Calculation of TMS320C2xx Power Dissipation

Application Report

Clay Turner
Digital Signal Processing Solutions – Semiconductor Group

SPRA088
October 1996

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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 of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

Appendixes

List of Illustrations

INTRODUCTION

The TMS320C2xx devices are a family of 16-bit fixed-point processors with enhanced processing capability over the previous TMS320C2x family of digital signal processors (DSPs). Architecture and design enhancements have produced a generation of processors that yield performance far beyond the previous generation of DSP devices while maintaining low power dissipation and low cost.

The TMS320C2xx family of DSPs are capable of processing speeds as high as 40 million instructions per second (MIPS) in order to handle a wide variety of high-performance applications. In addition to its superior performance, the device is designed to exhibit very low power dissipation, and features flexible power management modes which allow further reduction in power requirements.

The static CMOS technology used in the fabrication of the TMS320C2xx family of devices combines high density with low power dissipation. Because CMOS devices ideally draw current only when switching, this technology offers the potential for fully static devices with standby modes exhibiting very low current drain. These characteristics make the TMS320C2xx devices uniquely well-suited to portable power-sensitive and battery-operated applications such as digital cellular telephones, laptop modems, voicemail pagers, and so on.

Typical active current requirements for the TMS320C2xx are 1.9 mA/MIPS for 5-V operation, and 1.1 mA/MIPS for 3-V operation. These characteristics are further improved with the following power management features:

- Low-power mode (IDLE instruction) conserves power by halting the CPU while maintaining all peripheral functions. Serial ports and timers continue operation, restarting the CPU only when additional data processing is needed.
- CLKOUT switching allows the external CLKOUT1 signal to be stopped, providing power savings when external clock synchronization is not necessary.

This application report describes techniques for analyzing system and device conditions to determine operating current levels. From this analysis, power dissipation for the device can be determined. Knowledge of power dissipation can in turn be used to determine device thermal management requirements.

CMOS POWER CONSUMPTION

In CMOS logic, internal node voltages swing completely from one power supply rail to the other. The voltage change on a gate capacitance requires charge transfer, and therefore causes power consumption. Once the gate capacitance is charged, the gate can maintain a DC voltage level without any additional charge movement, and does not consume current. For this reason, CMOS circuitry only consumes power when switching states.

The required charge to change voltage levels on the gate is described by the following equation:

$$Qgate = Cgate \times V_{DD}$$

where	Qgate	is the charge required to change states	(Coulombs)
	V_{DD}	is the power-supply voltage	(Volts)
	Cgate	is the gate capacitance.	(Farads)

Repeated switching generates a current proportional to the switching frequency. Since current is defined as Coulombs per second (Amperes), the current can be calculated as follows:

$$I = Qgate \times frequency = (Cgate \times V_{DD}) \times frequency$$

where	I	is the current in Amperes (Coulombs per second).

For example, the current consumed by an 80-pF capacitor being driven by a 10-MHz CMOS level square wave with 5-V V_{DD} is calculated as follows:

$$I = (80 \times 10^{-12} \text{ Farads}) \times (5 \text{ V}) \times (10 \times 10^{6} \text{ Hz}) = 4 \text{ mA}$$

This approach can be generalized to include all of the internal node capacitances in a device to determine the total device current. From the equation above, it is clear that the current is proportional to the supply voltage level, the total number of charging nodes and their capacitances, and the switching frequencies of those nodes.

$$I_{device} = C_{total} \times V_{DD} \times f_{clock}$$

where	I_{device}	is the total device current
	C_{total}	is the total node capacitance of all internal switching nodes
	f_{clock}	is the switching frequency of the device clock.

Since knowing the number and capacitance of all of the internal switching nodes is in reality an unmanageable task, the current under different conditions can be determined empirically by measuring the current level for a particular algorithm under known frequency and supply voltage conditions, and then scaling the current value to determine the behavior under different conditions. This is the method used in this report. Current was measured under known conditions and scaled to represent a frequency-dependent current factor usually expressed in mA per MHz or mA per MIPS depending on the situation. This current factor then can be multiplied by the operating frequency to determine the total current.

The total power dissipation is dependent on internal operation as well as external bus cycles and the loads associated with the external buses. Bipolar devices (TTL loads), pullup resistors and other devices consume DC power that adds a constant offset to the total current. The effects of these additional components are examined later in this report.

GENERAL DEVICE CURRENT CHARACTERISTICS

In general, device current requirements vary according to several system- and device-related considerations. Among these are supply voltage, temperature, and device program activity, along with other considerations normally associated with integrated circuits.

Additionally, because current requirements of CMOS technology are related to charging and discharging capacitances internal and external to the device, operating current also depends on considerations such as clock frequency, external load capacitance, and data patterns.

The ’C2xx devices exhibit power supply requirements consistent with CMOS device characteristics. Aspects of these characteristics that are necessary for analysis of device power supply current requirements are discussed in detail in this application report. A thorough working knowledge of device architecture and operation is critical to understanding the power analysis described in this document. Detailed information regarding ’C2xx device architecture and operation is found in the *TMS320C2xx User’s Guide* (literature number SPRU127B).

Current Components

The V_{DD} supply to the ’C2xx devices are internally separated into four groups, each supplying a different set of internal circuitry. The four groups are:

- Address
- Data
- Control
- Internal

The address and data groups supply the output drivers on the address and data buses, respectively. The control group supplies all other outputs and the internal group supplies all of the internal device logic. This internal power structure isolates noise generated in the high-current output drivers for the address and data buses from propagating through the internal logic.

In most applications, all of the V_{DD} connections are connected together externally and can be considered a single supply. Most device activity involves the internal portion of this supply, but the use of the external buses causes contribution from address, data and control lines.

Current Dependencies

The actual power-supply current depends on many factors. These factors can be divided into two groups: system-related, and device- or algorithm-related. Some system-related factors are shown below:

- Operation frequency
- V_{DD} supply voltage levels
- DC loads
- Operating temperature

Of the system-related factors, the most significant is operating frequency. As shown previously, the CMOS current is directly proportional to the switching rate. If the switching rate doubles, the frequency-dependent component of the device current also doubles. V_{DD} supply voltage levels also strongly affect the device current, but not as significantly as operating speed. DC loads (such as TTL loads) introduce a constant

offset in the device current and are not frequency-dependent. Operating temperature affects the total device current less significantly than any of the other factors.

Total device current is also affected by device- and algorithm-related factors such as:

- Program activity
- Internal and external bus data switching patterns
- Utilization of external buses

Since the total number of switching nodes affects the current, it is clear that the nature of the program activity strongly affects the current. Instructions that perform several parallel actions consume more current than simpler instructions. Instructions which involve moving data between the CPU and on-chip memory utilize internal address and data buses and use more current than instructions which involve the CPU alone.

The nature of the data being moved by way of internal buses also affects the current. Data buses are generally fabricated as a set of connections routed together. Each of these lines has a characteristic capacitance with respect to the silicon substrate on which they are fabricated. Since these bus lines are frequently routed adjacent to each other, they also possess an intersignal capacitance, or a capacitance between the adjacent bus lines. When the voltage on a bus line changes, these capacitances must also be charged and discharged as described previously. Since some of these capacitances exist between signal lines, the necessity to charge or discharge depends on the voltage levels on both lines. Therefore, the data patterns on the bus affect how many of these characteristic capacitances must charge, and consequently affect the total device current. For this reason, driving a bus with an alternating pattern of AAAAh/5555h consumes more current than driving 0000h/FFFFh. In both cases, all 16 lines are switching so the current contribution due to each line's capacitance with respect to the silicon substrate is the same. However, in the former case, the intersignal capacitance between the bus lines must charge and discharge, which consumes current. In the latter case, this does not occur since all of the bus lines follow the same voltage.

External buses have similar effects to the internal buses mentioned above with some additional considerations. External buses have greater intrinsic capacitance than internal buses because of the nature of the packaging and the presence of high-output current drivers for the pins. External buses also experience the load of the other external devices to which they are connected. Greater capacitance results in greater current. External buses also have data-dependent current levels, and use of the external buses instead of internal buses significantly adds to the total device current.

Considering the contribution of the factors mentioned above, the total I_{DD} power supply current can be described as the following equation:

$$I_{total} = (I_{int} + I_{addr} + I_{data} + I_{cntl}) \times V_S \times T$$

where

I_{total}	is the total I_{DD} supply current
I_{int}	is the current component due to all internal circuitry
I_{addr}	is the current component due to external address bus activity
I_{data}	is the current component due to external data bus activity
I_{cntl}	is the current component due to external control line activity
V_S	is the scale factor due to supply voltage
T	is a scale factor due to operating temperature

Algorithm Partitioning

The total current consumed by the device changes based on program activity. Some program sections use more current due to external bus usage or data patterns. Other sections may use less because the activity is completely confined to the CPU, or there is no activity such as in a power-down or idle state. Analysis of the device power consumption is greatly simplified by considering each section, or partition, of an algorithm that exhibits distinct concentrations of activity separately. These partitions of significant device activity can be considered the major contributors to the overall device current and shorter periods of less significant activity can be ignored. Each partition can be analyzed to determine it's current use and these contributions can then be time-averaged to determine the total device current. This approach generally simplifies the current analysis compared to a more detailed analysis that may not yield significantly increased accuracy for the increased analysis complexity.

Test Setup Description

All of the 'C2xx power measurements were performed on the test setup shown in Figure 1. Both 'C203 and 'C209 devices were tested. Pullup resistors were added to the data bus and to other control inputs as necessary. The power supply for the pullups was not connected through the meter in order to avoid inclusion of this current in the measurements because in a system, the power to drive the 'C2xx inputs high would be supplied by another external device. Capacitive loads were included on outputs where relevant to the measurement and were varied to determine the power supply dependence on this parameter. The maximum rated output load capacitance for the device is 80 pF.

The clock to the device was provided by a Sony/Tektronix AFG2020 function generator. Unless otherwise indicated, all measurements were made with the device in phase-locked loop (PLL) multiply-by-2 clock mode. Various frequencies were used as indicated.

Measurements were made using a Tektronix DM 501 digital multimeter, and unless otherwise specified, were made at a supply voltage of either 3 V or 5 V (as indicated), at an ambient temperature of 25°C. Current traces for the example code were made using a Tektronics AM 503 current probe.

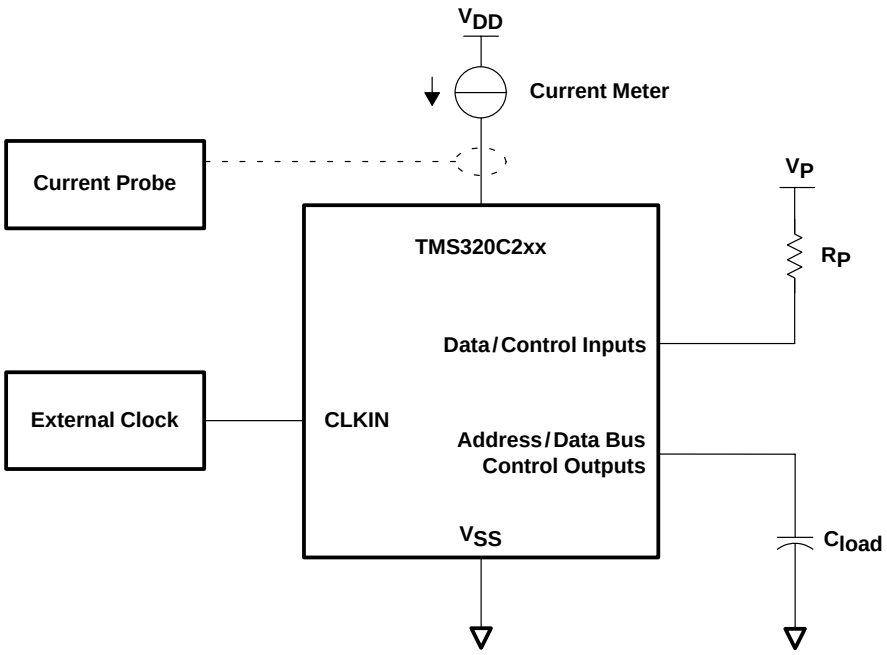

Figure 1. Test Setup

CURRENT DUE TO INTERNAL COMPONENTS

Clock Generation Circuitry

The TMS320C2xx digital signal processors are fabricated in a fully static CMOS technology. The current used by the device is entirely due to switching currents and virtually no current is consumed when the device is not switching. This provides the ability to conserve power by operating the device at very low clock rates, or stopping the clock, without corruption of data (assuming all timing requirements have been met as outlined in the data sheet). References to “processor clock” or “device clock” in this document refer to CLKOUT1.

Clock Modes and Their Affect on Power Consumption

A portion of the total device power is consumed by the circuit which generates and distributes the internal processor clock. The ’C2xx family provides two methods for generating the processor clock:

- **Internal divide-by-2 with external crystal**

  This option allows the user to construct a simple crystal oscillator using as little as three external components (a crystal and two capacitors) in conjunction with an internal inverting amplifier on the ’C2xx. The oscillator runs from the DSP power supply to generate an input frequency at X2/CLKIN. This frequency is divided by 2 to become CLKOUT1.

  Instead of using a crystal, the user can also inject an external clock source into X2/CLKIN with X1 left unconnected. The input frequency is still divided by two to generate CLKOUT1. In this case, less power is consumed by the DSP since it is not driving a crystal, but power is consumed somewhere else in the system to generate the input clock signal.

  Note that in internal divide-by-2 mode, the PLL circuitry is disabled and therefore cannot be used (and is not consuming any power).

- **Phase-locked loop (PLL) clock multiplication**

  The ’C2xx has an internal PLL which can lock onto an input clock signal and generate CLKOUT1 as a multiple of the frequency of the input signal. The frequency multiplication factors available are x1, x2, and x4. The availability of each of these multiplication factors varies on different ’C2xx devices. The user should refer to the *TMS320C2xx User’s Guide* (literature number SPRU127B) or the *TMS320C209, TMS320C203, TMS320VC203 Digital Signal Processor* data sheet (literature number SPRS025) for detailed information on available clock modes on each device.

Note that these two clock modes are mutually exclusive and cannot be operated together. When internal divide-by-2 mode is selected, the PLL is disabled and does not consume power. When PLL mode is selected, the oscillator circuitry is disabled and does not consume power.

The clock generation circuitry basically uses a constant amount of current at all times, unless the input clock speed is changed or stopped. This current forms background current that does not change due to CPU activity.

Power-Down Mode

The current use of each of these clock generation modes was measured with the CPU in power-down mode and the peripherals inactive. Power-down mode is a low-power mode that can be initiated by the $\overline{\text{HOLD}}$

signal or by execution of the IDLE instruction. When power-down mode is initiated, all CPU activities are halted. This reduction in activity saves power. The on-chip peripherals (timer, serial ports) are not halted and can run normally.

When power-down mode is initiated by execution of the IDLE instruction, all activities of the CPU are halted. The CLKOUT1 pin remains active (if its output has been enabled in the CLK register). Power-down mode terminates when the CPU receives a reset or an unmasked hardware interrupt (internal or external). Since the peripherals are still active during power-down mode, they can generate an interrupt to bring the CPU out of power-down mode if the interrupt is not masked.

When power-down mode is initiated by the $\overline{\text{HOLD}}$ signal, the CPU completes the current instruction which is executing and then internally executes an IDLE instruction. The CPU stays in power-down mode until $\overline{\text{HOLD}}$ is deasserted. After $\overline{\text{HOLD}}$ has been released, a hardware interrupt is required to initiate CPU activity again. For more detail on how to set up power-down mode, see the *TMS320C2xx User's Guide* (literature number SPRU127B).

Table 1 shows the power consumption of the clock generation circuitry. Note that these values contain both a frequency-independent and a frequency-dependent component. The clock generation circuitry is the only circuitry on the processor which contributes DC current components. The frequency-dependent component is expressed in mA per MHz of CLKOUT1.

Table 1. 'C2xx Clock Generation Power Consumption

CLOCK MODE	SUPPLY VOLTAGE	CURRENT USE IN IDLE MODE
Divide-by-2	3 V	0.1 mA + 0.45 mA per MHz
	5 V	0.2 mA + 0.80 mA per MHz
PLL multiply-by-1	3 V	0.3 mA + 0.28 mA per MHz
	5 V	0.8 mA + 0.60 mA per MHz
PLL multiply-by-2	3 V	0.3 mA + 0.25 mA per MHz
	5 V	0.8 mA + 0.50 mA per MHz
PLL multiply-by-4	3 V	0.3 mA + 0.21 mA per MHz
	5 V	0.8 mA + 0.46 mA per MHz

In the divide-by-2 clock mode, the frequency-independent component comes from the biasing circuitry necessary for proper operation of the oscillator circuit. The internal inverter which is used as an inverting amplifier for the oscillator has been biased intentionally in the transition region between the two logic states. This provides the gain and instability necessary for the oscillator to start upon power up; however, this biasing action also consumes DC current.

The phase-locked loop circuitry also requires special biasing for the PLL to operate properly. Different PLL modes generate different frequency-dependent contributions to the device current, but the DC current due to the PLL biasing remains essentially constant over each of the operating modes.

Internal CPU Activity

The power use of the 'C2xx devices is directly related to the level of CPU activity. Many factors affect CPU current use—including the instruction complexity (the amount of parallel operation being performed by the instruction), the usage of the internal buses (including the data patterns on the buses), and the effects of using the repeat option on instructions. These effects are examined in the sections that follow.

Internal CPU Functional Blocks

The 'C2xx CPU is composed of several hardware blocks which perform specific functions. The functions of these blocks are described in detail in the *TMS320C2xx User's Guide* (literature number SPRU127B).

The blocks also have specific sets of instructions associated with their functions. The CPU functions have been placed in the following groups for consideration of the power use:

- **Accumulator and Central Arithmetic/Logic Unit (CALU) functions**
  - Include shift operations, Boolean operations, non-multiplication arithmetic operations and accumulator load/store/modify operations.
- **Auxiliary Register and Auxiliary Register Arithmetic Unit (ARAU) functions**
  - Include auxiliary register load/store/modify operations that do not occur as a parallel operation in other CPU instructions
- **Multiplier functions**
  - Include T register operations and multiply/accumulate operations
- **Branch and Control functions**
  - Include branches, calls and returns

Instruction Complexity

The current used during specific CPU instructions is directly related to their complexity. Instructions that perform more parallel operations consume more current. Instructions that load or shift internal registers generally require the least current. Instructions that minimize use of the internal and external data buses also use less current. A more detailed examination of the effects of memory usage is covered later.

Current increases with use of arithmetic functions such as addition, subtraction and Boolean functions. Current further increases when the hardware multiplier is used. The ’C2xx processor provides several multiplication instructions that perform different levels of parallel operations; for example, these instructions can automatically load the T register, accumulate the previous product, move data through a memory array, or perform combinations of all three operations. The power use of the instructions increases as the amount of parallel operations increases. The most power-intensive instruction on the device is the MACD instruction, which multiplies two 16-bit values, accumulates (adds) the previous product to the accumulator register, and moves data in a memory array all in one clock cycle.

Power Effects of Repeated Instructions

The ’C2xx family of processors provides an automatic means for repeating, or “looping”, an instruction without the pipeline overhead associated with implementing a loop through the use of a branch instruction. The branch instruction disrupts the pipeline flow by causing a discontinuity in the program address sequence. The discontinuity causes the pipeline to flush, and consequently, causes the application to incur an extra four clock cycles for each pass of the loop. Since many computations and logical operations can require repetition of the same instruction, a branched loop becomes an inefficient method for repeating a single instruction in terms of pipeline overhead.

The ’C2xx processor solves this problem by offering a “repeat” instruction (RPT) that repeats the single instruction, following it up to 256 times with zero overhead. Assuming the pipeline is full when the RPT instruction is encountered, the instruction that follows it is repeated the specified number of times with zero pipeline overhead. In other words, the repeated instruction behaves much as it would if it were explicitly listed many times. But instead of consuming up to 256 words of memory (for straight-line code), the loop can be implemented in only two instructions.

Although some instructions are not useful to repeat, such as instructions which load or store values in the accumulator or auxiliary registers, many other instructions can be repeated, including arithmetic and logic instructions, multiply/accumulate operations, and “no operation” (NOP) instructions repeated for the purpose of implementing delays.

Repeated instructions deserve consideration in terms of power because they can consume significant lengths of time and the action being repeated changes the power consumption required to execute the instruction. Most instructions require less current to execute when they are repeated because, although the same action is performed repetitively, the instruction is only fetched once. Consequently, the power required to fetch the instruction is saved.

Multiply/accumulate (MAC/MACD) instructions consume more power when repeated because of an automatic feature of the instructions. When a single MAC is executed, the device multiplies data contained in the specified program memory address by the data contained in the specified data memory address. If multiplication/accumulation of a series of numbers is required (as in digital filters), the instruction can be repeated using indirect addressing to increment the data memory address, but the program memory address is automatically incremented by the CPU. This automatic address-increment function causes the instruction to require more power to execute. Note that this automatic increment of the program memory address only occurs when the instruction is looped using the RPT instruction. It does not occur if the MAC instruction is repeated as straight-line code. An example of code illustrating these two cases is shown below in Figure 2.

```
RPT   #99
MAC   coeff, *–
```

words: 3
cycles: 105†
power: 2.6 mA per MIPS @ 5 V†
1.4 mA per MIPS @ 3 V†

(a)

```
MAC   coeff, *–
MAC   coeff+1, *–
MAC   coeff+2, *–
      . . .
MAC   coeff+98, *–
MAC   coeff+99, *–
```

words: 200
cycles: 300†
power: 2.4 mA per MIPS @ 5 V†
1.3 mA per MIPS @ 3 V†

(b)

† Cycle counts and power use varies due to locations of operands in SARAM, DARAM or ROM

Figure 2. Example of Repeated (a) vs. Straight-Line (b) Code Use of MAC

Current Use of the ’C2xx Instruction Set

Measurements made on the power characteristics of the ’C2xx instruction set are included below. The measurements were performed by configuring the ’C2xx device in a known state, repeating instructions and measuring the current use during the repetition of the instruction. All measurements were made in PLL multiply-by-2 clock mode unless otherwise specified and the current readings include the current used by the clock generation circuitry. All instruction-set current measurements were made with the peripherals (timer, serial ports) inactive and with address visibility disabled. The effects of address visibility are discussed in more detail in the section “Address Visibility Feature”. The measurements were performed with the test code executing from on-chip single-access RAM (SARAM).

Instructions were repeated by two methods. In the “straight-line” case, the instruction under test was explicitly repeated many times, followed by a branch instruction back to the top of the block of the instructions. In the “RPT” case, the RPT instruction was used to generate many repetitions of the instruction under test, followed by a branch instruction to return control back to the RPT instruction. In each case, the two test cases are illustrated in Figure 3.

```
testloop
        I1
        I1
        ...     (256 times)
        I1
        I1
        B       testloop
```

(a) straight-line method

```
testloop
        RPT     #255
        I1
        B       testloop
```

(b) RPT method

Figure 3. Loop Implementation for Instruction Current Measurements

The measurements in Table 2 are intended to give a quantitative point of reference for the power characteristics of the ’C2xx instruction set. Actual observed current may vary somewhat due to pattern of data being manipulated, code source memory type and internal vs. external bus usage. The effects on instruction current use due to memory-related factors are considered in more detail in the next section. All of the measurements are given in units of mA per MHz of CLKOUT1. All measurements were performed using an addressed operand (either defined data memory address or indirect addressing) unless otherwise specified.

Table 2. TMS320C2xx Instruction Set Power Characteristics†

INSTRUCTIONS	STRAIGHT-LINE 5 V	RPT 5 V	STRAIGHT-LINE 3 V	RPT 3 V
GENERAL FUNCTIONS				
NOP	1.5	1.0	0.8	0.6
IDLE (divide-by-2 mode)	0.8	—	0.45	—
IDLE (PLL multiply-by-2 mode)	0.5	—	0.25	—
AUXILIARY REGISTER / ARAU FUNCTIONS‡				
MAR (w/o AR increment/decrement)	1.4	—	0.8	—
MAR (with AR increment/decrement)	1.5	—	0.8	—
SAR (to on-chip memory) §	1.6–2.1	—	0.9–1.2	—
LAR (to on-chip memory) §	1.4–1.6	—	0.8–0.9	—
LAR (immediate operand) §	1.7–1.9	—	0.8–0.9	—
BRANCH AND CONTROL FUNCTIONS				
B to self	1.7	—	1.0	—
B to another address	1.8	—	1.0	—
CALL / RET	1.6	—	1.0	—
PUSH	1.5	1.1	0.8	0.6
PSHD	1.6	1.3	0.9	0.7
POP	1.5	1.1	0.8	0.6
POPD	1.6	1.2	0.9	0.7
BLOCK MEMORY FUNCTIONS¶				
TBLR §#	¶	1.7–2.0	¶	0.9–1.1
TBLW §#	¶	1.7–2.0	¶	0.9–1.1
BLDD §	¶	1.9	¶	1.1
BLPD §	¶	2.0	¶	1.1
DMOV	1.8	1.5	1.0	0.8
MULTIPLIER FUNCTIONS				
MPY (immediate operand)	2.4	1.8	1.3	1.0
MPY	2.4	2.1	1.3	1.2

† All measurements are in units of mA per MHz of CLKOUT1.
‡ Type of auxiliary register increment/decrement (single / indexed / bit-reversed) has little effect on the current required to execute the instruction.
§ Current use of this instruction is data-pattern-dependent.
¶ The indicated block move instructions were only tested in auto-repeat mode since they are most commonly used in this mode.
# TBLR / TBLW instructions were tested with data source in SARAM and data destination in DARAM Block 0.
|| Accumulator shift values have little effect on the current required to execute the instruction.

Table 2. TMS320C2xx Instruction Set Power Characteristics† (Continued)

INSTRUCTIONS	STRAIGHT-LINE 5 V	RPT 5 V	STRAIGHT-LINE 3 V	RPT 3 V
MULTIPLIER FUNCTIONS (CONTINUED)				
MPYA	2.5	2.1	1.4	1.2
MAC	2.4	2.6	1.3	1.4
MACD	2.6	2.9	1.4	1.6
APAC	1.5	1.1	0.8	0.6
LT §	1.6–1.9	—	0.9–1.1	—
LT (with AR increment/decrement) §	1.7–2.0	—	1.0–1.1	—
LTA	1.7	—	0.9	—
LTD	1.8	—	1.1	—
SQRA	1.7	1.3	0.9	0.7
SQRS	1.7	1.3	0.9	0.7
ACCUMULATOR/CALU FUNCTIONS				
Logic functions:				
SFL (shift left)	1.4	1.0	0.8	0.6
SFR (shift right)	1.4	1.0	0.8	0.6
ROL (rotate left)	1.5	1.1	0.9	0.6
ROR (rotate right)	1.5	1.1	0.9	0.6
CMPL (complement)	1.6	1.2	0.9	0.7
AND	1.6	1.3	0.9	0.7
OR	1.6	1.3	0.9	0.7
XOR	1.7	1.3	0.9	0.7
Arithmetic functions:				
ADD (immediate operand)	1.5	—	0.8	—
ADD	1.6	1.3	0.9	0.7
SUB (immediate operand)	1.5	—	0.8	—
SUB	1.6	1.3	0.9	0.7
NEG	1.6	1.0	0.9	0.7

† All measurements are in units of mA per MHz of CLKOUT1.
‡ Type of auxiliary register increment/decrement (single / indexed / bit-reversed) has little effect on the current required to execute the instruction.
§ Current use of this instruction is data-pattern-dependent.
¶ The indicated block move instructions were only tested in auto-repeat mode since they are most commonly used in this mode.
# TBLR / TBLW instructions were tested with data source in SARAM and data destination in DARAM Block 0.
|| Accumulator shift values have little effect on the current required to execute the instruction.

Table 2. TMS320C2xx Instruction Set Power Characteristics† (Continued)

INSTRUCTIONS	STRAIGHT-LINE 5 V	RPT 5 V	STRAIGHT-LINE 3 V	RPT 3 V
Accumulator modifications:				
LACC (long immediate operand)‖	2.1	—	1.1	—
LACC	1.6	—	0.9	—
LACL (immediate operand)	1.5	—	0.8	—
LACL	1.6	—	0.9	—
LACT	1.6	—	0.9	—

† All measurements are in units of mA per MHz of CLKOUT1.
‡ Type of auxiliary register increment/decrement (single / indexed / bit-reversed) has little effect on the current required to execute the instruction.
§ Current use of this instruction is data-pattern-dependent.
¶ The indicated block move instructions were only tested in auto-repeat mode since they are most commonly used in this mode.
# TBLR / TBLW instructions were tested with data source in SARAM and data destination in DARAM Block 0.
‖ Accumulator shift values have little effect on the current required to execute the instruction.

Effects of Memory Usage on Power Consumption

Although device power use for a given algorithm is primarily based on CPU power, the memory environment in which the algorithm is running also plays a role in the total power consumption. An understanding of the effects of memory and bus configuration on power consumption are valuable in optimizing system and software design for peak performance. In this section, memory types, use of on-chip vs. off-chip memory, usage of internal buses, address visibility and power characteristics of each of the memory types is examined in detail.

Memory Types

The ’C2xx processor family provides several on-chip memory options that can be used in conjunction with external memory to lower system cost and reduce system power use. Several types of on-chip memory are available, including single-access RAM (SARAM), dual-access RAM (DARAM), and ROM. General information regarding the ’C2xx memory structure is described here. For detailed information on memory maps and memory configuration, refer to the *TMS320C2xx User’s Guide* (literature number SPRU127B).

The ’C2xx single-access RAM is static RAM, which, under normal conditions, allows a single memory read or write access in one clock cycle. However, with an algorithm configured appropriately, dual access (two memory operations in one cycle) is possible. The location in the memory map and the size of this memory block is dependent on the processor used; however, in each case, the total SARAM block is composed of 2K-word and 1K-word sub-blocks. For example, an 8K-word SARAM block is actually composed of four 2K-word sub-blocks. If the SARAM block is an odd multiple of 1K words in size, it is composed of all 2K-word sub-blocks plus one 1K-word sub-block (that is, $7K = 3 \times 2K + 1K$). The SARAM is single-access if both accesses are targeted at memory locations within the same 2K-word (or 1K-word) sub-block because they cannot be performed in the same clock cycle. If the two accesses are located in different sub-blocks, they can be performed in the same clock cycle and the memory essentially becomes dual-access. Knowledge of this fact can be used in the design of the memory configuration to take advantage of this conditional dual-access capability and increase CPU throughput by eliminating memory-based pipeline delays. As CPU throughput increases, current also increases, so this behavior should be considered when estimating power use of an algorithm.

The dual-access RAM (DARAM) is always dual access, regardless of the addresses of the memory operations. This memory is usually provided in three separate blocks labeled block 0 (B0), block 1 (B1) and block 2 (B2). These blocks are identical in behavior.

The on-chip ROM (read-only memory) is memory that can be used as program memory or used to store coefficient tables that are never altered. As with external ROM, the memory is non-volatile, meaning that it retains the data it contains even if device power is removed. This memory block is single-access. The ROM is mask-programmed at the factory at the time of fabrication and cannot be reprogrammed or erased. For information on how to submit code for on-chip ROM production, refer to the *TMS320C2xx User's Guide* (literature number SPRU127B) or contact Texas Instruments.

Of these memory types, ROM is used primarily for storing program code and constants. SARAM can be configured as either program or data memory, but it is usually used as program memory. DARAM block 2 and block 1 are configured as data memory. DARAM block 0 can be configured as either program or data memory.

Internal Memory Bus Structure and the External Memory Interface

The 'C2xx device is based on a modified Harvard architecture in which program and data occupy separate memory spaces. The device also uses a third memory space called I/O space to provide flexibility for memory-mapped access to input/output devices such as analog-to-digital (A/D) converters, digital-to-analog (D/A) converters, codecs and other devices. In the memory map for each of these spaces, some sections of memory are mapped as internal and some are mapped as external. When the CPU activity is limited to internal-memory accesses, the external-memory interface (composed of the external address, data and memory-control signals) is inactive. When an external-memory access occurs, the external-memory interface activates automatically. This feature conserves power since the output drivers for the external-memory interface are not operated when not necessary. As a result of this feature, less power is required to operate from internal memory than from external memory. The control of the external interface is automatic and does not require any special effort from the user.

Address Visibility Feature

The address visibility feature allows the user to set a control bit (the AVIS bit) which allows the internal program address to appear at the external address pins—even during internal-memory accesses. This allows the internal program address to be traced and interrupt vectors to be decoded in conjunction with the interrupt acknowledge ($\overline{\text{IACK}}$) when the interrupt vectors reside in on-chip memory. This ability can be a valuable debugging tool; however, this operation of the external address bus consumes power that is not necessary for the execution of the program. When the debugging is complete, the address visibility mode should be disabled to conserve power.

Power Use of Different Memory Types

The different types of on-chip memory also exhibit different power use characteristics for the same functions. Several common functions—including block transfers from program to data space, code execution from program space, and data read/writes to internal RAM memory—were evaluated.

Program Memory – SARAM vs. ROM

Memory accesses to program space can use ROM, SARAM, DARAM or external memory. Since the DARAM block in program space is 256 words, it is more common to store code and coefficients in SARAM, ROM or external memory. The current required to operate the external-memory interface is covered later in this document. Operations using SARAM and ROM were compared and yielded the following results:

Block Transfers

Block transfers from program space to data space are accomplished by using the BLPD (block move program to data) or the TBLR (table read) instructions in conjunction with the RPT instruction. For these transfers, the CPU current required with ROM as the source memory is approximately 10% less than the current required to perform the same operation with SARAM as the source memory.

Code Execution

Code execution from ROM also requires approximately **10%** less current than the CPU current required to execute the same code from SARAM.

Effects of Bus Data Patterns

As previously described, data buses have a characteristic capacitance associated with each line as well as intersignal capacitances between the lines. These capacitances cause the current required for a given operation to vary depending on the data patterns occurring on the bus. Measurements of this phenomenon were made by performing repetitive changing data on each read. Data patterns were varied to observe the current difference. The test was performed by reading from DARAM B1 from two adjacent addresses (differing by only 1 bit to minimize effects of the address bus). The reads alternate between the two locations, and the locations can be programmed with the desired data patterns.

The results, shown in Figure 4, provide a method to scale current use of a particular instruction based on the “data complexity”, or amount of changing data on the data bus. The data pattern FFFF/0000 shows the effect of having all 16 data lines alternating, which causes charging/discharging of the characteristic capacitance of the bus, but the adjacent lines are always at the same level which minimizes the intersignal effect. The data pattern AAAA/5555 causes both the characteristic and intersignal capacitances to be involved and is the worst-case current use. The FFFF/0000 pattern requires approximately 80% of the current required to perform the “worst-case” AAAA/5555 pattern. Figure 4 can be used to approximate the changes in current use due to changes in data-bus patterns.

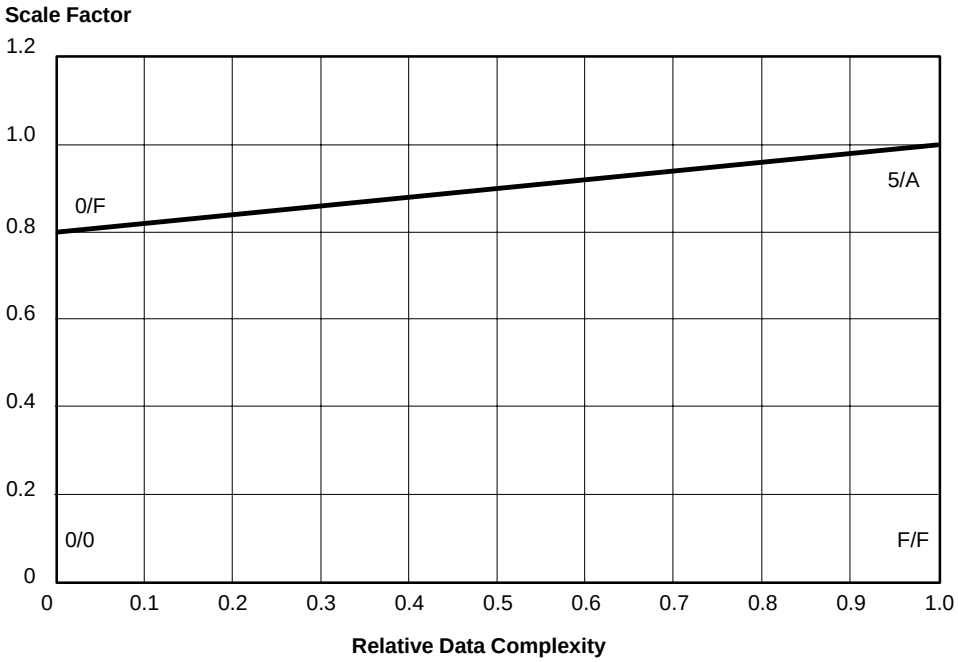

Figure 4. Current Scale Factor for Bus Switching

CURRENT DUE TO PERIPHERALS

Although the CPU contributes most of the current involved in executing the application code, the on-chip peripherals also contribute to the current use of the device. The ’C2xx family provides three on-chip peripheral devices for use with the CPU: the timer, the synchronous serial port, and the asynchronous serial port. The current contribution of each of these devices is considered in detail below.

Timer

Although the timer runs independently of the CPU, the speed of the timer is still based on CLKOUT1. Consequently, the current requirement of the timer changes with processor clock speed just as other functions do. The current use of the timer also depends on its activity. The more frequently the timer reloads, the higher the operating current is. However, in most cases, the timer counts many cycles between reloads, so the effect of the reload on the total current required is negligible.

Evaluations of the timer indicated the following current requirements as long as the timer is reloaded infrequently:

0.013 mA per MHz of CLKOUT1 at **5 V**
0.006 mA per MHz of CLKOUT1 at **3 V**

Since these values are relatively low, it may be desirable to neglect the timer in calculation of the current unless increased accuracy is desired or current use in the IDLE mode is being minimized. Figure 5 shows the current requirement of the timer as a function of frequency of timer reload.

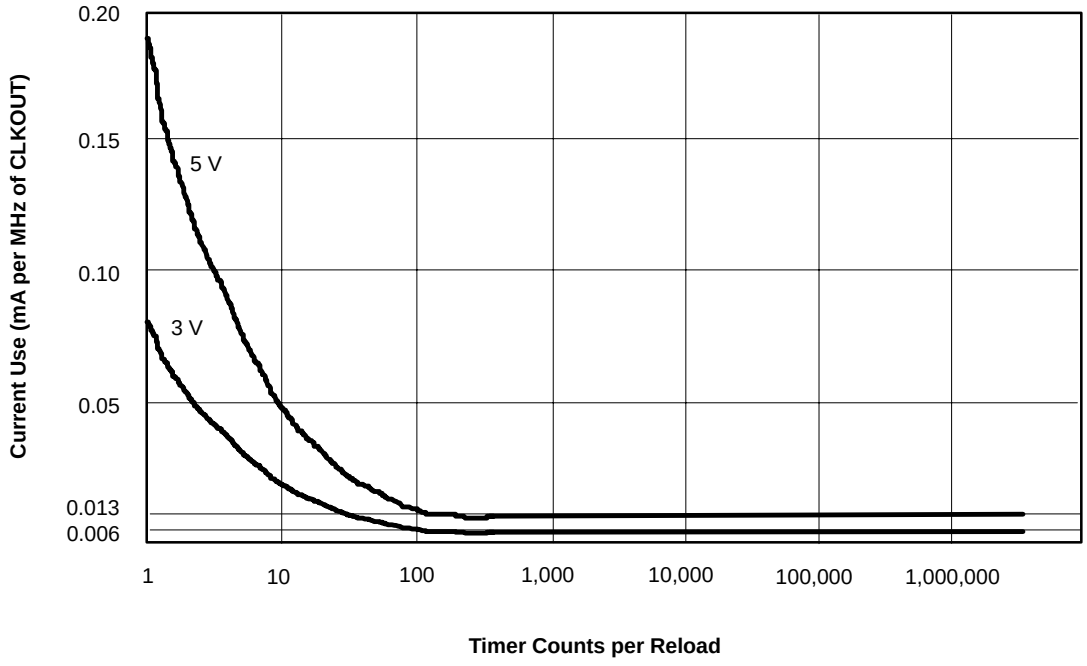

Figure 5. Timer Current vs. Reload Rate

Calculation Example:

For an application with the timer running at 20-MHz CLKOUT1 at 5 V, in which the timer reloads at a rate of 8 kHz, the timer period should be set to 2500 cycles (20 MHz/8 kHz = 2500). With the timer reloading

once every 2500 clock cycles, the current can be estimated from Figure 5 to be approximately 0.013 mA per MHz of CLKOUT1.

So, the additional current required to operate the timer is:

$$(0.013 \text{ mA per MHz}) \times (20 \text{ MHz}) = 260 \ \mu\text{A}$$

Synchronous Serial Port

The ’C2xx synchronous serial ports provide direct communication with serial devices such as codecs and A/D converters. The serial ports can also be used for intercommunication between processors in multiprocessing applications. As a peripheral device to the CPU, the serial ports also use power based on their activity and can be disabled to conserve power when not needed.

External Synchronous Serial Port Interface

Both the serial port receiver and transmitter are based on three signals: the serial port transfer clock (CLKR/CLKX), the frame sync (FSR/FSX), and the data line (DR/DX). For detailed information on the operation of the serial port, refer to the *TMS320C2xx User’s Guide* (literature number SPRU127B). Since the serial port clock is always running, it consumes the most current to operate. If the serial port clock is generated internally, more current is required than if it is generated externally, since the device consumes additional current due to driving the external CLKR/CLKX pins. Since the operation of the serial port is synchronized to the serial port clock only, it is not dependent on the CPU clock unless the serial port clock is generated internally.

The power used by the port to transfer data on DX and DR is affected by both the serial port operating mode (burst/continuous) and the data pattern being transferred, since both affect the rate of activity on the data signal. The worst-case power use of the serial port occurs when transferring alternating bits (that is, AAAAh) in continuous mode. This causes the data signal to toggle at the rate of the serial port clock.

Effect of the First-In, First-Out (FIFO) Buffer

The synchronous serial port also provides a 4-level first-in, first-out (FIFO) buffer to facilitate data transfer. The serial port can be programmed to generate interrupts after 1-, 2- or 4-word transfers, providing flexibility in use of the serial port and reducing the frequency of CPU interruption.

The power use of the synchronous serial port was evaluated in the 1-, 2- and 4-word interrupt modes. The difference of current use was found to be minor (approximately 0.04 mA per MHz of the serial port clock) between 1-word interrupt and 4-word interrupt modes. For most applications, this difference is negligible.

Synchronous Serial Port Measurements

The synchronous serial port receiver and transmitter were evaluated separately (since they can be operated separately). The measurements were made in continuous mode with continuous data-transfer occurring. This represents the highest current used by the synchronous serial port. The user can scale these values to determine actual current use based on the frequency of data transfer occurring in an application. The measurements that follow were made in continuous mode in 1-interrupt-per-word mode with the synchronous serial port interrupts masked to prevent contribution to the current from the CPU response to the interrupt. For both the receiver and transmitter measurements, CLKR/CLKX was generated internally. The low value of the measured current represents the data word FFFFh being transferred (least active data signal). The highest current value represents the data word AAAAh being transferred (most active data signal). The current measurements are represented in terms of mA per MHz of the serial port clock (not CLKOUT1). For the transmitter, the current required to operate the synchronous serial port clock

only is also included. This value represents the current used by the synchronous serial port if no data-transfer is occurring and if only the synchronous serial port clock is running. This measurement represents the current used to drive an unloaded CLKX signal.

Synchronous Serial Port Receiver:

0.16–0.30 mA per MHz of serial port clock at **3 V**
0.32–0.54 mA per MHz of serial port clock at **5 V**

Synchronous Serial Port Transmitter:

0.19–0.24 mA per MHz of serial port clock at **3 V**
0.58–0.75 mA per MHz of serial port clock at **5 V**

0.17 mA per MHz is required to operate CLKX only at **3 V**
0.55 mA per MHz is required to operate CLKX only at **5 V**

Asynchronous Serial Port

The asynchronous serial port provides a serial interface for communication without the necessity for a common synchronization clock between the transmitter and the receiver. This peripheral is useful for interfacing by way of asynchronous communications protocols such as RS232C. The asynchronous serial port is capable of simultaneously transmitting and receiving data at up to 250,000 10-bit characters per second.

The asynchronous serial port has only two external signals: the data-receive line (RX) and the data-transmit line (TX). Since the asynchronous serial port has no port clock that is always running, the power use of this peripheral is much lower than that of the synchronous serial port. Also, the data-transfer speed of the asynchronous serial port is slower than the power of that of the synchronous serial port—which also results in lower power use.

The asynchronous serial port receive and transmit functions were measured separately by transferring the worst-case data (AAh) at different baud rates. The current for this peripheral is also data-dependent because the less transitions the external signals make per second, the lower the current consumed. The worst-case current listed below can be scaled appropriately to the actual data. For the transmit measurements, the TX line was unloaded during the test.

Asynchronous Serial Port Receive Measurements:

1.5 μA per receive bit rate (in kilobits per second) at **3 V** (1 kilobit = 1000 bits)
2.5 μA per receive bit rate (in kilobits per second) at **5 V** (1 kilobit = 1000 bits)

Calculation Example:

The asynchronous serial port receiver operating at baud rate of 28 800 bits per second at 5 V generates:

$(2.5\ \mu A) \times (28.800) = \mathbf{72}\ \mu A$

Asynchronous Serial Port Transmit Measurements:

2.8 μA per transmit bit rate (in kilobits per second) at **3 V** (1 kilobit = 1000 bits)
4.6 μA per transmit bit rate (in kilobits per second) at **5 V** (1 kilobit = 1000 bits)

Calculation Example:

The asynchronous serial port transmitter operating at baud rate of 28 800 bits per second at 5 V generates:

$$(4.6\ \mu A) \times (28.800) = \mathbf{132}\ \mu A$$

The reader should keep in mind that the transmit or receive “bit rate” in the measurements above is the actual number of bits that is transferred per second. This is not the same as the “baud rate” mentioned in the *TMS320C2xx User’s Guide* (literature number SPRU127B). Changing the baud rate changes the time length for each bit, but if the same number of words are transmitted per second, the same number of bits is transmitted and consequently, the current remains the same. Since the current used by the asynchronous serial port is so low, for many applications, it may be negligible.

CURRENT DUE TO OUTPUTS

Now that the contribution of the internal components is known, current due to outputs can be considered. Outputs are any of the external signals which are driven by the processor including the address bus, the data bus and the control signals.

Categories of Outputs

Device outputs can be categorized into three functional groups. These categories are: data bus, address bus, and control outputs.

Data Bus

The address and data bus require an amount of current that is proportional to the overall parallel bus switching rate or memory cycle time. As discussed previously with respect to the internal buses, the worst case switching current for the bus occurs when all of the lines are switching and each line's steady state intervals are always in a logic state opposite to its neighbor's (that is, AAAAh, 5555h, AAAAh...). The current required to toggle all lines—where adjacent lines are in the same logic state—is approximately 80% of the worst-case value. The currents given below represent the worst-case current required to drive the unloaded external data bus.

The unloaded external data bus requires approximately:

0.12 mA per MHz of data-bus switching frequency **per bit** at **3 V**, and

0.20 mA per MHz of data-bus switching frequency **per bit** at **5 V**

So the worst-case switching current for the entire external data bus is:

1.92 mA per MHz of data-bus switching frequency at **3 V**, and

3.20 mA per MHz of data-bus switching frequency at **5 V**

In addition to the current shown above, the current due to external capacitive load on the data bus can be calculated as:

$$I_{load} = C_{load} \times V_{DD} \times F$$

where	I_{load}	is the additional current required per line due to the capacitive load
	C_{load}	is the external load capacitance per line
	V_{DD}	is the supply voltage
	F	is the switching frequency of the line

The maximum specified external capacitive load per pin for the 'C2xx family is 80 pF.

Address Bus

The address bus uses a similar amount of current as the data bus if all of the lines are switching; however, the more common operation is that the address lines are incrementing. Each of the bus lines has a characteristic capacitance. If, for simplicity, it is assumed that these are all equal, the total current required to drive the address bus is as follows:

$$I = CVF_0 + CVF_1 + CVF_2 + ... + CVF_{15} = CV\ (F_0 + F_1 + F_2 + ... + F_{15})$$

where	F_0	is the switching frequency of A_0
	F_1	is the switching frequency of A_1, and so on
	C	is the load capacitance, and
	V	is the voltage of V_{DD}

If the address bus is incrementing, each line toggles at a rate that is one-half of the next least-significant line as:

Since

$$F_{15} = \frac{1}{2}F_{14},\ F_{14} = \frac{1}{2}F_{13}, ...,\ F_2 = \frac{1}{2}F_1,\ F_1 = \frac{1}{2}F_0$$

then,

$$I = CV\left(F_0 + \frac{F_0}{2} + \frac{F_0}{4} + \frac{F_0}{8} + \frac{F_0}{16} + \frac{F_0}{32} + \frac{F_0}{64} + ... + \frac{F_0}{65536}\right) \approx 2CVF_0$$

Since each address bit is switching at a rate one-half of the next least-significant bit below it, the entire 16-bit bus requires an amount of current approximately equal to twice the current required to toggle one line only. This estimate assumes that the capacitive load on each of the address lines is the same. If branches occur where many address lines change, the instantaneous current increases accordingly.

The unloaded external address bus requires approximately:

0.12 mA per MHz of switching frequency **per bi**t at **3 V**, and
0.20 mA per MHz of switching frequency **per bit** at **5 V**

So, the worst-case switching current for the entire unloaded external address bus is:

1.92 mA per MHz of switching frequency at **3 V**, and
3.20 mA per MHz of switching frequency at **5 V**

The current required for incrementing the address bus is approximately:

0.24 mA per MHz of switching frequency at **3 V**, and
0.40 mA per MHz of switching frequency at **5 V**

The additional current due to external capacitive load on the address bus can also be calculated as shown above for the data bus.

Control Outputs

CLKOUT1 is the primary control output of concern in terms of power consumption because it is switching most rapidly. As discussed previously, if an external CLKOUT1 is not necessary, it can be disabled through the CLK register. Other control outputs, such as R/$\overline{W}$, $\overline{PS}$, $\overline{DS}$, $\overline{IS}$, $\overline{RD}$, $\overline{WE}$, $\overline{STRB}$, IO0, IO1, IO2, IO3, TOUT and XF, contribute to the overall power use depending on the system configuration. These outputs consume less current because they switch less frequently, but they can still be important to consider, especially when designing for low-power conditions.

The current required for driving each unloaded control output is approximately:

0.09 mA per MHz of switching frequency per output at **5 V**, and

0.17 mA per MHz of switching frequency per output at **3 V**.

The additional current for external capacitive loads on these outputs should also be added where necessary and can be calculated as shown above for the data bus.

The overall current contribution of the outputs is the sum of the individual groups as defined in the following equation:

$$I_{ext} = I_{add} + I_{data} + I_{ctrl}$$

where

I_{ext}	is the total current due to the outputs
I_{add}	is the current component due to address-bus activity
I_{data}	is the current component due to data-bus activity
I_{ctrl}	is the current component due to control output activity

Effects of Data Pattern Complexity

The measurements given above were taken under worst-case data-switching conditions—namely, switching between AAAAh and 5555h. This causes the maximum current use because both the characteristic capacitances and the intersignal capacitances of the bus must be charged on each switching cycle. If fewer lines are charging or if more adjacent lines share the same logic level, less current is required. A scale factor, as shown in Figure 4, can be applied to the calculation of the current due to the outputs to model more realistic conditions of data patterns on the bus.

Considerations of TTL and Other DC Loads

If any device outputs are routed to TTL or other predominantly DC loads, consideration must be made for these effects in the overall power calculation. The net result of DC loading to an output is that the current required to drive that output is increased by the magnitude of the average DC source current loading on the output.

As an example, consider a DC loading of 300 μA of source current per bit on the address-bus outputs when driving alternating 1’s and 0’s with a 50% duty cycle. Here, the increase in current is calculated as:

$$0.50 \times 300\ \mu A \times 16 = 2.4\ mA$$

In this case, an extra 2.4 mA of current is added to the current value required to drive the capacitive portion of the address-bus output loading calculated as described above.

TOTAL POWER DISSIPATION

The previous sections discussed power components associated with several different sections of the TMS320C2xx. These sections have been discussed separately to illustrate the contributions of each. The total power consumption of the device is the sum of the individual components. This total current value is determined as the total current supplied to the device through all of the V_{DD} inputs.

Multiple V_{DD} and V_{SS} pins are present on the device and can be routed to separate internal components. Consequently, all of the V_{DD} connections or all of the V_{SS} connections may not be common internally. Externally, all of these pins should be connected together in parallel to the power and ground planes, respectively, through as low an impedance as possible.

Calculation of Total Supply Current

Once all of the current components are calculated for unique periods of device activity, calculation of the total device power is straightforward. The separate components can be simply added to determine the overall current. Note that the CPU current measurements given in Table 2 include the current used by the clock-generation circuitry running in PLL multiply-by-two mode. If a different clock-generation mode is being used, these values can be adjusted by adding/subtracting the difference in current used by different clock modes (refer to Table 1).

To calculate the overall device current, a set of steps that examine the power issues discussed previously can be followed. These steps provide an approach to estimating the actual device power use. Some additional design considerations, which is discussed later in this report, can be employed to reduce power consumption.

Steps to Calculate Overall Device Power Consumption:

1. **Algorithm partitioning**

   The algorithm under consideration should be broken into sections of unique device activity and the power requirements for these section calculated separately. The sections can then be time-averaged to determine the overall device current requirement.

2. **CPU activity**

   The current contribution due to CPU activity can be determined by examining the code and determining the time-averaged current for each algorithm partition from the data in Table 2. Note that the data provided in Table 2 was measured with the ’C2xx running in PLL multiply-by-2 clock mode and the current required by the clock-generation circuitry is included in the measurements. If a different clock mode is used, these values may need to be adjusted. Table 1 shows the relative current use of each clock mode. Also, bear in mind that most measurements in this report are based on mA per MHz of CLKOUT1, not CLKIN. If the clock mode used is changed, the CLKIN speed also needs to be changed to generate the same CLKOUT1 frequency.

   When considering CPU current use, remember to consider the **effects of the RPT instruction** in the algorithm. RPT usually lowers the current required to execute a given instruction except in the case of multiply instructions, which increase in current due to automatic address generation when in repeated mode.

3. **Memory usage**

   Scale the current calculated in step 2 based on memory usage.

   Use of on-chip memory requires less current than off-chip memory (because of the additional current due to the external-memory interface).

   Running code from internal ROM requires less current than running from internal RAM.

4. **Peripherals**

   Consider the additional current required by use of the timer, synchronous serial port and asynchronous serial port

5. **Current due to outputs**

   Consider the current required by the algorithm to operate the **external address and data buses**. Scale these values to include the effects due to **capacitive loading** on the address and data buses.

   Include the current needed to operate other fast-switching outputs (CLKOUT1, R/$\overline{W}$). Don't forget to consider output pins the peripherals may be driving (that is, TOUT from the timer, or CLKX/FSX/DX from the serial port). The capacitive loads on these individual outputs should be considered and the current requirements scaled accordingly.

   Include the current needed to operate the **slow-switching outputs** ($\overline{HOLDA}$, $\overline{IACK}$). The capacitive loads on these individual outputs should also be considered and the current requirements scaled accordingly.

   Include the contribution of **TTL and other DC loads** on any outputs.

Calculation of Average Current

If power supply current is observed over the full duration of device activity, different segments of activity exhibits different current levels for different lengths of time. For example, a program may spend 80% of its time performing internal operations and drawing a current of 50 mA, but spend the remaining 20% of its time performing full-speed writes to an external device while drawing 110 mA.

While identifying peak current levels is important in order to establish power supply requirements, determining average current is often more important. This is particularly significant if periods of high-peak current are short in duration. Average current can be obtained by performing a weighted summation of the current due to various independent program segments over time. In the example just mentioned, the average current can be calculated as follows:

$$I = (0.8 \times 70\ mA) + (0.2 \times 110\ mA) = 62\ mA$$

Effects of Temperature and Supply Voltage on Device Operating Current

Two system factors, temperature and supply voltage, affect all of the current components equally. The effects of these factors can be included after the total device current has been calculated. Supply voltage and operating temperature should always be maintained within the required device specifications.

Device operating current is proportional to temperature; however, its variation due to temperature is small and therefore generally not significant. The device power measurements made in this report were taken at room temperature. If necessary, to account for absolute worst-case effects (including temperature across the total operating range of the device), power-supply current values can be scaled approximately +/– 1% for operation at device temperature extremes above and below room temperature, respectively.

Power-supply current requirements also vary proportionally to V_{DD} supply voltage levels. Figure 6 shows the variation in device operating current (as a scale factor) with respect to V_{DD} supply voltage. This data can be used to determine the scale factor that is applied to the total current calculated for any given period of device activity.

Effects due to temperature and supply voltage are the final factors to be applied to current values calculated for any given period of device activity. After scaling for these factors, actual current levels are produced, and average current for the entire duration of device operation can then be calculated, as described in the following sections.

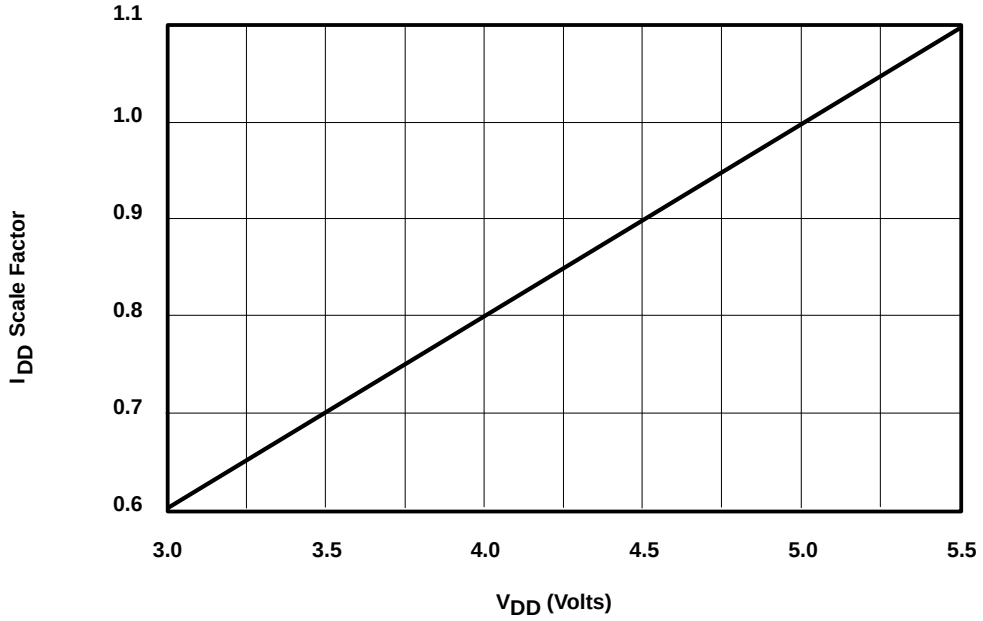

Figure 6. I_{DD} Variation With Respect to V_{DD} Supply Voltage

Thermal Management Considerations

Heating characteristics of the TMS320C2xx device are dependent upon power dissipation, which, in turn, is dependent on power-supply current. When making thermal management calculations, several considerations must be made that relate to the manner in which the power-supply current contributes to power dissipation and to the 'C2xx thermal characteristics' time constant. Depending on sources and destinations of current on the device, some current contributions to I_{DD} do not constitute a component of power dissipation in the 'C2xx. Consequently, if the total current flowing into V_{DD} is used to calculate power dissipation at a given value of V_{DD}, excessive values for power dissipation are obtained.

Power dissipation is defined as:

$$P = I \times V$$

where	P	is power
	I	is current, and
	V	is voltage

If device outputs are driving any DC load to a logic-high level, only a minor contribution is made to power dissipation because CMOS outputs typically drive to a level within a few tenths of a volt of the power supply rails. If this is the case, these current components should be subtracted out of the total supply current value, and their contribution to power dissipation calculated separately and then added to the total power dissipation (see Figure 7). If this is not done, calculating the current resulting from driving a high logic level into a DC load gives unrealistically high power dissipation values. The error occurs because the currents resulting from driving a logic-high level into a DC load appear as a portion of the current used to calculate power dissipation due to V_{DD} at a given value.

Furthermore, external loads draw output current only when outputs are being driven high, because when outputs are in a logic-low state, the device is sinking current supplied from an external source. Therefore, the power dissipation due to outputs being driven low does not have a contribution through I_{DD}, but does contribute to power dissipation with a magnitude of:

$$P = V_{OL} \times I_{OL}$$

where V_{OL} is the low-level voltage, and

I_{OL} is the current being sunk by the output as shown in Figure 7.

The power dissipation component due to outputs being driven low should be calculated and added to the total power dissipation.

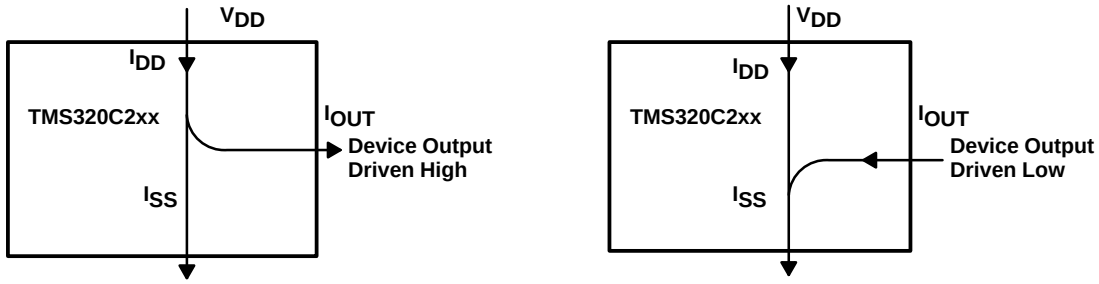

Figure 7. Device I/O Currents

When outputs with DC loads are switched, the power dissipation components from outputs being driven high and outputs being driven low are averaged and added to the total device power dissipation. Calculating power components due to DC loading of the outputs should be made separately for each independent and unique period of device activity before average power is calculated.

When using power dissipation values to determine thermal management, the average power should be used unless the time duration of the individual periods of device activity is long. The thermal characteristics of the 'C2xx package are exponential in nature with a time constant on the order of several minutes; therefore, when the device is subjected to a change in power, the package requires several minutes or more to reach thermal equilibrium. If the time duration of periods of device activity exhibiting high power dissipation values is short (on the order of a few seconds or less), in comparison to the package thermal characteristics' time constant, the average power, calculated in the same manner as the average current described previously, should be used.

Maximum device temperature should be calculated on the basis of actual time duration for the periods of device activity involved. For example, if a particular device activity lasts for twelve minutes, the device essentially reaches thermal equilibrium due to the total power dissipation during the period of device activity.

Note that the average power should be determined by calculating the power for each period of device activity (including all considerations described above) and performing a time average of these values, rather than simply multiplying the average current by V_{DD}, as determined in the previous subsection.

When the average power has been determined, specific device temperature calculations can be made using the thermal impedance characteristics for the type of package being used.

SYSTEM DESIGN CONSIDERATIONS FOR MINIMIZING POWER DISSIPATION

There are several issues that can be considered in the design process to reduce power consumption of a particular ’C2xx application. Although some of these issues have already been covered in this report, they are included here for convenience.

System Clock and Switching Rates

The power consumption of the ’C2xx device is directly proportional to the system clock (CLKOUT1) switching speed. If the clock speed doubles, the current doubles. Obviously, power can be saved by operating the device at the lowest clock speed possible that still meets the specifications for the device and the requirements of the application.

As the clock speed increases, the current increases proportionally, but the time required to execute the same operation decreases proportionally. If the clock speed doubles, twice the current is required but half of the time is required for the same operation. For example, consider a section of code that runs for 500 clock cycles at 1.2 mA per MHz. At a CLKOUT1 speed of 10 MHz, the code requires 12 mA for 50 μs (500×50 ns cycles). If the clock speed is doubled to 20 MHz, the current required doubles to 24 mA but the time duration required for the operation is cut in half to 25 μs. This behavior is illustrated in Figure 8. So the energy required to complete the operation is the same since it only depends on the number of internal logic state transitions the device makes.

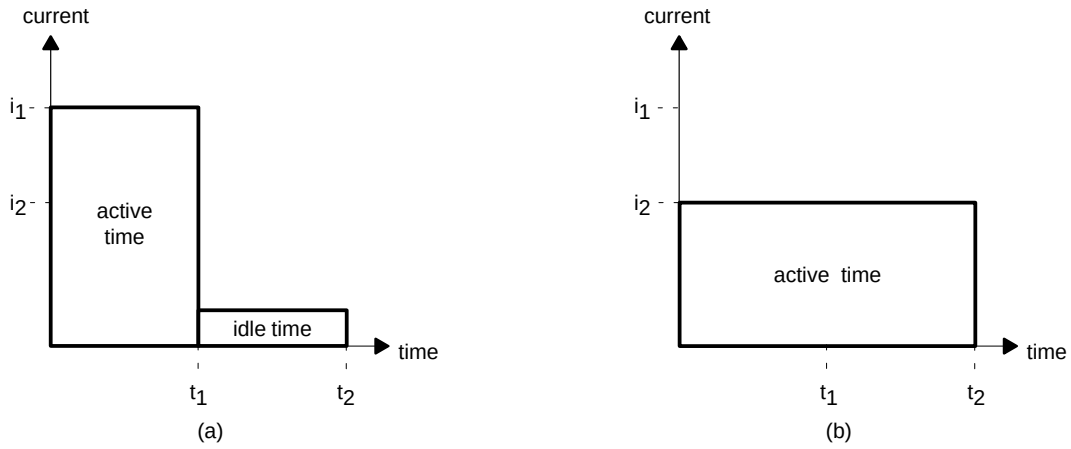

Figure 8. Algorithm Current Use vs. Clock Speed

It would appear that there is no difference between operating the device at the lowest speed possible for the application and at the highest speed supported by the device since the energy required by the algorithm is the same. So why not operate the device at its highest speed even if all of the MIPS capacity is not needed?

Consider the case where the algorithm is completed quickly and the remaining time is spent in a low-power mode (IDLE)—as shown in Figure 8(a), and the case where the system clock is slowed down so the algorithm requires all of the time available—as shown in Figure 8(b). In both cases, the energy required to perform the algorithm is the same. However, in case (a), power is also consumed for the rest of the time in the IDLE mode. This does not happen in case (b) because there is no time spent in IDLE mode. For this reason, if the application does not require the entire MIPS capability of the device, it is more power-efficient to slow down the system clock to minimize the IDLE time.

CLKOUT1 Switching

Some ’C2xx devices include a power-conservation feature that provides the option to disable external activity on the CLKOUT1 pin. The CLKOUT1 pin is provided as a master processor clock reference that can be used by external devices to synchronize with the ’C2xx CPU. In a system where there are no external devices that need a clock reference, there is no need to consume power operating the CLKOUT1 pin driver circuitry. In this case, the CLKOUT1 pin can be disabled so that it no longer consumes power. The internal processor clock remains unaffected.

The CLKOUT1 pin is disabled by setting the CLKOUT1 bit in the CLK register. If this bit is 1, CLKOUT1 is inactive and no signal reaches the CLKOUT1 pin. If the bit is 0, the clock signal is available at the pin and it operates normally.

At 5 V, the CLKOUT1 pin requires **0.17** mA per MHz (6.8 mA at 40 MIPS).

At 3 V, the CLKOUT1 pin requires **0.09** mA per MHz (3.6 mA at 40 MIPS).

This amount of current can be saved if the CLKOUT1 output is disabled. Refer to the *TMS320C2xx User’s Guide* (literature number SPRU127B) for more detail on which devices support CLKOUT1 switching.

Stopping the Internal Processor Clock

Many applications—such as pagers, cellular telephones and automotive applications—are battery-operated and are consequently very sensitive to power consumption. When these portable devices are not in use, the power being drawn from the batteries can be of critical importance. Although the IDLE mode mentioned above significantly reduces the processor power use, it does not halt the operation of the peripherals (timer, serial ports) and it does not stop the processor clocks. When an application is turned off, it may be desirable to stop the processor clock to achieve the absolute lowest power consumption. Since the ’C2xx family of processors is designed using a fully static CMOS technology, stopping the processor clock is possible without any corruption of internally stored data as long as adherence to the timing specifications in the data sheet is maintained.

Several conditions are necessary to safely stop the input clock (CLKIN) and achieve the absolute lowest device power consumption:

- The processor must be in IDLE mode.
- Divide-by-2 clock mode must be selected.
- CLKIN must be synchronously stopped in a logic-high state while meeting the rise/fall time maximum and pulse duration minimum specifications as indicated in the data sheet.
- All unused pins configured as inputs must be driven high or pulled-up using 10-kΩ resistors (with the exception of $\overline{\text{TRST}}$, which has an internal pull-down and should be left unconnected).

Under these conditions, the device consumes current in the nanoampere range. The conditions described above must be used for proper operation of the processor. If these conditions are not met, the following may occur:

- If the clock is stopped when the processor is not in IDLE mode, or if the input clock timing fails to meet the data sheet specifications, data corruption and/or erratic operation may result.
- If the clock is stopped in one of the PLL clock modes, the PLL continues to attempt to track the input clock (which is not present) and current use varies as it does so, even up to the milliampere range. The input clock can be safely stopped in the PLL clock mode (after IDLE mode has been

executed), but this does not result in the lowest power state. The PLL also requires a transitory phase to resyncronize after the input clock is restarted. The device should not be removed from IDLE mode until the transitory phase has been completed. Refer to the data sheet for more detail about the PLL transitory phase.

- If CLKIN is held low, the device consumes from 70−250 μA in divide-by-2 clock mode.
- Any unused inputs left floating may drift to an intermediate voltage between the power rails and cause internal buffer levels to float in transition regions. This causes current consumption and the device current increases with each input that is not controlled.

On-Chip vs. Off-Chip Memory

On-chip memory requires less power because the external memory interface is not driven during internal accesses. Minimizing accesses to external memory space lowers the device current requirement.

On-Chip ROM vs. On-Chip RAM

Use of internal ROM requires less power than use of internal RAM. Code execution from internal ROM requires about 10% less CPU current than the same code executing from internal SARAM.

Capacitive Loading of Outputs

Increased capacitive loading on device outputs increases the current required to drive the output pins. Minimizing this loading minimizes the current required to operate these pins.

Address Visibility

When address visibility is enabled, addresses are passed to the external address bus even during internal memory accesses. This feature is very useful primarily as a development and debugging tool, but it should be disabled when debug is complete to minimize activity on the external memory interface.

DC Loading of Outputs

DC loading of outputs due to TTL or other sources should be minimized to conserve power.

Power-Down Mode

When device CPU activity is not necessary, the device should be placed in IDLE mode to save power. This is achieved by executing the IDLE instruction or by a logic-low input on the $\overline{\text{HOLD}}$ pin. In IDLE mode, the internal clocks to the CPU are shut off. Only the peripherals and the PLL remain active. This state conserves a considerable amount of power. The IDLE state is easily exited by the occurrence of an internal or external interrupt. See the *TMS320C2xx User's Guide* (literature number SPRU127B) for more information.

POWER CALCULATION EXAMPLE

In order to illustrate the techniques described earlier in this report, this section presents a finite impulse response (FIR) filter as a simple example of a power dissipation calculation. The program reads in input samples from an external data memory location at a rate of 8 kHz. After each input sample is read, the program performs a FIR filter and then writes all of the current data samples in the filter to external data memory. When this operation is complete, the processor goes into the IDLE mode while waiting for the next timer interrupt, which indicates it is time to read another input sample. The entire process then repeats. A listing of the program is provided in Appendix A.

System Environment

The TMS320C209 processor is running at 20 MIPS (20 MHz CLKOUT1) in PLL multiply-by-2 clock mode. The V_{DD} supply voltage is 5 V. The external address bus, data bus and relevant control signal outputs ($\overline{WE}$, $\overline{STRB}$, $\overline{RD}$) are loaded with 20 pF capacitance per pin. The external data bus is pulled-up through 10-kΩ resistors. Zero wait-states are assumed for all external memory accesses.

Timer Configuration

Since the application reads input data samples at a rate of 8 kHz, the timer is configured to generate a timer interrupt every 125 µs. To generate this time period, the timer is loaded with a count of 2500 clock cycles as the period (20 MHz/8 kHz = 2500). The timer interrupt is unmasked so that it can be used to exit IDLE mode, but interrupts are disabled (INTM = 1) so there is no need to generate a timer interrupt service routine.

Algorithm Partitioning

The example application is separated into three sections and the power dissipation calculated for each section separately. These sections are chosen based on program activity. In the first section, the device reads a single input data word and runs the FIR filter. In the second section, the processor writes a table of the current data sample values in the filter to an external memory location. In the third section, the processor enters IDLE mode and waits for the next timer interrupt. When the interrupt occurs, the process is repeated with a new data sample. For each of these three sections, the current contribution due to the internal CPU activity, the internal peripheral activity, the external-memory interface activity and the external loads are calculated. After the current contribution of each of the sections is calculated, these sections are time-averaged to determine the over-current use for the application.

Timer Activity

The timer runs during all of the algorithm partitions mentioned above so its contribution to each of the sections is equal. The current use of the timer can be estimated from Figure 5. The timer is updating once every 2500 CLKOUT1 cycles. At this rate, the estimated current use for the timer is 0.013 mA per MHz of CLKOUT1.

Timer current use: **0.013** mA per MHz of CLKOUT1 for 100% of the execution time

Data Input and the FIR Section

The first section occurs after the timer interrupt has indicated that it is time to read an input data sample. A single input data word is read from external data memory at data memory address 2000h and copied into a working buffer at internal data memory address 0300h. Then the FIR filter routine runs. The filter is 256 taps and the coefficients are assumed to have already been stored in a table called "COEFF". When the filter routine completes, the result is stored at data memory variable "result". Reading the input samples

and storing the result requires very few clock cycles (6 cycles) compared to the FIR filter calculation (258 cycles), so the current use for this section is primarily due to the filter calculation and the other instructions can be considered negligible.

When the MACD instruction is repeated, it executes in single cycles unless limited by the memory configuration. In this case, the coefficients are stored in on-chip SARAM and the data samples are stored in on-chip DARAM. This memory configuration allows the CPU to complete one MACD per CLKOUT1 cycle. Table 2 indicates that a repeated MACD at 5 V requires approximately 2.9 mA per MHz of CLKOUT1. If a different memory configuration, which caused wait states to occur during the MACD execution, were used, this current value would have to be scaled according to the speed of actual execution.

Since both the coefficients and the data samples have been stored on-chip, the external memory interface is not active during this section and does not consume power, so the following components contribute to the current required during the data input / FIR section:

Internal CPU contribution:	**2.9** mA per MHz of CLKOUT1
External interface contribution:	**0**
External loads contribution:	**0**
Total:	**2.9** mA per MHz of CLKOUT1

External Table-Write Section

In the second section, the processor writes a table of data samples from the working buffer in DARAM out to external data memory at address 3000h. This section requires only a repeated BLDD instruction (block move from data memory to data memory). Since this instruction is repeated 256 times, it dominates the current use in this section and the current contribution of the other instructions can be considered negligible.

Since all memory in this example requires zero wait states, the BLDD instruction requires two cycles of each execution because external data writes require a minimum of two cycles. Details of the timing requirements and behavior of an external memory interface write cycle are shown in the data sheet.

First, the internal CPU contribution to the operating current is calculated. From Table 2, a BLDD instruction requires approximately 1.9 mA per MHz of CLKOUT1. This value, however, was measured for single cycle (on-chip to on-chip) transfer. Since the destination for the data in this example is in external memory, the instruction requires two cycles per execution, because external write cycles require two cycles per write. Two cycles per write instead of one causes the BLDD to take two cycles to execute instead of one. Since the execution speed is cut in half, the current required to perform this instruction is also cut in half to 0.95 mA per MHz of CLKOUT1.

Internal CPU contribution:	**0.95** mA per MHz of CLKOUT1

During the repeated BLDD instruction, the external address bus is incrementing. As shown previously, an incrementing address bus requires an amount of current equivalent to toggling two lines of the bus at the same speed as the least significant bit. Since the external writes take two cycles, the external addresses only change every two cycles, so the fastest toggling address line can change states and change back in a minimum of four clock cycles; therefore, the address bus lines toggle at a speed that is one-fourth of CLKOUT1.

The current required to increment the external address bus at 5 V was previously shown to be 0.4 mA per MHz of the switching frequency of the fastest address line (least significant bit). Since the switching frequency of the fastest address line is one-fourth of CLKOUT1 and the current required to increment the bus is:

$0.4 \times (1/4) =$ **0.1** mA per MHz of CLKOUT1

The current required to run the loads on the address bus is calculated as:

$2CVF = 2 \times (20\ pF) \times (5\ V) \times (1/4 \text{ x CLKOUT1}) =$ **0.05** mA per MHz of CLKOUT1

The current required to drive the external data bus varies depending on the data pattern being written. If the data is unchanging from cycle to cycle, no current would be required because no nodes are changing state. If the worst-case data happened (AAAAh, 5555h, AAAAh, ... for example), the current required would be 3.2 mA per MHz of switching frequency, as determined previously. For the same reasons as specified above for the address bus, the switching frequency of the external data bus is one-fourth of CLKOUT1. So the worst case current required to operate the external data bus is calculated as:

$(3.20) \times (1/4) =$ **0.8** mA per MHz of CLKOUT1 (worst case)

Since it is unlikely that all data lines change on every write, an average value can be chosen between best case (no lines changing) and worst case (16 lines changing). This average value assumes that 8 lines change on each word. In this case, the current required to operate the external data bus would be:

$8 \times (0.2) \times (1/4 \text{ of CLKOUT1}) =$ **0.4** mA per MHz of CLKOUT1 (average case)

The current required to drive the external loads on the data bus under the same conditions is calculated as:

$8CVF = 8 \times (20\ pF) \times (5\ V) \times (1/4 \text{ x CLKOUT1}) =$ **0.2** mA per MHz of CLKOUT1

The control pins operating during the repeated external writes are $\overline{WE}$ and $\overline{STRB}$. Although other control pins are active before and after the repeated BLDD execution, they do not change during its execution. Refer to the Memory Interface External Write Timing diagram in the TMS320C2xx data sheet (literature number SPRS025) for more detailed information. As determined previously, the current required to drive the control pins is 0.17 mA per MHz of switching frequency. $\overline{WE}$ and $\overline{STRB}$ both switch at a frequency one-half of CLKOUT1, so the current required to operate these control pins is calculated as:

$(2\ pins) \times (0.17) \times (1/2 \times CLKOUT1) =$ **0.17** mA per MHz of CLKOUT1

The current required to drive the external loads on the control pins is calculated as:

$2CVF = (2\ pins) \times (20\ pF) \times (5\ V) \times (1/2 \times CLKOUT1) =$ **0.1** mA per MHz of CLKOUT1

The sum of these calculations indicates the overall supply current required during the external table write section:

Internal CPU requirement:	**0.95** mA per MHz of CLKOUT1
External address bus requirement:	**0.10** mA per MHz of CLKOUT1
External address load requirement:	**0.05** mA per MHz of CLKOUT1
External data bus requirement:	**0.40** mA per MHz of CLKOUT1
External data load requirement:	**0.20** mA per MHz of CLKOUT1
External control pins requirement:	**0.17** mA per MHz of CLKOUT1
External control load requirement:	**0.10** mA per MHz of CLKOUT1
Total:	**1.97** mA per MHz of CLKOUT1

IDLE Section

During the IDLE section, the processor enters IDLE mode and waits for a timer interrupt. When the timer interrupt occurs, the processor continues program execution with the data input and FIR filter section. It does not perform a timer interrupt service routine because the interrupts were disabled. In IDLE mode, the processor requires 0.5 mA per MHz of CLKOUT1 at 5 V when running in PLL multiply-by-2 mode.

IDLE requirement: **0.5** mA per MHz of CLKOUT1

Determining the Time-Averaged Current

To determine the total current required by the device, the current required by each section of the algorithm is time-averaged. For this calculation, the time required to execute each section of the code must be determined. The data input/FIR section requires 265 cycles to execute. At 20 MHz CLKOUT1, execution of this section takes 13.25 μs. The table write section requires 522 cycles to execute. Execution of this section takes 26.1 μs. The entire cycle repeats every 125 μs, so the remainder of this time (85.65 μs) is spent in the IDLE section. The contribution of the timer current is present 100% of the time.

The execution times can be expressed as a percentage of the total time as:

Data input/FIR section: (13.25 μs/125 μs) = 10.6% of total time
Table write section: (26.1 μs/125 μs) = 20.8% of total time
IDLE section: (85.65 μs/125 μs) = 68.6% of total time

The current required for each section is determined by multiplying the “mA per MHz” current by the CLKOUT1 speed:

Data input/FIR section: (2.9) × (20) = 58.0 mA
Table write section: (1.97) × (20) = 39.4 mA
IDLE section: (0.50) × (20) = 10.0 mA
Timer: (0.013) × (20) = 0.26 mA

Finally, the time-averaged current is determined by multiplying the current required for each of the sections by the fractional time the section is executing, and then summing each of those contributions.

$$I_{total} = I_1t_1 + I_2t_2 + I_3t_3 + I_4t_4$$
$$= (58.0)\,(0.106) + (39.4)\,(0.208) + (10)\,(0.686) + (0.26)\,(1)$$
$$= \mathbf{21.5\ mA}$$

Experimental Results

In order to confirm the values for current calculated in the example, the actual power supply current for this sample program was measured using the test setup previously described in this report. The actual measured current values are included below:

Data input/FIR section:	59.8 mA
Table write section:	40.7 mA
IDLE section:	11.1 mA
Overall program:	**22.4 mA**

A listing of the sample program and a photograph of the actual current waveforms observed are included in Appendix A.

Summary and Conclusion

The power-supply current requirement for the TMS320C2xx DSPs cannot be expressed simply in terms of operating frequency, supply voltage, and output capacitance. A more complete specification, one based on device activity, must be used to determine an accurate power-supply current requirement. This application report presents the information necessary to accurately analyze power supply current requirements. These requirements are based on the knowledge of various periods of device activity and their operation of the TMS320C2xx in terms of internal and external activity.

The power supply current requirements for the TMS320C2xx DSPs depend on system parameters as well as device activity. Dependencies related to system parameters include operating frequency, supply voltage, operating temperature, and output capacitance. The components related to device activity include CPU activity, peripheral activity, and external bus operations.

Taking into account the current effects and dependencies involved in analysis of device power dissipation, system design may be performed proactively to minimize device and system power dissipation. With the combination of 40 MIPS performance and low-power dissipation design, the TMS320C2xx family of DPSs is an ideal solution for power-sensitive applications.

APPENDIX A. EXAMPLE PROGRAM LISTING

```
       .mmregs
       .global   init,temp,temp2,result
       .bss      temp,1
       .bss      temp2,1
       .bss      result,1
       .text
*****************************
*   Initialization
*****************************
*
init
   setc   intm            ;disable interrupts
   ldp    #0              ;set data page to 0
   lacl   #0
   sacl   IMR             ;mask all interrupts
   lacl   #0
   sacl   GREG            ;GREG = 0000 – all memory local
   ldp    #temp           ;set data page
   lacl   #30h            ;set TCR control word
   sacl   temp            ;store control word
   out    temp,0fffch     ;send to TCR
   lacc   #09c4h          ;set timer count = 2500 (8 kHz at 20 MHz CLKOUT)
   sacl   temp
   out    temp,0fffdh     ;write control word to PRD
   lacl   #00h            ;AVIS = 0, ISWS=PSWS=DSWS=0 wait states
   sacl   temp            ;store control word
   out    temp,0ffffh     ;send to wait state generation register
   clrc   cnf             ;configure B0 in data memory space
   splk   #0008h,IMR      ;unmask timer interrupt
   lacl   #20h            ;start timer
   sacl   temp
   out    temp,0fffch
   mar    *,7
```

```
****************************************
*      IDLE section                    *
****************************************
begin
   splk   #0ffffh,IFR  ;clear IFR
   lar    ar7,#0300h
   idle                ;wait for timer interrupt
****************************************
*      Data input and FIR filter section   *
****************************************
filter
   bldd   #2000h,*     ;read input data
   lar    ar7,#03ffh
   lacl   #0           ;clear accumulator
   rpt    #255         ;RUN 256-tap FIR filter
   macd   coeff,*-
   apac
   sach   result       ;store result
****************************************
*      External table write section     *
****************************************
write
   lar    ar7,#0300h
   rpt    #255         ;write filter samples to external memory
   bldd   +, #03000h
   b      begin        ;repeat
```

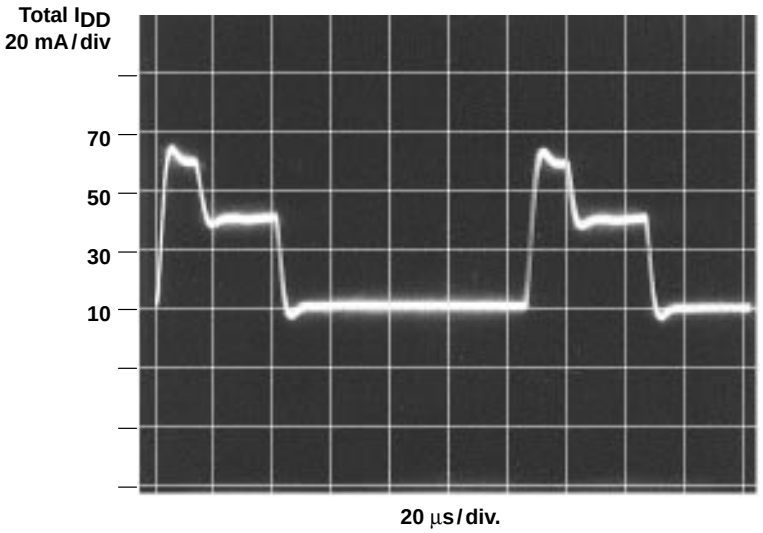

Figure 9. Actual Measured Current