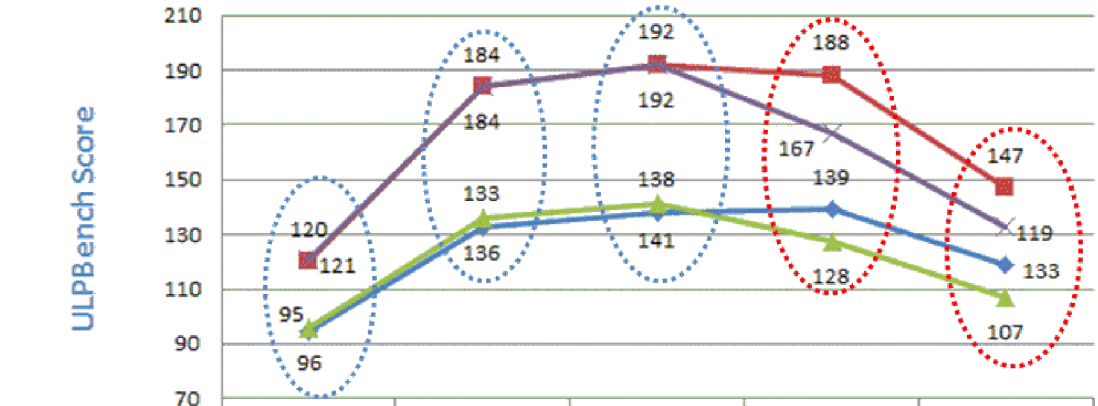
Combining these dependencies, Table 2 summarizes the effects of increasing the system operating frequency.

Table 2. Effects of Increasing System Operating Frequency

Number of Flash Wait States (WS)	Maximum MCLK Frequency (MHz)	
	AM_LDO_VCORE0, AM_DCDC_VCORE0	AM_LDO_VCORE1, AM_DCDC_VCORE1
0	16	24
1	24	48

7.2 Speed Does Not Always Improve Energy Efficiency

Running the CPU at 48 MHz in can result in a lower ULPMark score than running it at 24 MHz. At 48 MHz, workload is definitely completed faster than at 24 MHz, but core voltage must be increased, and there are mode transition latencies involved that can result in lower ULPMark scores. This is described in the following sections as part of the effect due to transition latencies. Note that this is true for example such as ULPBench. There may be other algorithms where data processing from SRAM by CPU may be done more frequently and performance at 48 MHz would be better. In Figure 2, 3-, 12-, and 16-MHz scores are with 0 wait states, and 24- and 48-MHz scores are with 1 wait state. The 48-MHz score is with VCORE1 setting. BUF_EN or BUF_DIS indicate that the 128-bit prefetch buffer is enabled or disabled, respectively.



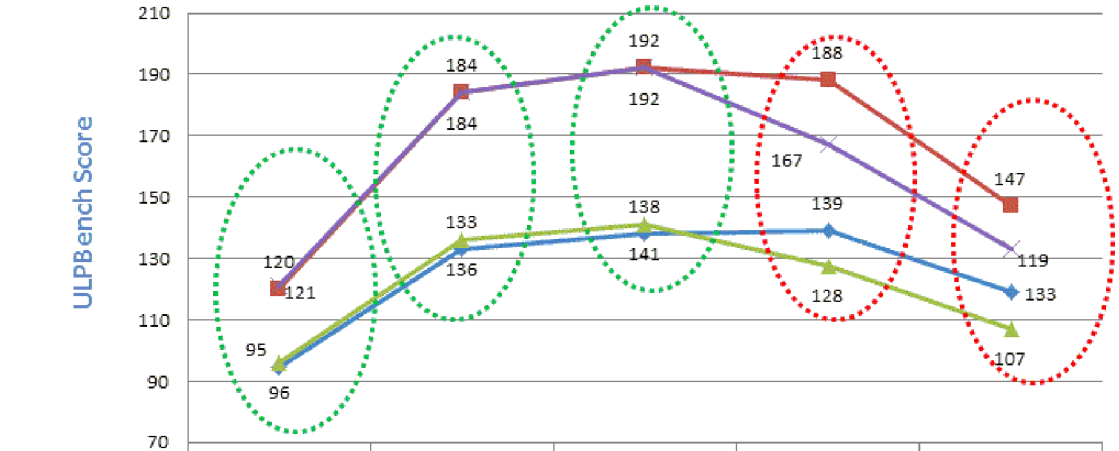
	3MHz	12MHz	16MHz	24MHz	48MHz
LDO_BUF_EN	95	133	138	139	119
DCDC_BUF_EN	120	184	192	188	147
LDO_BUF_DIS	96	136	141	128	107
DCDC_BUF_DIS	121	184	192	167	133

Figure 2. MSP432P4xx ULPBench Results

7.3 Efficiency of Flash Instruction and Data Buffers and Correlation to Wait States

128-bit instruction and data buffers for flash reads can be used for better throughput and power consumption if code execution is linear. If execution is not linear, there may be overhead due to reading 128 bits by using more power and discarding some of the words. MSP432P4xx provides flexibility to enable or disable the buffers dynamically, and software can optimally use this feature depending on the linear or nonlinear nature of the code execution (see Figure 3).

Nonlinearity of the code can be seen for two different cases. First is when the code execution happens with 0 wait states, where the scores are similar. Second is when code execution happens with 1 wait state, where it is clearly seen from the results that the code execution with buffer enabled improves power and performance and, hence, overall energy efficiency.



	3MHz	12MHz	16MHz	24MHz	48MHz
LDO_BUF_EN	95	133	138	139	119
DCDC_BUF_EN	120	184	192	188	147
LDO_BUF_DIS	96	136	141	128	107
DCDC_BUF_DIS	121	184	192	167	133

Figure 3. MSP432P4xx ULPBench Results

7.4 V_{CORE} Switching Latency Diminishes Energy Saving

Code execution of ULPBench workload function during active mode of MSP432P4xx takes insignificant number of cycles (<0.06%) with high optimization of IAR compiler at 16 MHz, 0-wait state condition compared the time spent in LPM3 mode (approximately 99.94%). So any additional time it spends in active mode can severely hamper the ULPBench score.

For example, if the lowest power consumption is needed in LPM3 mode with VCORE0 setting, and the highest performance is needed at 48 MHz, then there can be many mode transitions involved as per the guidelines in the *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual*.

AM_LDO_VCORE0 → AM_LDO_VCORE1 → AM_DCDC_VCORE1 → AM_LDO_VCORE1 → AM_LDO_VCORE0 → LPM3_VCORE0

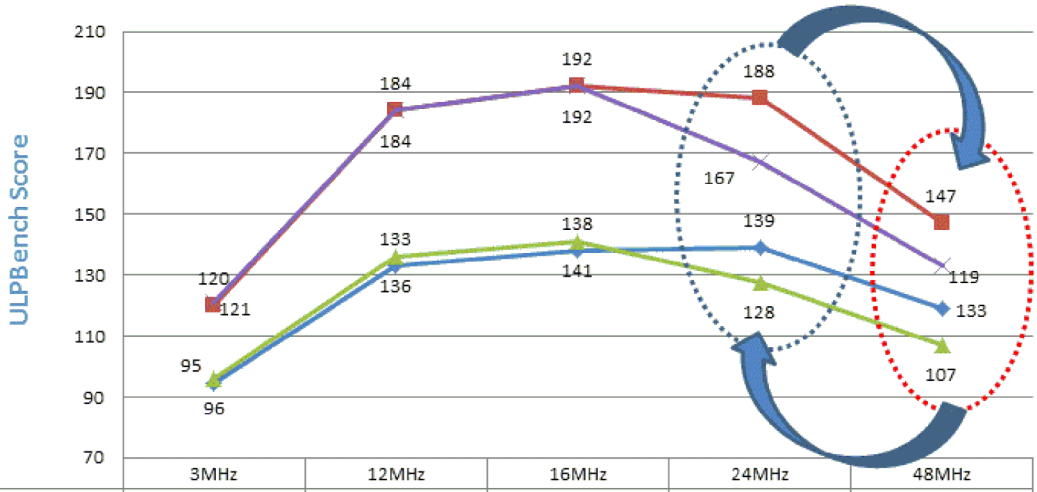
For example, additional transitions can be avoided if higher performance is needed as soon as the device wakes up from LPM3 mode.

AM_LDO_VCORE0 → AM_LDO_VCORE1 → AM_DCDC_VCORE1 → AM_LDO_VCORE1 → LPM3_VCORE1

In both the cases, transition time can be significant and can affect the duration for which the device is in active mode, which affects the ULPBench score. So it is important to understand four aspects from the application's standpoint.

- Duration spent in active operational mode by an application
- Duration spent in low-power mode by an application
- Duration spent in transitioning between different active modes
- Duration spent in transitioning between active and low-power modes

The transition function to change from LPM3 to active mode is done to initiate the mode transition and continue with the rest of the code execution without waiting for PMR_BUSY flag to be set. This helps to continue with code execution in LDO mode until LDO-to-DCDC transition is complete. This saves time waiting for the transition to finish.



	3MHz	12MHz	16MHz	24MHz	48MHz
LDO_BUF_EN	95	133	138	139	119
DCDC_BUF_EN	120	184	192	188	147
LDO_BUF_DIS	96	136	141	128	107
DCDC_BUF_DIS	121	184	192	167	133

Figure 4. MSP432P4xx ULPBench Results

8 Summary

Section 1 described generic vectors to reduce power consumption on a generic microcontroller design, and the rest of the sections described how these can be easily achieved on the SimpleLink MSP432 platform:

- Choose the right combination of CPU frequency and flash wait states for optimal performance. See the data sheet for additional information.
- Use features that allow 128-bit instruction and data buffer in flash controller to be enabled and disabled dynamically according to linear or nonlinear nature of code execution from flash memory. This can affect performance and power consumption.
- Choose optimal operating voltage point to balance performance and power consumption
- Optimize different mode transition latencies by minimizing the transitions depending on the time duration spent in each mode

9 Other Methods to Optimize Device Power Consumption

9.1 Optimize Device Power Consumption

- Terminate and configure I/Os as soon as possible after application begins. By default, all of the I/Os are in input mode. Depending on the application, relevant I/Os should be driven externally, should be internally or externally pulled up or down, or should be programed in output mode with value driven as 0 or 1.
- Use smart module enable and disable features to consume power only when module is needed to be active. For example, use the UCSWRST bit in eUSCIA and eUSCIB modules to keep the module in reset or active.
- Use an optimal clock source. MODOSC or SYSOSC are also automatically enabled to provide MODCLK or SYSCLK to ADC14 when needed and are disabled when not needed for ADC14 or for rest of the device. Use these as ADC14 clock sources for optimal ADC operation.

9.2 Leverage Other MSP432 ULP Tools

The MSP432 microcontroller ecosystem also provides a unique and powerful set of ultra-low-power tools and software that developers can use to further optimize the application power consumption. Starting with a well-designed embedded systems using silicon and hardware knowledge such as from this application report, embedded developers can then take advantage of static code analyzer tool such as ULP Advisor to further identify areas in the code to optimize for execution time and to reduce power consumption. Next, during debug phase, developers can use EnergyTrace technology to construct an energy profile of the application and correlate critical energy consumption spikes to areas in the code. This allows developers to identify high-energy components in the configuration or software for further power improvement.

10 Disclaimer

- ULPBench measurements, current consumption measurements were done on limited number of units which were from a specific manufacturing lot. Results measured on a LaunchPad™ development kit or with different units may vary.
- External board components can also affect current consumption. For example, LFXT current consumption can change significantly depending on the external capacitors.
- See the electrical specification in the data sheet for various parameters and their MIN, TYP, and MAX values.

11 References

1. *Multi-Frequency Range and Tunable DCO on MSP432P401xx Microcontrollers*
2. *Maximizing MSP432P4xx Voltage Regulator Efficiency*
3. *Getting Started With EEMBC ULPBench™ on MSP-EXP432P401R*
4. *Leveraging Low-Frequency Power Modes on SimpleLink™ MSP432P4xx Microcontrollers*
5. *MSP432P4xx SimpleLink™ Microcontrollers Technical Reference Manual*
6. *MSP432P401R, MSP432P401M SimpleLink™ Mixed-Signal Microcontrollers*
7. EnergyTrace Technology
8. ULP Advisor

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from March 19, 2015 to March 7, 2017 **Page**

- Added to list of authors 1
- Added "SimpleLink" branding and updated titles of referenced document as necessary 1
- Changed 750 nA to 700 nA under "[Vector 3] Maximize sleep time" in Section 1, *Power Optimization Vectors on a Microcontroller* 2
- Added Section 2, *Designing a ULP Application Using the SimpleLink MSP432 SDK* 3
- Added Section 3, *Power Management* 3
- Added the paragraph that starts "Developers can programmatically control the VCORE level..." in Section 4.1, *CPU and Digital Logic Core Voltage* 4
- Added the paragraph that starts "Developers can programmatically control whether to use the DC-DC or LDO..." in Section 4.2, *Regulators: LDO and DCDC* 4
- Updated current values in Table 1, *LDO and DCDC Current Consumption* 5
- Added the paragraph that starts "Developers can programmatically control the clock system..." in Section 5, *Clock System* 6
- Updated the current values in the paragraph that starts "The digitally controlled oscillator (DCO) provides..." in Section 5, *Clock System* 6
- Changed "24 MHz" to "16 MHz" in the paragraph that starts "The ULPBench example..." in Section 5, *Clock System* 6
- Added the paragraph that starts "Developers can programmatically control how many wait states..." in Section 6.1.1, *Wait State and Clock Dependency* 7
- Added the paragraph that starts "To offer optimal power consumption and performance..." in Section 6.1.2, *Data and Instruction Buffers* 7
- Updated the paragraph that starts "As shown in Table 2, flash memory operation..." in Section 7.1, *Is Faster Always Better?* 9
- Updated the first paragraph in Section 7.2, *Speed Does Not Always Improve Energy Efficiency* 9
- Replaced the second paragraph in Section 7.3, *Efficiency of Flash Instruction and Data Buffers and Correlation to Wait States* 10
- Replaced the paragraph that now starts "The transition function to change from LPM3..." in Section 7.4, V_{CORE} *Switching Latency Diminishes Energy Saving* 11
- Added to Section 11, *References* 13

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